Conditional Attribute-Based Proxy Re-Encryption and Its Instantiation

Lisha Yao, Jian Weng ✉, Bimei Wang

College of Cyber Security, Jinan University, China
{yaolishaqh,cryptjweng,bimeiwang1}@gmail.com

Abstract. In attribute-based proxy re-encryption (AB-PRE) and attribute-based conditional proxy re-encryption (AB-CPRE) systems, the proxy transforms a ciphertext associated with policy f to a ciphertext associated with policy g or transforms a ciphertext for delegator satisfying a fine-grained condition to a ciphertext for delegatee. However, such PRE schemes have found many practical applications requiring fine-grained access control while keeping flexible delegation. Unfortunately, the existing PRE schemes are impossible to handle simultaneously with the above scenarios. In this work, we introduce the notion of conditional attribute-based proxy re-encryption (CAB-PRE), which enables a proxy only to transform a ciphertext associated with policy f meeting the special delegation requirements by delegator to a ciphertext associated with policy g. We formalize its honestly re-encryption attacks (HRA) security model that implies CPA-secure, giving a concrete CAB-PRE scheme based on learning with errors (LWE) assumption. Finally, we show that CAB-PRE implies AB-PRE and AB-CPRE notions, and propose their constructions.

Keywords: conditional attribute-based proxy re-encryption, honestly re-encryption attacks, learning with errors

1 Introduction

As for everyone familiar with public-key cryptography, certainly don't be unfamiliar with proxy re-encryption [4], a cryptographic primitive proposed by Blaze, Bleumer, and Strauss. It is an efficient and essential technique to reach secure communication with others. Recall a regular scene in the company: Alice is a manager who may handle several files every day. Suddenly, one day she fell ill and was unable to work. She called her assistant Bob to complete some businesses. Without loss of generality, we make some reasonable assumptions about Alice. First, these files that Alice received are ciphertext forms encrypted from her public key. Second, Alice doesn't want to give out her secret key. Naturally, these waiting files should be re-encrypted to Bob via a re-encryption key, so that he can handle them, in which transformation proceeds by a semi-trust proxy. Briefly speaking, proxy re-encryption (PRE) enables a proxy to transform a ciphertext under one's public key to a ciphertext under the other's public key while remains message unaltered.

If the public key is considered an explicit and public identity for every user, PRE seems to point in a kind of deterministic direction. However, it is unnecessary that require intended recipients in many scenarios. Attribute-based proxy re-encryption (AB-PRE) [13, 17, 24] has such functionality, which is adapted from attribute-based encryption (ABE) and could update access policy by re-encryption procedure. Turn to the above case, applying AB-PRE, Alice could convert her jobs under an access policy to new jobs under another access policy, enabling more assistants to complete instead of point-to-point transformation.

As we all know, the shortcoming of PRE (or AB-PRE) is the lack of flexible delegation, which means Alice's all ciphertexts will be transformed without discrepancy once generating a re-encryption key. It is impractical and insecure to start with common sense. Fortunately, there exists a notion of conditional proxy re-encryption (CPRE) [27], which is an elegant solution: one makes a "decision" on ciphertext whether it to be or not to be re-encrypted through appending conditions. A trivial condition in the ciphertext is an equivalent match, for which a ciphertext could be re-encrypted if and only if (i.e., iff) $w' = w$, where w and w' are conditions related to the ciphertext and the re-encryption key, respectively. This condition for increased expressiveness can be improved, such as attribute-based conditional proxy re-encryption (AB-CPRE) [16,28] for different predicates like puncturing, bit-fixing, and inner-product.

Although the AB-PRE and AB-CPRE schemes focus on different goals, the underlying syntax has much in common, and a synergistic effect when constructing both primitives could arise. From a functional and security perspective, the desirable properties of that fine-grained access policy and flexible delegation can be realized both once in a single scheme. By applying ABE as underlying construction and increasing conditions to control ciphertext transformation, it may well happen that a part of ciphertexts associated with policy f is re-encrypted to the ciphertexts associated with policy g whenever the conditions are satisfied. For the sake of security, it is sufficient to prove secure under honestly re-encryption attacks (HRA) [9] (implies CPA-secure). Note that we restrict our attention to unidirectional, single-hop proxy re-encryption.

1.1 Related Work

At present, there have been many results about AB-PRE [13, 15] and AB-CPRE [18, 28] with various functionalities based on classical number-theoretic assumptions. In this section, we mainly review the related development of lattice-based constructions.

Based on the KP-ABE scheme introduced by Boneh et al. [5], there are two selectively CPA-secure PRE schemes. Luo et al. [17] proposed a first multi-hop LWE-based (KP)AB-PRE scheme. Their transformation approach is intuitive, sampling $\mathbf{R}_{f\to g} \leftarrow \mathsf{SamplePre}(\mathbf{A}\|\mathbf{B}_f, \mathbf{T}_{\mathbf{A}\|\mathbf{B}_f}, \mathbf{A}\|\mathbf{B}_g, \sigma)$, which could further extend to multi-hop settings under the secret key $\mathbf{T}_{\mathbf{A}\|\mathbf{B}_g}$. Liang et al. [16] proposed two LWE-based unidirectional (KP)AB-CPRE schemes consisting of single-hop and multi-hop by applying the key switching technique, which enables an access policy as a condition to realize fine-grained delegation control. In addition to CPA security, the scheme also proved key privacy property.

However, Cohen [9] demonstrated that CPA-secure in PRE is inadequate to protect the delegator's secret key from a single honestly re-encrypted ciphertext. Their work introduced a new security definition: HRA, a strengthening of CPA security. They showed that if a CPA-secure scheme has re-encryption simulatability, then it is also HRA-secure. Subsequently, Fuchsbauer et al. [12] studied CPA and HRA in adaptive settings and gave requirements for their satisfaction. Döttling et al. [11] presented the concept of universal proxy re-encryption (UPRE) and defined HRA security for it. Recently, Susilo et al. [24] formalized a notion of HRA-secure (KP)AB-PRE. They constructed a single-hop unidirectional (KP)AB-PRE scheme and proved selectively secure in the standard model based on the LWE problem. Concretely, it was also inspired by [5] and adopted the key switching technique [6] to design that scheme. From its security proof, we observe that it always needs to generate the secret key when answering re-encryption queries of which the distinction just is applying different trapdoors.

1.2 Our Contributions

We are motivated by the concept of combined scheme [21,26], which incorporates the public-key encryption scheme and the signature scheme via a single keypair, enabling them to enjoy both privacy and authenticity. Although it looks nothing to do with PRE, reminiscent of two primitives of AB-PRE and AB-CPRE. The former has a fine-grained access policy, and the latter has flexible delegation, so

Why don't we merge their functionalities to achieve a more expressive PRE scheme?

We give an affirmative answer to the question above by providing a notion of CAB-PRE and its instantiation over lattices. On a high level, our contribution is as follows:

- We propose a concept of conditional attribute-based proxy re-encryption (CAB-PRE) and its HRA-secure model. It is worth noting that we strengthen the original definition of HRA-secure for PRE [9]. In particular, the previous one only permits an adversary to query the re-encryption oracle from a corrupt user to an honest user under the honestly generated ciphertexts. We give a slightly stronger definition that allows an adversary to query the re-encryption key oracle in a meaningful way in a case that the conditions are unsatisfaction.
- We give a construction of CAB-PRE satisfying adaptively HRA-secure, based on the hardness of learning with errors (LWE) problem, for which we inspired from a ciphertext-policy ABE (CP-ABE) scheme [25], key switching technique [6], and constrained pseudorandom function (cPRF) [10]. We focus on inner-product predicates that are constraint conditions in encryption procedure. We weakened the relationship between the secret key and the re-encryption key by changing the random matrix, which means it is possible to generate a re-encryption key without the related secret key.

- We show that CAB-PRE implies the AB-PRE and AB-CPRE schemes are also HRA-secure. The construction of AB-PRE is almost identical to CAB-PRE since their underlying structure is the same except for the delegation condition, which can be a solution for [24]'s open problem. From CAB-PRE to AB-CPRE, we adapt from "dual" public-key encryption [14] and use an access policy for $t-$CNF in place of inner-product predicates. Unfortunately, to answer the re-encryption queries, the proof of AB-CPRE remains has a problem generating the re-encryption key via secret key.

1.3 Technical Review

To instantiate our proposed notion, we adopt the CP-ABE scheme [25] and combine key switching technology [6] and cPRF for inner-product predicates [10] to construct a unidirectional, single-hop CAB-PRE scheme. By definition 3.1, CAB-PRE consists of seven algorithms: Setup, KeyGen, Enc_1, Enc_2, Dec, $\mathsf{ReKeyGen}$, ReEnc, in which Enc_1 denotes first-level ciphertext or re-encrypted ciphertext that transformed from the original ciphertext, and Enc_2 denotes second-level ciphertext or original ciphertext that directly encrypted from the encryption algorithm. In this work, the second-level ciphertext could be converted to first-level ciphertext when access policy and condition are satisfied. In particular, the decryption operation essentially has no difference for two types of ciphertexts other than the restrictions of noise bound. Observe that cPRF for inner-product predicates that expressibility is between that of $t-$CNF and NC^1 has a similar logic to the transformation condition in our construction. There is the perfect incorporation of key switching and inner-product predicates at the expense of the dimension expanding when generating a re-encryption key. However, it does not affect the final ciphertext size, which is still constant.

The details of CAB-PRE scheme and security proof are in Section 4. Roughly speaking, assume that P is a conforming cPRF for $t-$CNF predicates achieves single-key adaptively secure, has properties of gradual evaluation and key simulation which play an important role in construction and security proof. In [25] scheme, the decryption succeeds conditioned on $f(x) = 1 \land r \neq r'$ where $r \leftarrow \mathsf{P.Eval}(\sigma, x)$, and $r' \leftarrow \mathsf{P.ConstrainEval}(s_f, x)$ with $s_f \leftarrow \mathsf{P.KeySim}(P.pp, f)$. Let C_β be an inner-product predicate, and use vector decomposition (Definition 2.8) to process the predicate vector prevents noise explosion. We add inner-product vectors in Enc_1 and $\mathsf{ReKeyGen}$ phases such that the ciphertexts meet requirements and can be re-encrypted whenever performing re-encryption operations. Note that we provide a weak form of anonymity, namely attribute hiding, vector α embedded in $\mathbf{s}^T(\mathbf{D} + \mathbf{W} \otimes \mathsf{P2}(\alpha)^T) + \mathbf{e}_3^T$ remains hidden since $\mathbf{s}, \mathbf{e}_3$ are secrets and $\mathbf{D}, \mathbf{W}$ are uniformly random. In re-encryption key generation, we select a function $g \in \mathcal{F}$ and let $\mathbf{H} = \mathbf{B} \| (\mathbf{A}_g - s_g \otimes \mathbf{G})$. We then reconstruct $\mathbf{V}' = (\mathbf{V} + \mathbf{D}) \cdot \mathsf{BD}(\beta)$ as a random matrix. By using the key switching technique, we capture the re-encryption key.

To prove the adaptive HRA security, we introduce two randomized algorithms: $\mathsf{Semi}-\mathsf{KeyGen}_1$ and $\mathsf{Semi}-\mathsf{KeyGen}_2$ are crucial points for answering re-encryption queries so that we can always simulate a re-encryption key without the help of the secret key. However, we also need the master secret key when the adversary has decryption capability. Concretely, we first sample a short vector $\mathbf{k}$ (i.e., $(\mathbf{B}\|\mathbf{A}_{x,r})\mathbf{k} = \mathbf{V}$) and output an extended trapdoor in the KeyGen algorithm. We further sample a short vector $\mathbf{d}$ (i.e., $(\mathbf{B}\|\mathbf{A}_{x,r})\mathbf{d} = \mathbf{V}'$) based on the extended trapdoor to generate a re-encryption key. Due to $\mathbf{d} \neq \mathbf{k}$, we can circumvent the secret key and simulate a re-encryption key straightway derived from the master secret key. There is a trick that applies some algorithms with "symmetric" properties, such as $\mathsf{SamlpleLeft}$ and $\mathsf{SamlpleRight}$. Finally, we insert LWE instance $(\mathbf{B}\|\mathbf{V}, \mathbf{u}_0\|\mathbf{u}_2')$ into challenge ciphertext $(\mathbf{u}_0, \mathbf{u}_2 = \mathbf{u}_2' + \mu_b(\mathbf{0}\|\mathbf{g}^T))$ enabling the adversary cannot to distinguish.

1.4 Organization

In section 2, we introduce some related cryptographic primitives and lattice background. In section 3, we give the definitions and security model of CAB-PRE. In section 4, we present an instantiation scheme based on lattices and prove its HRA-secure. In section 5, we propose AB-PRE and AB-CPRE constructions via CAB-PRE. At last, we conclude our paper in section 6.

2 Preliminaries

2.1 Constrained PRF, Conforming cPRF and $t-$CNF Predicates

Definition 2.1 (Constrained PRF [10, 25]). let $\mathcal{F}$ be a family of functions with domain $\{0,1\}^l$ and range $\{0,1\}$. A constrained pseudorandom function (cPRF) for $\mathcal{F}$ is defined by a tuple of probabilistic polynomial-time (PPT) algorithms $\Pi_{\mathsf{CPRF}} = (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{ConstrainEval})$ where:

- $\mathsf{Setup}(1^\lambda) \to (pp, msk)$: The setup algorithm takes as input the security parameter 1^λ, outputs a public parameter pp and a master secret key msk.
- $\mathsf{Eval}(msk, x) \to y$: The evaluation is a deterministic algorithm which takes as input the master secret key msk and a bit-string $x \in \{0,1\}^l$, outputs $y \in \{0,1\}^k$.
- $\mathsf{Constrain}(msk, f) \to sk_f$: The constrained key generation takes as input the master secret key msk and a function $f \in \mathcal{F}$ specifying the constraint, outputs a constrained key sk_f.
- $\mathsf{ConstrainEval}(sk_f, x) \to y'$: The constrained evaluation is a deterministic algorithm which takes as input a constrained key sk_f and a bit-string $x \in \{0,1\}^l$, outputs $y' \in \{0,1\}^k$.

Correctness. We say a cPRF scheme Π_{CPRF} is correct if for all $f \in \mathcal{F}$ and $x \in \{0,1\}^l$ such that $f(x) = 1$, we have $\mathsf{Eval}(msk, x) = \mathsf{ConstrainEval}(sk_f, x)$ where $(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda)$ and $sk_f \leftarrow \mathsf{Constrain}(msk, f)$.

Pseudorandomness. The single-key adaptive security of a cPRF is defined formally by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

- **Setup**: At the beginning of the game, the challenger $\mathcal{C}$ prepares $(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda)$ and sends pp to $\mathcal{A}$.
- **Phase 1**: $\mathcal{A}$ can adaptively make two types of queries:
  - Evaluation Queries: Upon a query $x \in \{0,1\}^l$, the challenger evaluates $y \leftarrow \mathsf{Eval}(msk, x)$ and returns y to $\mathcal{A}$.
  - Constrained Key Queries: This oracle can only be queried once. Upon a query $f \in \mathcal{F}$, the challenger computes $sk_f \leftarrow \mathsf{Constrain}(msk, f)$ and returns sk_f to $\mathcal{A}$.
- **Challenge**: $\mathcal{A}$ chooses a target bit-string $x^* \in \{0,1\}^l$. The challenger flips a coin $b \overset{\$}{\leftarrow} \{0,1\}$. If $b = 1$, it evaluates $y^* \leftarrow \mathsf{Eval}(msk, x^*)$. Otherwise, it samples $y^* \overset{\$}{\leftarrow} \{0,1\}^k$. Finally, $\mathcal{C}$ returns y^* to $\mathcal{A}$.
- **Phase 2**: $\mathcal{A}$ continues to make queries as same as Phase 1 with the restrictions that cannot query x^* in evaluation oracle and cannot query any circuit f enables $f(x^*) = 1$ in constrained key oracle.
- **Guess**: Eventually, $\mathcal{A}$ outputs b' as a guess for b.

The adversary $\mathcal{A}$ wins the game if $b' = b$, which advantage is at most $1/2 + \mathsf{negl}(\lambda)$.

Definition 2.2 (Conforming cPRF [25]). We call a cPRF scheme is conforming, except for correctness and pseudorandomness defined above, if the following properties hold.

Gradual Evaluation. Let $\mathsf{Constrain}$ (in addition to $\mathsf{Eval}, \mathsf{ConstrainEval}$) algorithm is deterministic. Fixing pp from $\mathsf{Setup}(1^\lambda)$, for any $f \in \mathcal{F}$ and $x \in \{0,1\}^l$ such that $f(x) = 1$, define the following circuits:

- $U_{\sigma \to x} : \{0,1\}^\lambda \to \{0,1\}^k$ takes as input msk and computes $\mathsf{Eval}(msk, x)$.
- $U_{\sigma \to f} : \{0,1\}^\lambda \to \{0,1\}^{l_f}$ takes as input msk and computes $\mathsf{Constrain}(msk, f)$.
- $U_{f \to x} : \{0,1\}^{l_f} \to \{0,1\}^k$ takes as input sk_f and computes $\mathsf{ConstrainEval}(sk_f, x)$.

Key Simulation. We require a PPT algorithm $\mathsf{KeySim}(pp, f) \to sk_f$ such that any adversary $\mathcal{A}$ has at most $1/2 + \mathsf{negl}(\lambda)$ probability of winning the following game against a challenger $\mathcal{C}$.

- **Setup**: $\mathcal{C}$ generates $(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda)$ and sends pp to $\mathcal{A}$.
- **Phase 1**: $\mathcal{A}$ makes (polynomial times) evaluation queries. For a bit-string $x \in \{0,1\}^l$, $\mathcal{C}$ returns $y \leftarrow \mathsf{Eval}(msk, x)$.
- **Challenge**: For a challenge constraint $f^* \in \mathcal{F}$, $\mathcal{C}$ samples $b \overset{\$}{\leftarrow} \{0,1\}$ and returns $sk_{f^*} \leftarrow \mathsf{Constrain}(msk, f^*)$ if $b = 0$, otherwise it returns $sk_{f^*} \leftarrow \mathsf{KeySim}(pp, f^*)$.
- **Phase 2**: Same as Phase 1 except that cannot query x in evaluation oracle so that $f^*(x) = 0$.
- **Guess**: $\mathcal{A}$ outputs a bit b'. $\mathcal{A}$ wins the game if $b' = b$.

In this work, we will use a class of predicates: $t-$ conjunctive normal form (t-CNF), where each clause is of constant locality. We give a definition as below.

Definition 2.3 ($t-$CNF Predicates [10, 25]). A $t-$CNF predicate $f : \{0,1\}^l \to \{0,1\}$ such that $t \leq l$ is a set of clauses $f = \{(T_i, f_i)\}_i$, where for all i, $T_i \subseteq [l]$, $|T_i| = t$ and $f_i : \{0,1\}^t \to \{0,1\}$. For all $x \in \{0,1\}^l$, a $t-$CNF predicate $f(x)$ is computed as

$$f(x) = \bigwedge_i f_i(x_{T_i})$$

where $x_{T_i} \in \{0,1\}^t$ is the bit string consisting of the bits of x in the indices of T_i. At last, a family of $t-$CNF predicates $\mathcal{F}$ is the set of $t-$CNF predicates with input length l.

2.2 Lattices, Discrete Gaussian, Bounded Distributions

Notations. We use bold symbols denote matrices or vectors and regular lowercase letters denote single elements. Let $(\cdot\|\cdot)$ (resp. $(\cdot;\cdot)$) denote the horizontal concatenation (resp. vertical concatenation) of vectors or matrices. For a distribution or set X, we write $x \overset{\$}{\leftarrow} X$ to denote the operations of sampling uniformly and randomly x according to X. Let $\models$ (resp. $\not\models$) denote the satisfied relationship (resp. dissatisfied relationship) between two conditions.

Matrix Norm. For a vector $\mathbf{u}$, let $\|\mathbf{u}\|$ denote its l_2 norm. For a matrix $\mathbf{R} \in \mathbb{Z}^{n \times m}$, let $\widetilde{\mathbf{R}}$ be the Gram-Schmidt orthogonalization of $\mathbf{R}$. we define the following matrix norms:

– $\|\mathbf{R}\|$ denote the l_2 length of the longest column of $\mathbf{R}$.
– $\|\mathbf{R}\|_\infty$ denote the maximum element in $\mathbf{R}$.

Note that $\|\mathbf{R}\|_\infty \leq \|\mathbf{R}\| \leq nm\|\mathbf{R}\|_\infty$ and that $\|\mathbf{RS}\|_\infty \leq m\|\mathbf{R}\|_\infty\|\mathbf{S}\|_\infty$.

Lattice. In this work, we will be using two kinds of integer lattices. For q prime, given $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{u} \in \mathbb{Z}_q^n$, denote:

$$\Lambda_q^\perp(\mathbf{A}) = \{\mathbf{v} \in \mathbb{Z}^m : \mathbf{Av} = \mathbf{0} \pmod{q}\},$$

$$\Lambda_q^{\mathbf{u}}(\mathbf{A}) = \{\mathbf{v} \in \mathbb{Z}^m : \mathbf{Av} = \mathbf{u} \pmod{q}\}.$$

Observe that if $\mathbf{t} \in \Lambda_q^{\mathbf{u}}(\mathbf{A})$ then $\Lambda_q^{\mathbf{u}}(\mathbf{A}) = \Lambda_q^\perp(\mathbf{A}) + \mathbf{t}$ and hence $\Lambda_q^{\mathbf{u}}(\mathbf{A})$ is a shift of $\Lambda_q^\perp(\mathbf{A})$.

Gadget Matrix. Let $n, q \in \mathbb{Z}$, $\mathbf{g} = (1, 2, 4, \cdots, 2^{\lceil \log q \rceil - 1}) \in \mathbb{Z}_q^{\lceil \log q \rceil}$ and $m = n\lceil \log q \rceil$. The gadget matrix is defined as $\mathbf{G} = \mathbf{g} \otimes \mathbf{I}_n \in \mathbb{Z}_q^{n \times m}$ denotes a tensor product of the vector $\mathbf{g}$ and the matrix $\mathbf{I}_n$, such that the lattice $\Lambda_q^\perp(\mathbf{G})$ has a public known basis $\mathbf{T_G}$ with $\|\widetilde{\mathbf{T_G}}\| \leq \sqrt{5}$.

Discrete Gaussian [1]. Let L be a subset of $\mathbb{Z}^m$. For any vector $\mathbf{c} \in \mathbb{R}^m$ and any positive parameter $\sigma \in \mathbb{R}$, define:

$$\rho_{\sigma,\mathbf{c}}(\mathbf{x}) = \exp(-\pi \frac{\|\mathbf{x} - \mathbf{c}\|^2}{\sigma^2}) \text{ and } \rho_{\sigma,\mathbf{c}}(L) = \sum_{\mathbf{x} \in L} \rho_{\sigma,\mathbf{c}}(\mathbf{x}).$$

A discrete Gaussian distribution on L with center $\mathbf{c}$ and parameter σ is

$$\forall \mathbf{y} \in L, \mathcal{D}_{L,\sigma,\mathbf{c}}(\mathbf{y}) = \frac{\rho_{\sigma,\mathbf{c}}(\mathbf{y})}{\rho_{\sigma,\mathbf{c}}(L)}.$$

The distribution $\mathcal{D}_{L,\sigma}$ ($\mathbf{c} = \mathbf{0}$ when ommitted) will most often be defined over lattice $L = \Lambda_q^{\perp}(\mathbf{A})$ for a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ or over a coset $L = \Lambda_q^{\perp}(\mathbf{A}) + \mathbf{t}$ where $\mathbf{t} \in \mathbb{Z}^m$.

Definition 2.4 (Tailcut [1, 20]). Let $q \geq 2$, $m > n$ and $\mathbf{A}$ be a matrix in $\mathbb{Z}_q^{n\times m}$. Let $\mathbf{T_A}$ be a basis for $\Lambda_q^{\perp}(\mathbf{A})$ and $\tau \geq \|\widetilde{\mathbf{T_A}}\|\omega(\sqrt{\log m})$. For $\mathbf{u} \in \mathbb{Z}_q^n$, we have

$$\Pr[\mathbf{x} \xleftarrow{\$} \mathcal{D}_{\Lambda_q^{\mathbf{u}}(\mathbf{A}),\tau} : \|\mathbf{x}\| > \tau\sqrt{m}] \leqslant \mathsf{negl}(\lambda).$$

Bounded Distributions. The following properties below can help us set the parameters appropriately.

Definition 2.5([7,25]). A distribution χ supported over $\mathbb{Z}$ is (B, ϵ)-*bounded*, if we have $\Pr_{x \xleftarrow{\$} \chi}[|x| > B] < \epsilon$.

Definition 2.6 ([7,25]). A distribution $\tilde{\chi}$ supported over $\mathbb{Z}$ is (B, ϵ)-*swallowing* if for all $y \in [-B, B] \cap \mathbb{Z}$, we have that $\tilde{\chi}$ and $y + \tilde{\chi}$ are within ϵ statistical distance.

2.3 Lattice Algorithms, Lattice Evaluation and LWE

Lattice Algorithms. Throughout the paper we will use the following lattice algorithms:

Lemma 2.1 (TrapGen [19]). Let $n, m, q > 0$ be integers with $m \geq O(n \log q)$. A PPT algorithm $\mathsf{TrapGen}(1^n, m, q)$ that outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and a full-rank matrix $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$, where $\mathbf{T_A}$ is a basis for $\Lambda_q^{\perp}(\mathbf{A})$, the distribution of $\mathbf{A}$ is $2^{-\Omega(n)}$-close to uniform and $\|\widetilde{\mathbf{T_A}}\| = O(\sqrt{n \log q})$.

Lemma 2.2 (SamplePre [14]) Let $q \geq 2$, $m > n$. A PPT algorithm $\mathsf{SamplePre}(\mathbf{A}, \mathbf{T_A}, \mathbf{u}, \tau)$ that, given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, a basis $\mathbf{T_A}$ for $\Lambda_q^{\perp}(\mathbf{A})$, a vector $\mathbf{u} \in \mathbb{Z}_q^n$ and a Gaussian parameter $\tau \geq \|\widetilde{\mathbf{T_A}}\|\omega(\sqrt{\log m})$, outputs a vector $\mathbf{e} \in \mathbb{Z}^m$ sampled from a distribution $2^{-\Omega(n)}$-close to $\mathcal{D}_{\Lambda_q^{\mathbf{u}}(\mathbf{A}),\tau}$.

Lemma 2.3 (SampleLeft [1]). Let $q > 2$, $m > n$. A PPT algorithm $\mathsf{SampleLeft}(\mathbf{A}, \mathbf{T_A}, \mathbf{B}, \mathbf{u}, \tau)$ that, given matrices $\mathbf{A} \in \mathbb{Z}_q^{n\times m}, \mathbf{B} \in \mathbb{Z}_q^{n\times m_1}$, a basis $\mathbf{T_A}$ for $\Lambda_q^{\perp}(\mathbf{A})$, a vector $\mathbf{u} \in \mathbb{Z}_q^n$ and a Gaussian parameter $\tau \geq \|\widetilde{\mathbf{T_A}}\|\omega(\sqrt{\log(m + m_1)})$, outputs a vector $\mathbf{e} \in \mathbb{Z}^{m+m_1}$ sampled from a distribution $2^{-\Omega(n)}$-close to $\mathcal{D}_{\Lambda_q^{\mathbf{u}}(\mathbf{A}\|\mathbf{B}),\tau}$.

Lemma 2.4 (SampleRight [1]). Let $q > 2$, $m > n$. A PPT algorithm $\mathsf{SampleRight}(\mathbf{A}, \mathbf{G}, \mathbf{R}, \mathbf{T_G}, \mathbf{u}, \tau)$ that, given matrices $\mathbf{A} \in \mathbb{Z}_q^{n\times k}, \mathbf{G} \in \mathbb{Z}_q^{n\times m}, \mathbf{R} \in \mathbb{Z}^{k\times m}$, a basis $\mathbf{T_G}$ for $\Lambda_q^{\perp}(\mathbf{G})$, a vector $\mathbf{u} \in \mathbb{Z}_q^n$ and a Gaussian parameter $\tau \geq \|\widetilde{\mathbf{T_G}}\| \cdot s_R \cdot \omega(\sqrt{\log m})$ (where $s_R := \|\mathbf{R}\|$), outputs a vector $\mathbf{e} \in \mathbb{Z}^{m+k}$ sampled from a distribution $2^{-\Omega(n)}$-close to $\mathcal{D}_{\Lambda_q^{\mathbf{u}}(\mathbf{A}\|\mathbf{AR}+\mathbf{G}),\tau}$.

Lemma 2.5 (ExtendRight. [5,8]) Let $n, m, q > 0$ be integers with q prime. A PPT algorithm $\mathsf{ExtendRight}(\mathbf{A}, \mathbf{T_A}, \mathbf{B})$ that, given matrices $\mathbf{A} \in \mathbb{Z}_q^{n\times m_1}, \mathbf{B} \in \mathbb{Z}_q^{n\times m_2}$ and a basis $\mathbf{T_A}$ for $\Lambda_q^{\perp}(\mathbf{A})$, outputs a basis $\mathbf{T}_{(\mathbf{A}\|\mathbf{B})}$ for $\Lambda_q^{\perp}(\mathbf{A}\|\mathbf{B})$ such that $\|\widetilde{\mathbf{T_A}}\| = \|\widetilde{\mathbf{T}_{(\mathbf{A}\|\mathbf{B})}}\|$.

Lemma 2.6 (ExtendLeft [5, 8]) Let $n, m, q > 0$ be integers with q prime. A PPT algorithm $\mathsf{ExtendLeft}(\mathbf{A}, \mathbf{G}, \mathbf{T_G}, \mathbf{R})$ that, given matrices $\mathbf{A} \in \mathbb{Z}_q^{n\times k}, \mathbf{G} \in \mathbb{Z}_q^{n\times m}, \mathbf{R} \in \mathbb{Z}^{k\times m}$, a basis $\mathbf{T_G}$ for $\Lambda_q^{\perp}(\mathbf{G})$, outputs a basis $\mathbf{T_H}$ for $\Lambda_q^{\perp}(\mathbf{H})$ where $\mathbf{H} = (\mathbf{A}\|\mathbf{AR} + \mathbf{G})$, such that $\|\widetilde{\mathbf{T_H}}\| \leq \|\widetilde{\mathbf{T_G}}\|(1 + \|\mathbf{R}\|)$.

Lemma 2.7 (RandBasis [5, 8]). Let $m, q \geq 2$ be integers with q prime. A PPT algorithm $\mathsf{RandBasis}(\mathbf{A}, \mathbf{T_A}, \tau)$ that, given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, a basis $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$ for $\Lambda_q^{\perp}(\mathbf{A})$ and a Gaussian parameter $\tau = \|\widetilde{\mathbf{T_A}}\|\omega(\sqrt{\log m})$, outputs a basis $\mathbf{T'_A}$ for $\Lambda_q^{\perp}(\mathbf{A})$ sampled from a distribution that is statistically close to $\mathcal{D}^m_{\Lambda_q^{\perp}(\mathbf{A})),\sigma}$. Note that $\|\widetilde{\mathbf{T'_A}}\| < \tau\sqrt{m}$ with all but negligible probability.

We use the abstraction form of lattice evaluation as extracted in [25].

Theorem 2.1 (Lattice Evaluation [25]). Let $n, q, l \in \mathbb{N}$ and $m = n\lceil \log q\rceil$, there exist two deterministic algorithms called **EvalF** and **EvalFX** respectively. For any depth d boolean circuit $f : \{0,1\}^l \to \{0,1\}^k$ and for every $x \in \{0,1\}^l$, for any matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times ml}$, the outputs $\mathbf{H} \leftarrow \mathbf{EvalF}(f, \mathbf{A})$ and $\hat{\mathbf{H}} \leftarrow \mathbf{EvalFX}(f, x, \mathbf{A})$ are both in $\mathbb{Z}^{ml\times mk}$ and it holds that $\|\mathbf{H}\|_\infty$, $\|\hat{\mathbf{H}}\|_\infty \leqslant (2m)^d$ and

$$[\mathbf{A} - x \otimes \mathbf{G}]\hat{\mathbf{H}} = \mathbf{AH} - f(x)\mathbf{G} (\text{mod } q).$$

Moreover, for any pair of circuits $f : \{0,1\}^l \to \{0,1\}^k$, $g : \{0,1\}^k \to \{0,1\}^t$ and for any matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times ml}$, the outputs $\mathbf{H}_f \leftarrow \mathbf{EvalF}(f, \mathbf{A})$ and $\mathbf{H}_g \leftarrow \mathbf{EvalF}(g, \mathbf{AH}_f)$ and $\mathbf{H}_{g\circ f} \leftarrow \mathbf{EvalF}(g \circ f, \mathbf{A})$ satisfy $\mathbf{H}_f\mathbf{H}_g = \mathbf{H}_{g\circ f}$.

Learning with errors. The learning with errors (LWE) problem was introduced by Regev [23]. In this work we will use its decisional version and Hermite normal form (HNF).

Definition 2.7 (Decisional LWE (DLWE) and Its HNF [2,23]). Let λ be the security parameter, $n = n(\lambda)$ and $q = q(\lambda)$ be integers and let $\chi = \chi(\lambda)$ be a probability distribution over $\mathbb{Z}$. The $\mathrm{DLWE}_{n,q,\chi}$ problem states that for all $m = \mathsf{poly}(n)$, $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, $\mathbf{e} \xleftarrow{\$} \chi^m$, and $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^m$, it holds that

$$(\mathbf{A}, \mathbf{A}^T\mathbf{s} + \mathbf{e}) \quad \text{and} \quad (\mathbf{A}, \mathbf{u})$$

are computationally indistinguishable regardless of $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$ or $\mathbf{s} \xleftarrow{\$} \chi^n$.

Corollary 2.1([22,23,25]). For all $\epsilon > 0$ there exist functions $q = q(n) \leq 2^n$, $\chi = \chi(n)$ such that χ is B-bounded for some $B = B(n)$, $q/B \leq 2^{n^\epsilon}$ and such that $\mathrm{DLWE}_{n,q,\chi}$ is at least as hard as the classical hardness of GapSVP_γ and the quantum hardness of SIVP_γ for $\gamma = 2^{\Omega(n^\epsilon)}$.

2.4 Vector Decomposition and Generalized Leftover Hash Lemma

Key Switching. We show some subroutines of the key switching procedure as proposed in [6].

Definition 2.8 (Vector Decomposition). We define the functions mapping vectors to a higher dimension as below:

- $\mathsf{BD}(\mathbf{v})$: A deterministic function that given a vector $\mathbf{v} \in \mathbb{Z}_q^n$, let $\mathbf{v}_i \in \{0,1\}^n$ be such that $\mathbf{v} = \sum_{i=1}^{\lceil \log q \rceil - 1} 2^i \mathbf{v}_i$, outputs a vector $\widetilde{\mathbf{v}} \in \{0,1\}^{n\lceil \log q \rceil}$, where $\widetilde{\mathbf{v}} = (\mathbf{v}_0; \cdots; \mathbf{v}_{\lceil \log q \rceil - 1})$.
- $\mathsf{P2}(\mathbf{x})$: A deterministic function that given a vector $\mathbf{x} \in \mathbb{Z}_q^n$, outputs a vector $\bar{\mathbf{x}} \in \mathbb{Z}_q^{n\lceil \log q \rceil}$, where $\bar{\mathbf{x}} = [\mathbf{x}; 2\mathbf{x}; \cdots; 2^{\lceil \log q \rceil - 1}\mathbf{x}]$.
- For vectors $\mathbf{v}, \mathbf{x} \in \mathbb{Z}_q^n$, it holds that $\langle \mathsf{BD}(\mathbf{v}), \mathsf{P2}(\mathbf{x}) \rangle = \langle \mathbf{v}, \mathbf{x} \rangle \bmod q$.

Randomness Extraction. We introduce a generalization version of the leftover hash lemma from [1].

Definition 2.9 (Generalized Leftover Hash Lemma [1]). Suppose that $m > (n+1)\log q + \omega(\log n)$ and that $q > 2$ is prime. Let $\mathbf{S} \xleftarrow{\$} \{-1,0,1\}^{m \times k}$ where $k = k(n)$. Choose matrices $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ and $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{n \times k}$. Then, for all vectors $\mathbf{e} \in \mathbb{Z}_q^m$, the distribution $(\mathbf{A}, \mathbf{AS}, \mathbf{S}^T\mathbf{e})$ is statistically close to the distribution $(\mathbf{A}, \mathbf{B}, \mathbf{S}^T\mathbf{e})$.

Note that even without revealing some small amount of information of $\mathbf{S}$ (i.e., $\mathbf{S}^T\mathbf{e}$), the distributions $(\mathbf{A}, \mathbf{AS})$ and $(\mathbf{A}, \mathbf{B})$ are statistically close.

3 Definition of Conditional Attribute-Based Proxy Re-encryption

In this section, we give the definitions of conditional attribute-based proxy re-encryption (CAB-PRE). Specifically, we present the syntax of unidirectional, single-hop CAB-PRE for ciphertext policy and its HRA security notions. A CAB-PRE scheme enables a proxy to convert ciphertexts of a policy f into ciphertexts of a policy g under certain constraints. It uses additional conditions to control which ciphertexts can be chosen instead of all.

3.1 Unidirectional, Single-Hop CAB-PRE

Definition 3.1 (Conditional Attribute-Based Proxy Re-Encryption for Ciphertext Policy). Let $\mathcal{F} : \{0,1\}^l \to \{0,1\}$ be a function class. A conditional attribute-based proxy re-encryption scheme CAB-PRE for policies in $\mathcal{F}$ is a tuple of PPT algorithms $\Pi_{\mathsf{CAB-PRE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_1, \mathsf{Enc}_2, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$.

- $\mathsf{Setup}(1^\lambda) \to (pp, msk)$. On input the security parameter 1^λ, the setup algorithm outputs the public parameters pp along with a master secret key msk.
- $\mathsf{KeyGen}(msk, x) \to sk_x$. On input a master secret key msk and an attribute string $x \in \{0,1\}^l$, the key generation algorithm outputs a secret key sk_x.
- $\mathsf{Enc}_1(f, \mu) \to ct_1$. On input a policy $f \in \mathcal{F}$ and a plaintext $\mu \in \{0,1\}$, the first-level encryption algorithm outputs a first-level ciphertext ct_1.
- $\mathsf{Enc}_2(f, \mu, w) \to ct_2$. On input a policy $f \in \mathcal{F}$, a plaintext $\mu \in \{0,1\}$ and a condition w, the second-level encryption algorithm outputs a second-level ciphertext ct_2. Note that in the setting of single-hop, it could distinguish first-level ciphertexts and second-level ciphertexts.

- $\mathsf{Dec}(sk_x, ct \in \{ct_1, ct_2\}) \to \mu/\perp$. On input a secret key sk_x and a ciphertext $ct \in \{ct_1, ct_2\}$, the decryption algorithm outputs a bit $\mu \in \{0,1\}$ if $f(x) = 1$ or the error symbol $\perp$.
- $\mathsf{ReKeyGen}(sk_x, g, w') \to rk^{w'}_{x\to g}$. Given a secret key sk_x, a policy $g \in \mathcal{F}$, and a condition w', this algorithm outputs a re-encryption key $rk^{w'}_{x\to g}$.
- $\mathsf{ReEnc}(rk^{w'}_{x\to g}, ct_2) \to ct_1/\perp$. Given a re-encryption key $rk^{w'}_{x\to g}$ and a second-level ciphertext ct_2, this algorithm outputs a first-level ciphertext ct_1 if $f(x) = 1$ and $w' \models w$, or the error symbol $\perp$.

Definition 3.2 (CAB-PRE: Correctness). A unidirectional, single-hop CAB-PRE is correct if:

- For all $x \in \{0,1\}^l$ and $f \in \mathcal{F}$ for which $f(x) = 1$, and for any condition w and all $\mu \in \{0,1\}$, it holds that

$$\Pr[\mathsf{Dec}(sk_x, ct \in \{ct_1, ct_2\}) \neq \mu] = \mathsf{negl}(\lambda),$$

  where $ct_1 \leftarrow \mathsf{Enc_1}(f, \mu)$ and $ct_2 \leftarrow \mathsf{Enc_2}(f, \mu, w)$.
- For any condition w' and $rk^{w'}_{x\to g} \leftarrow \mathsf{ReKeyGen}(sk_x, g, w')$, it holds that

$$\Pr[\mathsf{Dec}(sk_y, ct_1) \neq \mu] = \mathsf{negl}(\lambda),$$

  if $g(y) = 1$ and $w' \models w$ for $y \in \{0,1\}^l$ and $g \in \mathcal{F}$, where $ct_1 \leftarrow \mathsf{ReEnc}(rk^{w'}_{x\to g}, ct_2)$.

Definition 3.3 (CAB-PRE: Security Game for HRA). The adaptive HRA security game of an unidirectional, single-hop CAB-PRE scheme between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. The game consists of three phases as below.

Phase 1 (Setup): This is the setup phase. The challenger generates (pp, msk) by running $\mathsf{Setup}(1^\lambda)$ algorithm and gives the public parameter pp to $\mathcal{A}$. Then, the challenger initializes a counter $\mathsf{numCt} := 0$, a policy-value store $\mathsf{C} := \emptyset$ and a set $\mathsf{Deriv} := \emptyset$.

Phase 2 (Oracle query): This is the oracle query phase.

- $\mathcal{O}_{\mathsf{KeyGen}}(x)$: For a key query x, the challenger generates $sk_x \leftarrow \mathsf{KeyGen}(msk, x)$ and gives it to $\mathcal{A}$.
- $\mathcal{O}_{\mathsf{Enc}}(f, w, \mu)$: For an encryption query (f, w, μ), the challenger computes

$$ct \leftarrow \begin{cases} \mathsf{Enc_1}(f, \mu), & \text{if } w = \mathsf{Null} \\ \mathsf{Enc_2}(f, w, \mu), & \text{otherwise} \end{cases}$$

  sets $\mathsf{numCt} := \mathsf{numCt} + 1$, adds ct in C with policy (f, w, numCt), and gives (numCt, ct) to $\mathcal{A}$.
- $\mathcal{O}_{\mathsf{ReKey}}(x, (g, y), w')$: For a re-encryption key query $(x, (g, y), w')$ where y denotes an attribute string, there exist two cases:
  1. $g(y) = 0$, the challenger generates $rk^{w'}_{x\to g} \leftarrow \mathsf{ReKeyGen}(sk_x, g, w')$; If the re-encryption key distribution is statistically close to the uniform (i.e., key privacy [3]), the challenger could choose a random value that is identically and independently distributed according to its distribution.

2. $g(y) = 1$, return $\perp$ if x was not queried in $\mathcal{O}_{\mathsf{KeyGen}}$ and $w' \models w$ of that queried in $\mathcal{O}_{\mathsf{Enc}}$ when $w \neq \mathsf{Null}$. Otherwise, the challenger produces $rk_{x \to g}^{w'} \leftarrow \mathsf{ReKeyGen}(sk_x, g, w')$.

After that, $\mathcal{C}$ gives $rk_{x \to g}^{w'}$ to $\mathcal{A}$.

– $\mathcal{O}_{\mathsf{Cha}}(f^*, w^*, (\mu_0^*, \mu_1^*))$: This oracle can only be invoked once. For a challenge query $(f^*, w^*, (\mu_0^*, \mu_1^*))$, it requires $f^*(x) = 0$ (where x has been queried in $\mathcal{O}_{\mathsf{KeyGen}}$ and $\mathcal{O}_{\mathsf{ReKey}}$). The challenger flips a bit $b \in \{0, 1\}$, generates

$$ct^* \leftarrow \begin{cases} \mathsf{Enc}_1(f^*, \mu_b^*), & \text{if } w^* = \mathsf{Null} \\ \mathsf{Enc}_2(f^*, w^*, \mu_b^*), & \text{otherwise} \end{cases}$$

sets $\mathsf{numCt} := \mathsf{numCt} + 1$ and $\mathsf{Deriv} := \mathsf{Deriv} \cup \mathsf{numCt}$. It adds ct^* in C with policy $(f^*, w^*, \mathsf{numCt})$ and gives (numCt, ct^*) to $\mathcal{A}$.

– $\mathcal{O}_{\mathsf{ReEnc}}(x, (g, y), w', (f, w, k))$: For a re-encryption query $(x, (g, y), w', (f, w, k))$ where $k \leq \mathsf{numCt}$, the challenger does the following operations.
  1. If $w' \not\models w$, return $\perp$.
  2. If there is no value in C with policy $(f, w \neq \mathsf{Null}, k)$, return $\perp$.
  3. If $f(x) = 0$, return $\perp$.
  4. If $g(y) = 1 \cap k \in \mathsf{Deriv}$, return $\perp$.
  5. Otherwise, let ct be that value in C. The challenger produces $ct' \leftarrow \mathsf{ReEnc}(rk_{x \to g}^{w'}, ct)$ where $rk_{x \to g}^{w'} \leftarrow \mathsf{ReKeyGen}(sk_x, g, w')$, sets $\mathsf{numCt} := \mathsf{numCt} + 1$, adds ct' in C with policy $(f, \mathsf{Null}, \mathsf{numCt})$. If $k \in \mathsf{Deriv}$, set $\mathsf{Deriv} := \mathsf{Deriv} \cup \mathsf{numCt}$. Finally, it gives (numCt, ct') to $\mathcal{A}$.

Phase 3 (Decision): This is the decision phase. $\mathcal{A}$ outputs a bit b' for b. The adaptive HRA advantage of $\mathcal{A}$ wins the game is defined as

$$\mathsf{Adv}_{\mathcal{A}, \Pi_{\mathsf{CAB-PRE}}}^{\mathsf{HRA}} = |\Pr[b' = b] - 1/2|.$$

Definition 3.4 (CAB-PRE: Adaptive HRA Security). Given a security parameter 1^λ, we say the scheme $\Pi_{\mathsf{CAB-PRE}}$ for ciphertext policy is unidirectional, single-hop adaptively HRA-secure if for all PPT adversaries $\mathcal{A}$, there has a negligible function $\mathsf{negl}(\lambda)$ such that

$$\mathsf{Adv}_{\mathcal{A}, \Pi_{\mathsf{CAB-PRE}}}^{\mathsf{HRA}} \leq \mathsf{negl}(\lambda).$$

4 Single-Hop Construction Based on Lattices

In this section, we present a CAB-PRE scheme for inner-product predicate based on DLWE assumption. Our construction is derived from the CP-ABE scheme [25], and also suppose that the underlying cPRF keeps single-key adaptive security.

Let $\mathsf{P} = (\mathsf{P.Setup}, \mathsf{P.Eval}, \mathsf{P.Constrain}, \mathsf{P.ConstrainEval})$ be a conforming cPRF for a class family $\mathcal{F}$ of $t-$CNF predicates with input length l and output length k. Assume that the master secret key length of P is λ. For all $f \in \mathcal{F}$ let l_f denote the size of constrained key which can be computed efficiently giving function f

and descriptions of $\mathsf{P.Constrain}$ algorithm. Let C_β be an inner-product predicate with dimension l that on input $\alpha \in \mathbb{Z}_q^l$, then $C_\beta(\alpha) = 1$ iff $\langle \beta, \alpha \rangle = 0$. Note that $U_{\sigma\to x}, U_{\sigma\to f}, U_{f\to x}$ are circuits defined as Definition 2.2. Define an adaptive-secure CAB-PRE for inner-product as follows.

– $\mathsf{Setup}(1^\lambda)$: Run $(P.pp, P.msk) \leftarrow \mathsf{P.Setup}(1^\lambda)$, set $\sigma = P.msk$. $n, q, m', \tau, \chi, \tilde{\chi}$ are parameters and let $m = n\lceil \log q \rceil$. Invoke $(\mathbf{B}, \mathbf{T_B}) \leftarrow \mathsf{TrapGen}(1^n, m', q)$. Sample matrices $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m\lambda}$ and $\mathbf{V}, \mathbf{D} \xleftarrow{\$} \mathbb{Z}_q^{n\times l\lceil \log q\rceil}$. Output $pp = (\mathbf{B}, \mathbf{A}, \mathbf{V}, \mathbf{D}, P.pp)$ and $msk = (\sigma, \mathbf{T_B})$.
– $\mathsf{KeyGen}(msk, x)$: Compute $\mathbf{H}_{\sigma\to x} \leftarrow \mathsf{EvalF}(U_{\sigma\to x}, \mathbf{A})$, let $\mathbf{A}_x = \mathbf{A}\mathbf{H}_{\sigma\to x}$. Evaluate $r \leftarrow \mathsf{P.Eval}(\sigma, x)$ and $\mathbf{H}_r \leftarrow \mathsf{EvalF}(I_r, \mathbf{A}_x)$ where $I_r : \{0,1\}^k \to \{0,1\}$ is a function that on input r' returns 0 iff $r \neq r'$, set $\mathbf{A}_{x,r} = \mathbf{A}_x\mathbf{H}_r$. Compute $\mathbf{T}_{\mathbf{B}\|\mathbf{A}_{x,r}} \leftarrow \mathsf{RandBasis}(\mathbf{B}\|\mathbf{A}_{x,r}, \mathsf{ExtendRight}(\mathbf{B}, \mathbf{T_B}, \mathbf{A}_{x,r}), \tau)$ and sample $\mathbf{k} \leftarrow \mathsf{SamplePre}(\mathbf{B}\|\mathbf{A}_{x,r}, \mathbf{T}_{\mathbf{B}\|\mathbf{A}_{x,r}}, \mathbf{V}, \tau)$ such that $(\mathbf{B}||\mathbf{A}_{x,r})\mathbf{k} = \mathbf{V}$. Output $sk_x = (r, \mathbf{T}_{\mathbf{B}\|\mathbf{A}_{x,r}}, \mathbf{k})$.
– $\mathsf{Enc}_1(f, \mu)$: Compute $s_f \leftarrow \mathsf{P.KeySim}(P.pp, f)$. Choose randomly $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$,$\mathbf{e}_0 \xleftarrow{\$} \chi^{m'}$,$\mathbf{e}_1 \xleftarrow{\$} \tilde{\chi}^{ml_f}$,$\mathbf{e}_2 \xleftarrow{\$} \chi^{l\lceil \log q\rceil}$, compute

$$\mathbf{u}_0 = \mathbf{s}^T\mathbf{B} + \mathbf{e}_0^T,$$
$$\mathbf{u}_1 = \mathbf{s}^T(\mathbf{A}_f - s_f \otimes \mathbf{G}) + \mathbf{e}_1^T,$$
$$\mathbf{u}_2 = \mathbf{s}^T\mathbf{V} + \mathbf{e}_2^T + \mu(\mathbf{0}\|\mathbf{g}^T),$$

the zero vector has dimension $(l-1)\lceil \log q\rceil$, where $\mathbf{A}_f = \mathbf{A}\mathbf{H}_{\sigma\to f}$, $\mathbf{H}_{\sigma\to f} \leftarrow \mathsf{EvalF}(U_{\sigma\to f}, \mathbf{A})$. Output $ct_1 = (s_f, \mathbf{u}_0, \mathbf{u}_1, \mathbf{u}_2)$.
– $\mathsf{Enc}_2(f, \mu, \alpha)$: Choose vectors $\mathbf{W} \xleftarrow{\$} \mathbb{Z}_q^n, \alpha \in \mathbb{Z}_q^l$ and an error $\mathbf{e}_3 \xleftarrow{\$} \chi^{l\lceil \log q\rceil}$. Identical to Enc_1 except that adding

$$\mathbf{u}_3 = \mathbf{s}^T(\mathbf{D} + \mathbf{W} \otimes \mathsf{P2}(\alpha)^T) + \mathbf{e}_3^T.$$

Output $ct_2 = (s_f, \mathbf{u}_0, \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)$.
– $\mathsf{Dec}(sk_x, ct)$: Parse $sk_x = (r, \mathbf{T}_{\mathbf{B}\|\mathbf{A}_{x,r}}, \mathbf{k})$ and $ct \in \{ct_1, ct_2\}$. Compute $r' \leftarrow U_{f\to x}(s_f)$, abort if $r' = r$. Otherwise, capture $\mathbf{A}_x$ and $\mathbf{A}_f$ as the same as KeyGen and Enc_1, respectively. Evaluate

$$\hat{\mathbf{H}}_{r,r'} \leftarrow \mathsf{EvalFX}(I_r, r', \mathbf{A}_x),$$
$$\hat{\mathbf{H}}_{s_f\to r'} \leftarrow \mathsf{EvalFX}(U_{f\to x}, s_f, \mathbf{A}_f).$$

Lastly, compute

$$\mu = \left\lceil \mathbf{u}_2 - (\mathbf{u}_0\|\mathbf{u}_1\hat{\mathbf{H}}_{s_f\to r'}\hat{\mathbf{H}}_{r,r'})\mathbf{k} \right\rfloor_2,$$

in which $\lceil\cdot\rfloor_2 : \mathbb{Z}_q \to \{0,1\}$ indicates its penultimate is closer modulo q to 0 or to $2^{\lceil \log q\rceil - 2}$. Noting it has the same decryption algorithm regardless of which level ciphertexts (first or second). However, there is a coefficient factor associated with the predicate vector β when decrypting a transformed ciphertext ct'. In this case, we will recover the message μ by tailoring its noise bound.

– $\mathsf{ReKeyGen}(sk_x, g, \beta)$: Parse $sk_x = (r, \mathbf{T}_{\mathbf{B}\|\mathbf{A}_{x,r}}, \mathbf{k})$. Choose a function $g \in \mathcal{F}$ that has the property of gradual evaluation (Definition 2.2) of that the size of constrained key is l_g, and a predicate vector $\beta \in \mathbb{Z}_q^l$. Sample $s_g \leftarrow \mathsf{P.KeySim}(P.pp, g)$, let $\mathbf{H} = \mathbf{B}\|(\mathbf{A}_g - s_g \otimes \mathbf{G})$ where $\mathbf{A}_g = \mathbf{A}\mathbf{H}_{\sigma\to g}$, $\mathbf{H}_{\sigma\to g} \leftarrow \mathsf{EvalF}(U_{\sigma\to g}, \mathbf{A})$. Compute $\mathbf{A}_{x,r}$ as in KeyGen, let $\mathbf{V}' = (\mathbf{V}+\mathbf{D})\cdot \mathsf{BD}(\beta)$ and sample $\mathbf{d} \leftarrow \mathsf{SamplePre}(\mathbf{B}\|\mathbf{A}_{x,r}, \mathbf{T}_{\mathbf{B}\|\mathbf{A}_{x,r}}, \mathbf{V}', \tau)$ such that $(\mathbf{B}\|\mathbf{A}_{x,r})\mathbf{d} = \mathbf{V}'$. Select randomly $\mathbf{r}_1 \in \chi^{(m'+m)\lceil \log q\rceil \times n}$, $\mathbf{r}_2 \in \chi^{(m'+m)\lceil \log q\rceil \times (m'+ml_g)}$, $\mathbf{r}_3 \in \chi^{(m'+m)\lceil \log q\rceil \times l\lceil \log q\rceil}$, $\gamma \in \{0,1\}^{l\lceil \log q\rceil}$, set

$$\mathbf{Z} = \begin{pmatrix} \mathbf{r}_1\mathbf{H} + \mathbf{r}_2 & \mathbf{r}_1\mathbf{V} + \mathbf{r}_3 - \mathsf{P2}(\mathbf{d}) \otimes \gamma \\ \mathbf{0}_{l\lceil \log q\rceil \times (m'+ml_g)} & \mathsf{BD}(\beta) \otimes \gamma \end{pmatrix}.$$

Output $rk_{x\to g}^{\beta} = (s_g, r, \beta, \gamma, \mathbf{Z})$.

– $\mathsf{ReEnc}(rk_{x\to g}^{\beta}, ct_2)$: Parse $rk_{x\to g}^{\beta} = (s_g, r, \beta, \gamma, \mathbf{Z})$ and $ct_2 = (s_f, \mathbf{u}_0, \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)$. Compute $r' \leftarrow U_{f\to x}(s_f)$, abort if $r' = r$. Otherwise, compute $\mathbf{A}_x$, $\mathbf{A}_f$, $\hat{\mathbf{H}}_{r,r'}$ and $\hat{\mathbf{H}}_{s_f\to r'}$ as in KeyGen, Enc and Dec. Finally, let $\mathbf{u}_1' = \mathbf{u}_1\hat{\mathbf{H}}_{s_f\to r'}\hat{\mathbf{H}}_{r,r'}$ and $\mathbf{u}_2' = \mathbf{u}_2 + \mathbf{u}_3$, evaluate

$$ct_{f\to g} = (\mathsf{BD}(\mathbf{u}_0\|\mathbf{u}_1')\|\mathbf{u}_2') \cdot \mathbf{Z}.$$

Output $ct' = (s_g, \beta, \gamma, ct_{f\to g})$.

4.1 Correctness and Choice of Parameters

Theorem 4.1. The CAB-PRE scheme is correct with respect to f and C_β under proper parameters as below.

Proof. To succinctly, we consider correct decryption into two parts. One is the plain ciphertexts of first-level or second-level since they have the same operations during decrypting. The other is re-encrypted ciphertext.

First, for the security parameter λ, a function $f \in \mathcal{F}$, and an attribute string $x \in \{0,1\}^l$ such that $f(x) = 1 \wedge I_r(r') = 0$, if $(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda)$, $sk_x \leftarrow \mathsf{KeyGen}(msk, x)$, $ct_1 \leftarrow \mathsf{Enc}(f, \mu)$ and $ct_2 \leftarrow \mathsf{Enc}(f, \mu, \alpha)$, then we have $\mu \leftarrow \mathsf{Dec}(sk_x, ct \in \{ct_1, ct_2\})$. Recall [25] shows that $r \neq r'$ with all but negligible probability via a reduction to the pseudorandomness game of P, because $r \leftarrow \mathsf{P.Eval}(P.msk, x)$, and $r' \leftarrow U_{f\to x}(s_f)$ is computed as $\mathsf{P.ConstrainEval}(s_f, x)$ where $s_f \leftarrow \mathsf{P.KeySim}(P.pp, f)$. Hence given the function I_r, there exist $I_r(r') = 0$ with a non-negligible advantage.

We compute $\mathbf{A}_x = \mathbf{A}\mathbf{H}_{\sigma\to x} = \mathbf{A}\mathbf{H}_{\sigma\to f}\mathbf{H}_{f\to x} = \mathbf{A}_f\mathbf{H}_{f\to x}$ where $\mathbf{H}_{f\to x} \leftarrow \mathsf{EvalF}(U_{f\to x}, \mathbf{A})$ and $\mathbf{A}_f = \mathbf{A}\mathbf{H}_{\sigma\to f}$, $\mathbf{H}_{\sigma\to f} \leftarrow \mathsf{EvalF}(U_{\sigma\to f}, \mathbf{A})$. Since the property of gradual evaluation of P when $f(x) = 1$ is that the effective sub-circuit of $U_{f\to x} \circ U_{\sigma\to f}$ is equivalent to the circuit $U_{\sigma\to x}$, then we know $\mathbf{H}_{\sigma\to x} = \mathbf{H}_{\sigma\to f}\mathbf{H}_{f\to x}$.

Based on Theorem 2.1, we capture $(\mathbf{H}_r, \hat{\mathbf{H}}_{r,r'})$ and $(\mathbf{H}_{f\to x}, \hat{\mathbf{H}}_{s_f\to r'})$. Precisely,

$$\mathbf{H}_r \leftarrow \mathsf{EvalF}(I_r, \mathbf{A}_x),$$

$$\hat{\mathbf{H}}_{r,r'} \leftarrow \mathsf{EvalFX}(I_r, r', \mathbf{A}_x),$$

$$\hat{\mathbf{H}}_{s_f \to r'} \leftarrow \mathsf{EvalFX}(U_{f\to x}, s_f, \mathbf{A}_f).$$

Then, computing

$$\begin{aligned}(\mathbf{A}_f - s_f \otimes \mathbf{G})\hat{\mathbf{H}}_{s_f\to r'}\hat{\mathbf{H}}_{r,r'} &= [\mathbf{A}_f \mathbf{H}_{f\to x} - U_{f\to x}(s_f)\otimes \mathbf{G}]\hat{\mathbf{H}}_{r,r'} \\ &= [\mathbf{A}_x - r' \otimes \mathbf{G}]\hat{\mathbf{H}}_{r,r'} \\ &= \mathbf{A}_x \mathbf{H}_r - I_r(r')\otimes \mathbf{G} \\ &= \mathbf{A}_{x,r}.\end{aligned}$$

Thus,

$$\begin{aligned}\mathbf{u}_2 - (\mathbf{u}_0 \| \mathbf{u}_1 \hat{\mathbf{H}}_{s_f\to r'}\hat{\mathbf{H}}_{r,r'})\mathbf{k} &= \mathbf{u}_2 - (\mathbf{u}_0 \| \mathbf{s}^T \mathbf{A}_{x,r} + \mathbf{e}_1^T \hat{\mathbf{H}}_{s_f\to r'}\hat{\mathbf{H}}_{r,r'})\mathbf{k} \\ &= \mathbf{u}_2 - \mathbf{s}^T(\mathbf{B}\|\mathbf{A}_{x,r})\mathbf{k} - (\mathbf{e}_0^T \| \mathbf{e}_1^T \hat{\mathbf{H}}_{s_f\to r'}\hat{\mathbf{H}}_{r,r'})\mathbf{k} \\ &= \mu(\mathbf{0}\|\mathbf{g}^T) + \mathbf{e}_2^T - (\mathbf{e}_0^T \| \mathbf{e}_1')\mathbf{k},\end{aligned}$$

where $\mathbf{e}_1' = \mathbf{e}_1^T \hat{\mathbf{H}}_{s_f\to r'}\hat{\mathbf{H}}_{r,r'}$.

Let d_{Con} and d_{ConEval} denote the depth of $U_{\sigma\to f}$ and $U_{f\to x}$, respectively, which are bounded by the depth $d = \mathsf{poly}(\lambda)$ of $U_{\sigma\to x}$ since P is the gradual depth of $U_{\sigma\to f}$ and $U_{f\to x}$. Note that

$$\begin{aligned}\|\mathbf{e}_1'\|_\infty &\le m^2 l_f k \|\mathbf{e}_1^T\|_\infty \|\hat{\mathbf{H}}_{s_f\to r'}\|_\infty \|\hat{\mathbf{H}}_{r,r'}\|_\infty \\ &\le m^2 l_f k \tilde{B}(2m)^{d_{\mathsf{ConEval}}+1}\end{aligned}$$

and

$$\|\mathbf{k}\|_\infty \le \tau\sqrt{m'+m},$$

due to $\mathbf{e}_1 \in \tilde{\chi}^{ml_f}, \hat{\mathbf{H}}_{s_f\to r'} \in \mathbb{Z}^{ml_f \times mk}, \hat{\mathbf{H}}_{r,r'} \in \mathbb{Z}^{mk\times m}$ and the tailcut inequation (Definition 2.4) of discrete Gaussian.

Therefore, if $l \in O(n)$, $m', l_f, k \in O(n\lceil \log q\rceil)$, $\tilde{B} \in O(B, n)$ and $\tau \in O(\lambda k (2m)^{d+3})$, then

$$\begin{aligned}\|\mathbf{e}_2^T - (\mathbf{e}_0^T \| \mathbf{e}_1')\mathbf{k}\|_\infty &\le \|\mathbf{e}_2^T\|_\infty + (m'\|\mathbf{e}_0^T\|_\infty + m\|\mathbf{e}_1'\|_\infty)\|\mathbf{k}\|_\infty \\ &\le l\lceil \log q\rceil B + (m'B + m^3 l_f k \tilde{B}(2m)^{d_{\mathsf{ConEval}}+1})\tau\sqrt{m'+m} \\ &\le B\cdot \mathsf{poly}(n\lceil\log q\rceil)\cdot(2m)^{d_{\mathsf{ConEval}}+d+4}.\end{aligned}$$

To capture correct decryption, the magnitude of penultimate coordinate should be less than $q/8$, i.e., $\|\mathbf{e}_2^T - (\mathbf{e}_0^T\|\mathbf{e}_1')\mathbf{k}\|_\infty < q/8$. Now, by choosing the parameters

- q, χ, B as Corollary 2.1 and note that $q \le 2^n$ and $q/B \ge 2^{n^\epsilon}$,
- $n \ge \lambda$ such that $(2n^2)^{(2d+4)} \le 2^{n^\epsilon}$ where $\epsilon \in (0,1)$ and $n \le d^{O(1/\epsilon)}$
- $m' = (n+1)\lceil \log q\rceil + 2\lambda$, $B' = m'm\lambda B(2m)^{d_{\mathsf{Con}}}$, $E' \le 2^{n^\epsilon}$

Let

$$E = B \cdot \mathsf{poly}(n\lceil \log q\rceil) \cdot (2m)^{d_{\mathsf{ConEval}}+d+4}$$

and

$$E' = 8E/B = 8 \cdot \mathsf{poly}(n\lceil \log q\rceil) \cdot (2m)^{d_{\mathsf{ConEval}}+d+4}.$$

Since $E' < q/B$, then $E = BE'/8 < q/8$. Therefore,

$$\|\mathbf{e}_2^T - (\mathbf{e}_0^T\|\mathbf{e}_1')\mathbf{k}\|_\infty \leq B \cdot \mathsf{poly}(n\lceil \log q\rceil) \cdot (2m)^{d_{\mathsf{ConEval}}+d+4} < q/8$$

is overwhelming, and the decryption of plain ciphertext is correct. That completes the correctness of part 1.

Now it remains to show how to guarantee the correctness of re-encrypted ciphertext. Given a re-encryption key and a second-level ciphertext associated with function f, which can be efficiently re-encrypted to the first-level ciphertext associated with function g, under certain constraints with inner-product predicates between two vectors α and β over $\mathbb{Z}_q^l$. Specifically, parse the re-encryption key $rk_{x\to g}^{\beta} = (s_g, r, \beta, \gamma, \mathbf{Z})$ and the second-level ciphertext $ct_2 = (s_f, \mathbf{u}_0, \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)$, the re-encryption process in the following way:

Assume that $r' \neq r$ where $r' \leftarrow U_{f\to x}(s_f)$, let

$$\mathbf{u}_1' = \mathbf{u}_1\hat{\mathbf{H}}_{s_f\to r'}\hat{\mathbf{H}}_{r,r'} = \mathbf{s}^T\mathbf{A}_{x,r} + \mathbf{e}_1^T\hat{\mathbf{H}}_{s_f\to r'}\hat{\mathbf{H}}_{r,r'}$$

and

$$\mathbf{u}_2' = \mathbf{u}_2 + \mathbf{u}_3 = \mathbf{s}^T(\mathbf{V} + \mathbf{D} + \mathbf{W} \otimes \mathsf{P}_2(\alpha)^T) + \bar{\mathbf{e}}^T + \mu(\mathbf{0}\|\mathbf{g}^T).$$

where $\bar{\mathbf{e}} = \mathbf{e}_2 + \mathbf{e}_3$. If $l = (l_g + 1)n + \lambda + 1$ and $C_\beta(\alpha) = 1$, we have

$$\begin{aligned} ct_{f\to g} &= (\mathsf{BD}(\mathbf{u}_0\|\mathbf{u}_1')\|\mathbf{u}_2') \cdot \mathbf{Z} \\ &= (\mathbf{c}_0\|\mathbf{c}_1 \quad \mathbf{c}_2). \end{aligned}$$

Let $\bar{\mathbf{s}} = [\mathsf{BD}(\mathbf{u}_0\|\mathbf{u}_1')\mathbf{r}_1]^T$, then

$$\begin{aligned} \mathbf{c}_0\|\mathbf{c}_1 &= \bar{\mathbf{s}}^T(\mathbf{B}\|(\mathbf{A}_g - s_g \otimes \mathbf{G})) + \mathsf{BD}(\mathbf{u}_0\|\mathbf{u}_1')\mathbf{r}_2 \\ &= \bar{\mathbf{s}}^T(\mathbf{B}\|(\mathbf{A}_g - s_g \otimes \mathbf{G})) + (\bar{\mathbf{e}}_0^T\|\bar{\mathbf{e}}_1^T) \end{aligned}$$

and

$$\mathbf{c}_2 = \bar{\mathbf{s}}^T\mathbf{V} + \bar{\mathbf{e}}_2 + \mu(\mathbf{0}\|\mathbf{g}^T) \cdot \mathsf{BD}(\beta) \otimes \gamma,$$

where $\bar{\mathbf{e}}_2 = \mathsf{BD}(\mathbf{u}_0\|\mathbf{u}_1')\mathbf{r}_3 - [(\mathbf{e}_0^T\|\mathbf{e}_1')\mathbf{d} - \bar{\mathbf{e}}^T \cdot \mathsf{BD}(\beta)] \otimes \gamma$ and $\|\bar{\mathbf{e}}_2\|_\infty \leq B \cdot \mathsf{poly}(n\lceil \log q\rceil) \cdot (2m)^{d_{\mathsf{ConEval}}+d+4}$.

Since $\beta = (\beta_1, ..., \beta_l) \in \mathbb{Z}_q^l$, it holds that $(\mathbf{0}\|\mathbf{g}^T) \cdot \mathsf{BD}(\beta) \neq 0$ (i.e., let $\beta_l = q-1$) with a significant probability. Thus, let

$$(\mathbf{0}\|\mathbf{g}^T) \cdot \mathsf{BD}(\beta) \otimes \gamma = (\sum_{i=0}^{\lceil \log q\rceil - 1} \beta_{l,i} \cdot 2^i) \otimes \gamma,$$

where $\beta_{l,i} \in \{0,1\}$ denotes the binary decomposition of β_l. Without loss of generality, let $sk_y = (\bar{r}, \mathbf{T}_{\mathbf{B}\|\mathbf{A}_{y,\bar{r}}}, \bar{\mathbf{k}})$ be the secret key for attribute string y, which satisfies $g(y) = 1$. Suppose that the penultimate bit of γ is 1, to capture correct decryption, compute

$$\mu = \left\lceil \mathbf{c}_2 - (\mathbf{c}_0 \| \mathbf{c}_1 \hat{\mathbf{H}}_{s_g \to \bar{r}'} \hat{\mathbf{H}}_{\bar{r},\bar{r}'}) \bar{\mathbf{k}} \right\rfloor_2 ,$$

here $\lceil \cdot \rfloor_2$ denotes whether its penultimate is closer modulo q to 0 or to $\sum_{i=0}^{\lceil \log q \rceil - 1} \beta_{l,i} \cdot 2^i$.

Based on our parameter settings and analysis, the norm of error term is bounded by

$$\|\bar{\mathbf{e}}_2^T - (\bar{\mathbf{e}}_0^T \| \bar{\mathbf{e}}_1') \bar{\mathbf{k}}\|_\infty \leq B \cdot \mathsf{poly}(n \lceil \log q \rceil) \cdot (2m)^{d_{\mathsf{ConEval}} + d + 4} < \sum_{i=0}^{\lceil \log q \rceil - 1} \beta_{l,i} \cdot 2^{i+1}$$

in which $\bar{\mathbf{e}}_1' = \bar{\mathbf{e}}_1^T \hat{\mathbf{H}}_{s_g \to \bar{r}'} \hat{\mathbf{H}}_{\bar{r},\bar{r}'}$ with an overwhelming probability, and the decryption of transformed ciphertext is correct. That completes the correctness of part 2.

4.2 Security Proof

For now, we will construct two efficient randomized algorithms $\mathsf{Semi-KeyGen}_1$ and $\mathsf{Semi-KeyGen}_2$ that are the heart of the HRA security proof.

- $\mathsf{Semi-KeyGen}_1(msk, x) \to \hat{sk}_x$: Here $msk = (\sigma, \mathbf{T_B}), x \in \{0,1\}^l$, compute $\mathbf{A}_{x,r}$ as in KeyGen algorithm. Capture $\hat{sk}_x$ by calling $\mathsf{SampleLeft}(\mathbf{B}, \mathbf{A}_{x,r}, \mathbf{T_B}, \mathbf{V}', \tau)$.
- $\mathsf{Semi-KeyGen}_2(msk, x) \to \hat{sk}_x$: Here $msk = \sigma, x \in \{0,1\}^l$, compute $\mathbf{A}_{x,r}$ as in KeyGen algorithm. Capture $\hat{sk}_x$ by calling $\mathsf{SampleRight}(\mathbf{B}, \mathbf{G}, \mathbf{R}, \mathbf{T_G}, \mathbf{V}', \tau)$ if $\mathbf{A}_{x,r} = \mathbf{BR} + \mathbf{G}$, where $\mathbf{R}$ and $\mathbf{V}'$ are fixed matrices.

Note that by Lemma 2.3 and Lemma 2.4, the output of $\mathsf{SampleLeft}$ and $\mathsf{SampleRight}$ algorithms are distributed statistically close to $\mathcal{D}_{\Lambda_q^{\mathbf{V}'}(\mathbf{B}\|\mathbf{A}_{x,r}),\tau}$. Thus, sampling $\hat{sk}_x$ from $\mathsf{Semi-KeyGen}_1$ or $\mathsf{Semi-KeyGen}_2$ are statistically indistinguishable.

Theorem 4.2. For a class family $\mathcal{F}$, if P is a conforming cPRF, then CAB-PRE above is a unidirectional, single-hop, adaptively HRA-secure scheme under the hardness of $\text{DLWE}_{n,q,\chi}$ problem.

Proof. Let $\mathcal{A}$ be a PPT adversary that attacks the adaptive HRA security of CAB-PRE. We show that the proof proceeds in a sequence of games. In each game, we define S_i to be the event that $\mathcal{A}$ wins in $\mathbf{Game}_i$.

$\mathbf{Game}_0$: This is the original security game from Definition 3.3. Note that most operations are identical to the real scheme, for the adversary's queries. However, we pay attention to responses of the re-encryption key and re-encryption queries since there are some rigorous restrictions and subtle transformations, which may be different but equivalent to the real scheme. These queries are handled as follows:

- When $\mathcal{A}$ makes a re-encryption key query on $(x, (g, y), \beta)$ such that $g(y) = 0$. The challenger chooses two matrices $\mathbf{Z}_1 \xleftarrow{\$} \chi^{(m'+m)\lceil \log q \rceil \times (m'+ml_g)}$ and $\mathbf{Z}_2 \xleftarrow{\$} \chi^{(m'+m)\lceil \log q \rceil \times l \lceil \log q \rceil}$, simulates the third entries of the re-encryption key as
$$\mathbf{Z} = \begin{pmatrix} \mathbf{Z}_1 & \mathbf{Z}_2 \\ \mathbf{0}_{l\lceil \log q \rceil \times (m'+ml_g)} & \mathsf{BD}(\beta) \otimes \gamma \end{pmatrix}.$$
It returns $rk^{\beta}_{x \to g}$ to $\mathcal{A}$.
- When $\mathcal{A}$ makes a re-encryption query on $(x, (g, y), \beta, (f, \alpha, k)$ such that $g(y) = 1$ and $k \notin \mathsf{Deriv}$. The challenger first invokes $\mathsf{Semi-KeyGen}_1(msk, x)$ algorithm (since there is $msk = (\sigma, \mathbf{T_B})$) to obtain $\hat{sk}_x (= \mathbf{d})$ without generating secret key sk_x. It then computes $ct' \leftarrow \mathsf{ReEnc}(rk^{\beta}_{x \to g}, ct_2)$ where $rk^{\beta}_{x \to g} \leftarrow \mathsf{ReKeyGen}(\hat{sk}_x, g, \beta)$ and gives ct' to $\mathcal{A}$.

By Definition 3.3 and Definition 3.4, we have $|\Pr[S_0] - 1/2| \leq \mathsf{negl}(\lambda)$.

Remark 1. Recall in the real scheme, the challenger creates
$$\mathbf{Z} = \begin{pmatrix} \mathbf{r}_1\mathbf{H} + \mathbf{r}_2 & \mathbf{r}_1\mathbf{V} + \mathbf{r}_3 - \mathsf{P2}(\mathbf{d}) \otimes \gamma \\ \mathbf{0}_{l\lceil \log q \rceil \times (m'+ml_g)} & \mathsf{BD}(\beta) \otimes \gamma \end{pmatrix},$$
where $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$ are sampled from discrete Gaussian distributions. Observed that it is indistinguishable with uniform distribution (i.e., $\mathbf{Z}_1$ and $\mathbf{Z}_2$) based on the hardness of HNF-LWE problem (Definition 2.7, more proof details please refer to [16]).

Game$_1$: In this game, we change $r \xleftarrow{\$} \{0,1\}^k$ instead of $r \leftarrow \mathsf{P.Eval}(\sigma, x)$ when generating sk_x, if the challenge ciphertext or its derivatives (i.e., $k \in \mathsf{Deriv}$) are queried in $\mathcal{O}_{\mathsf{ReEnc}}$. Based on the pseudorandomness game of P, we have $|\Pr[S_0] - \Pr[S_1]| \leq \mathsf{negl}(\lambda)$.

Game$_2$: In this game, we change the way s_{f^*} for a target function f^* is created. Concretely, instead of computing $s_{f^*} \leftarrow \mathsf{P.KeySim}(P.pp, f^*)$, it generates $s_{f^*} \leftarrow \mathsf{P.Constrain}(\sigma, f^*)$. We show $|\Pr[S_1] - \Pr[S_2]| = \epsilon_{\mathsf{KeySim}}$, where $\epsilon_{\mathsf{KeySim}}$ is the advantage of breaking key simulation game of P and this is negligible. Suppose there exist an adversary $\mathcal{A}_0$ such that $|\Pr[S_1] - \Pr[S_2]|$ is non-negligible, we build an algorithm $\mathcal{C}$ that wins the key simulation game of P with an overwhelming advantage.

1. At the beginning, $\mathcal{C}$ receives $P.pp$. Then, it generates pp and msk as in the previous game.
2. Upon input a bit string $x \in \{0,1\}^l$ for $\mathcal{O}_{\mathsf{KeyGen}}(x)$, $\mathcal{C}$ captures r_x by sending x to the evaluation oracle of the key simulation game. Let $r = r_x$, it answers sk_x as in the previous game.
3. Receive a challenge tuple $(f^*, \alpha^*, (\mu_0^*, \mu_1^*))$. $\mathcal{C}$ sends f^* to the challenge oracle of the key simulation game.

4. Receive sk_{f^*} and let $s_{f^*} = sk_{f^*}$. $\mathcal{C}$ computes ct^* as same as the previous game and returns it to $\mathcal{A}_0$.
5. Answer the subsequent queries as in Step 2.
6. $\mathcal{A}_0$ guesses it is operating with a $\mathbf{Game}_0$ or $\mathbf{Game}_1$ challenger. At last, $\mathcal{C}$ outputs $\mathcal{A}_0$'s guess as the answer to the key simulation game challenge it is trying to distinguish.

If the challenger chooses $b = 1$ in the key simulation game, $\mathcal{C}$ provides a view of $\mathbf{Game}_0$ to $\mathcal{A}_0$. Otherwise, $\mathcal{C}$ provides a view of $\mathbf{Game}_1$ to $\mathcal{A}_0$. In other words, any advantage of $\mathcal{A}_0$ distinguishes between these two games translates to identical advantage in key simulation game. Therefore, if $|\Pr[S_1] - \Pr[S_2]|$ is non-negligible, then $\mathcal{C}$ could break key simulation game with a non-negligible advantage.

Remark 2. We know that the challenge ciphertext or its derivatives can be re-encrypted when they satisfy the restrictions of re-encryption operations, especially $f^*(x) = 1$ and $g(y) = 0$. However, there is a contradiction since we require $f^*(x) = 0$ in challenge oracle, which means x cannot be queried in $\mathcal{O}_{\mathsf{KeyGen}}$ and $\mathcal{O}_{\mathsf{ReKey}}$. In $\mathbf{Game}_0$, to answer these queries, the challenger could instead generate the corresponding sk_x (note that $f^*(x) = 1$) and $rk^{\beta}_{x \to g}$, in which $r \neq r'$ with a non-negligible probability. When r' is computed by $\mathsf{P.ConstrainEval}(s_{f^*}, x)$ where $s_{f^*} \leftarrow \mathsf{P.Constrain}(\sigma, f^*)$, however, we have $r = r'$ in a significant probability since $r \leftarrow \mathsf{P.Eval}(\sigma, x)$, which conflicts with the security proof. To ensure that the challenge ciphertext or its derivatives could be re-encrypted in $\mathbf{Game}_2$, we change the sources of r as shown in $\mathbf{Game}_1$.

$\mathbf{Game}_3$: In this game, we change the way the matrix $\mathbf{A}$ is generated. Recall in the previous game, the challenger chooses $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m\lambda}$. Now it first samples a matrix $\mathbf{R} \xleftarrow{\$} \{0,1\}^{m' \times m\lambda}$ and sets $\mathbf{A} = \mathbf{BR} + \sigma \otimes \mathbf{G}$. Since $m' \geq (n+1)\lceil \log q \rceil + 2\lambda$ and the generalized leftover hash lemma (Definition 2.9), the distribution $(\mathbf{B}, \mathbf{BR})$ is statistically indistinguishable to the distribution $(\mathbf{B}, \mathbf{A})$ where $\mathbf{A}$ is a uniform matrix in $\mathbb{Z}_q^{n \times m\lambda}$. Thus, we have $|\Pr[S_2] - \Pr[S_3]| \leq \mathsf{negl}(\lambda)$.

$\mathbf{Game}_4$: In this game, we change again the way challenge query f^* is answered and the way of generating $\mathbf{u}_1^*$. Concretely, when $\mathcal{A}$ makes a challenge query for $(f^*, w^*, (\mu_0^*, \mu_1^*))$, the challenger computes

$$\begin{aligned} \mathbf{A}_{f^*} - s_{f^*} \otimes \mathbf{G} &= \mathbf{AH}_{\sigma \to f^*} - U_{\sigma \to f^*}(\sigma) \otimes \mathbf{G} \\ &= (\mathbf{A} - \sigma \otimes \mathbf{G})\hat{\mathbf{H}}_{msk \to s_{f^*}} \\ &= \mathbf{BR}\hat{\mathbf{H}}_{msk \to s_{f^*}}, \end{aligned}$$

where $\hat{\mathbf{H}}_{msk \to s_{f^*}} \leftarrow \mathsf{EvalFX}(U_{\sigma \to f^*}, \sigma, \mathbf{A})$.

The way it generates $\mathbf{u}_0^*$, $\mathbf{u}_2^*$ (and $\mathbf{u}_3^*$ if $w \neq \mathsf{Null}$), remains unaltered. Recall in the previous game, by sampling $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$ and $\mathbf{e}_1 \xleftarrow{\$} \tilde{\chi}^{ml_f}$, it computes $\mathbf{u}_1^* =$

$\mathbf{s}^T(\mathbf{A}_{f^*} - s_{f^*} \otimes \mathbf{G}) + \mathbf{e}_1^T$. Now, $\mathbf{u}_1^*$ will be substituted as

$$\begin{aligned}\mathbf{u}_1^* &= \mathbf{u}_0^* \mathbf{R}\hat{\mathbf{H}}_{msk \to s_{f^*}} + \mathbf{e}_1^T \\ &= (\mathbf{s}^T\mathbf{B} + \mathbf{e}_0^T)\mathbf{R}\hat{\mathbf{H}}_{msk \to s_{f^*}} + \mathbf{e}_1^T \\ &= \mathbf{s}^T(\mathbf{A}_{f^*} - s_{f^*} \otimes \mathbf{G}) + \mathbf{e}_0^T\mathbf{R}\hat{\mathbf{H}}_{msk \to s_{f^*}} + \mathbf{e}_1^T,\end{aligned}$$

where $\mathbf{e}_0 \overset{\$}{\leftarrow} \chi^{m'}$. Note that

$$B' = \|\mathbf{e}_0^T\mathbf{R}\hat{\mathbf{H}}_{msk \to s_f}\|_\infty \le m'm\lambda\|\mathbf{e}_0^T\|_\infty\|\mathbf{R}\|_\infty\|\hat{\mathbf{H}}_{msk \to s_{f^*}}\|_\infty \le m'm\lambda B(2m)^{d_{\mathsf{Con}}},$$

in which d_{Con} denotes the depth of $U_{\sigma \to f^*}$. By Definition 2.6, let $\tilde{\chi}$ be B'-swallowing, it holds that $\mathbf{u}_1^*$ generated by two methods are within a negligible statistical distance. Therefore, we have $|\Pr[S_3] - \Pr[S_4]| \le \mathsf{negl}(\lambda)$.

Game$_5$: In this game, we change the way key queries are answered. When $\mathcal{A}$ queries on x, the challenger evaluates $r \leftarrow \mathsf{P.Eval}(\sigma, x)$ and $\hat{\mathbf{H}}_{msk \to r} \leftarrow \mathsf{EvalFX}(U_{\sigma \to x}, \sigma, \mathbf{A})$, and computes

$$\begin{aligned}(\mathbf{A} - \sigma \otimes \mathbf{G})\hat{\mathbf{H}}_{msk \to r} &= \mathbf{A}\mathbf{H}_{\sigma \to x} - U_{\sigma \to x}(\sigma) \otimes \mathbf{G} \\ &= \mathbf{A}\mathbf{H}_{\sigma \to x} - r \otimes \mathbf{G} \\ &= \mathbf{A}_x - r \otimes \mathbf{G}.\end{aligned}$$

Then, we have

$$(\mathbf{A}_x - r \otimes \mathbf{G})\hat{\mathbf{H}}_{r,r} = \mathbf{A}_x\mathbf{H}_r - I_r(r) \otimes \mathbf{G} = \mathbf{A}_{x,r} - \mathbf{G}$$

since $I_r(r) = 1$ and $\hat{\mathbf{H}}_{r,r} \leftarrow \mathsf{EvalFX}(I_r, r, \mathbf{A}_x)$. Therefore, it holds that $\mathbf{B}\mathbf{R}$ $\hat{\mathbf{H}}_{msk \to r}\hat{\mathbf{H}}_{r,r} = \mathbf{A}_{x,r} - \mathbf{G}$ due to $\mathbf{A} - \sigma \otimes \mathbf{G} = \mathbf{B}\mathbf{R}$. Note that

$$\mathbf{B}\|\mathbf{A}_{x,r} = \mathbf{B}\|(\mathbf{B}\mathbf{R}\hat{\mathbf{H}}_{msk \to r}\hat{\mathbf{H}}_{r,r} + \mathbf{G})$$

and

$$\|\mathbf{R}\hat{\mathbf{H}}_{msk \to r}\hat{\mathbf{H}}_{r,r}\|_\infty \le m^2\lambda k\|\mathbf{R}\|_\infty\|\hat{\mathbf{H}}_{msk \to r}\|_\infty\|\hat{\mathbf{H}}_{r,r}\|_\infty \le m^2\lambda k(2m)^{d+1}.$$

Lemma 2.6 and Lemma 2.7 show that when

$$\tau = O(\sqrt{m' + m}\|\mathbf{R}\hat{\mathbf{H}}_{msk \to r}\hat{\mathbf{H}}_{r,r}\|_\infty) = O(\lambda k(2m)^{d+3}),$$

it is efficient to compute $\mathbf{T}_{\mathbf{B}\|\mathbf{A}}$ by

$$\mathbf{T}_{\mathbf{B}\|\mathbf{A}_{x,r}} \leftarrow \mathsf{RandBasis}(\mathbf{B}\|\mathbf{A}_{x,r}, \mathsf{ExtendLeft}(\mathbf{B}, \mathbf{G}, \mathbf{T}_\mathbf{G}, \mathbf{R}\hat{\mathbf{H}}_{msk \to r}\hat{\mathbf{H}}_{r,r}), \tau).$$

Then it runs $\mathsf{SamplePre}$ algorithm as in **Game**$_4$. Besides, the challenger calls $\mathsf{Semi-KeyGen}_2(msk, x)$ algorithm when $\mathcal{A}$ makes a re-encryption query on $(x, (g, y), \beta, (f, \alpha, k))$ such that $g(y) = 1$ and $k \notin \mathsf{Deriv}$. Since the responses to key queries and re-encryption queries are statistically close to those in the previous game, the adversary's advantage is at most negligibly different from its advantage in $\mathbf{Game}_4$. Therefore, we have $|\Pr[S_4] - \Pr[S_5]| \leq \mathsf{negl}(\lambda)$.

$\mathbf{Game}_6$: In this game, we change the way the matrix $\mathbf{B}$ is generated. Concretely, the challenger chooses $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{n\times m'}$ without producing the corresponding trapdoor $\mathbf{T_B}$. By Lemma 2.1, this makes only $2^{-\Omega(n)}$-statistical distance with uniform. The challenger could answer all the key queries without the trapdoor because of the change we made in the $\mathbf{Game}_5$, the view of $\mathcal{A}$ is altered only negligibly. Therefore, we have $|\Pr[S_5] - \Pr[S_6]| \leq \mathsf{negl}(\lambda)$.

$\mathbf{Game}_7$: In this game, we change the way the challenge ciphertext is created. The challenger chooses $(\mathbf{u}_0^*, \mathbf{u}_1^*, \mathbf{u}_2^*, \mathbf{u}_3^*) \in \mathbb{Z}_q^{1\times(m'+ml_f+2l\lceil\log q\rceil)}$ (if $\alpha^* \neq \mathsf{Null}$) at random. Since the challenge ciphertext completely hides b, thus $\mathcal{A}$ has no advantage in this game. We claim that $|\Pr[S_6] - \Pr[S_7]|$ is negligible for a PPT adversary assuming the hardness of DLWE problem. To show that, we do by giving a reduction from DLWE problem.

Reduction from LWE. Suppose $\mathcal{A}$ has a non-negligible advantage in distinguish $\mathbf{Game}_6$ and $\mathbf{Game}_7$. We use $\mathcal{A}$ to construct an LWE algorithm $\mathcal{B}$.

LWE Instance. $\mathcal{B}$ receives an LWE instance as $(\mathbf{B}\|\mathbf{V}, \mathbf{u}_0\|\mathbf{u}_2') \in \mathbb{Z}_q^{n\times(m'+l\lceil\log q\rceil)} \times \mathbb{Z}_q^{1\times(m'+l\lceil\log q\rceil)}$. The task of $\mathcal{B}$ is to distinguish whether $\mathbf{u}_0\|\mathbf{u}_2' = \mathbf{s}^T(\mathbf{B}\|\mathbf{V}) + \bar{\mathbf{e}}$ for some $\mathbf{s} \in \mathbb{Z}_q^n$ and $\bar{\mathbf{e}} \in \chi^{m'+l\lceil\log q\rceil}$ or $\mathbf{u}_0\|\mathbf{u}_2' \xleftarrow{\$} \mathbb{Z}_q^{1\times(m'+l\lceil\log q\rceil)}$.

Phase 1 (Setup): $\mathcal{B}$ sets matrices $\mathbf{B}$ and $\mathbf{V}$ to be the LWE matrices. Note that unlike the real scheme since $\mathcal{B}$ does not require the trapdoor $\mathbf{T_B}$ of matrix $\mathbf{B}$ (i.e., the change we made in $\mathbf{Game}_6$). It assembles public parameters *pp* and master secret key as in the previous game: run $(P.pp, P.msk) \leftarrow \mathsf{P.Setup}(1^\lambda)$, set $\sigma = P.msk$ and sample matrices $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m\lambda}$ and $\mathbf{D} \xleftarrow{\$} \mathbb{Z}_q^{n\times l\lceil\log q\rceil}$. It gives $\mathcal{A}$ the public parameters

$$pp = (\mathbf{B}, \mathbf{A}, \mathbf{V}, \mathbf{D}, P.pp).$$

Then, $\mathcal{B}$ initializes a counter $\mathsf{numCt} := 0$, a policy-value store $\mathsf{C} := \emptyset$ and a set $\mathsf{Deriv} := \emptyset$.

Phase 2 (Oracle query): $\mathcal{B}$ answers $\mathcal{A}$'s key queries, encryption queries, re-encryption key queries and re-encryption queries as in $\mathbf{Game}_7$, except that challenge query.

- $\mathcal{O}_{\mathsf{Cha}}(f^*, \alpha^*, (\mu_0^*, \mu_1^*))$: To generate the challenge ciphertext, $\mathcal{B}$ first picks $b \in \{0,1\}$. It computes $s_{f^*}, \mathbf{u}_1^*, \mathbf{u}_3^*$ as in $\mathbf{Game}_7$ and $ct^* = (s_{f^*}, \mathbf{u}_0^* = \mathbf{u}_0, \mathbf{u}_1^*, \mathbf{u}_2^* = \mathbf{u}_2' + \mu_b(\mathbf{0}\|\mathbf{g}^T), \mathbf{u}_3^*)$ (if $\alpha^* \neq \mathsf{Null}$), sets $\mathsf{numCt} := \mathsf{numCt} + 1$ and $\mathsf{Deriv} := \mathsf{Deriv} \cup \mathsf{numCt}$. $\mathcal{B}$ adds ct^* in C with policy $(f^*, \alpha^*, \mathsf{numCt})$ and gives (numCt, ct^*) to $\mathcal{A}$.

Phase 3 (Decision): At the end of the game, $\mathcal{A}$ outputs if it is operating with a $\mathbf{Game}_6$ or $\mathbf{Game}_7$ challenger. $\mathcal{B}$ outputs $\mathcal{A}$'s guess as the answer to the LWE challenge it is trying to distinguish.

It can be seen that if $(\mathbf{B}\|\mathbf{V}, \mathbf{u}_0\|\mathbf{u}_2')$ is a valid LWE instance (i.e., $(\mathbf{u}_0\|\mathbf{u}_2') = \mathbf{s}^T(\mathbf{B}\|\mathbf{V}) + \bar{\mathbf{e}}$), the view of the adversary corresponds to $\mathbf{Game}_6$. Otherwise (i.e., $(\mathbf{u}_0\|\mathbf{u}_2') \xleftarrow{\$} \mathbb{Z}_q^{1\times(m'+l\lceil\log q\rceil)}$), it corresponds to $\mathbf{Game}_7$. First, observe that $\mathbf{u}_1^* = \mathbf{u}_0\mathbf{R}\hat{\mathbf{H}}_{msk\to s_{f^*}} + \bar{\mathbf{e}}_1^T$ is uniform and independent in $\mathbb{Z}_q^{1\times ml_f}$ by a standard application of the leftover hash lemma (Definition 2.9). Moreover, for any condition vector $\alpha^* \in \mathbb{Z}_q^l$, we have $\mathbf{u}_3^* = \mathbf{s}^T(\mathbf{D} + \mathbf{W} \otimes \mathsf{P2}(\alpha^*)^T) + \bar{\mathbf{e}}_3^T$ where the distribution of $\mathbf{D} + \mathbf{W} \otimes \mathsf{P2}(\alpha^*)^T$ is statistically close to the uniform distribution over $\mathbb{Z}_q^{n\times l\lceil\log q\rceil}$. We therefore conclude that supposing the hardness of DLWE problem we have $|\Pr[S_6] - \Pr[S_7]| \le \mathsf{negl}(\lambda)$.

Therefore, combing the above conclusions together, the theorem is proven. ■

5 AB-PRE and AB-CPRE

Recently, Susilo et al. [24] and Liang et al. [16] proposed AB-PRE and AB-CPRE constructions, respectively, which are based on (KP)ABE [5]. In this section, we show that our CAB-PRE notion implies these two primitives while remaining at the same level of HRA security.

AB-PRE. Recall that the definition of KP-ABPRE in [24], CP-ABPRE can be directly derived from our CAB-PRE scheme by removing the condition while canceling some condition requirements in corresponding security game (i.e., all conditions are matched by default), which is a solution to the open problem proposed in [24]. We build a CP-ABPRE scheme as follows. Note that these algorithms are identical to Section 4 except that remove Enc_2 algorithm and make some modifications.

- $\mathsf{Setup}(1^\lambda)$: Choose $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_q^n$. Output $pp = (\mathbf{B}, \mathbf{A}, \mathbf{v}, P.pp)$ and $msk = (\sigma, \mathbf{T_B})$.
- $\mathsf{KeyGen}(msk, x)$: Sample a short $\mathbf{k}$ such that $(\mathbf{B}||\mathbf{A}_{x,r})\mathbf{k} = \mathbf{v}$. Output $sk_x = (r, \mathbf{T}_{\mathbf{B}||\mathbf{A}_{x,r}}, \mathbf{k})$.
- $\mathsf{Enc}(f, \mu)$: Compute $u_2 = \mathbf{s}^T\mathbf{v} + e_2 + \mu\lfloor q/2\rceil$ where $e_2 \xleftarrow{\$} \chi$. Output $ct = (s_f, \mathbf{u}_0, \mathbf{u}_1, u_2)$.
- $\mathsf{Dec}(sk_x, ct)$: Output 1 if $|u_2 - (\mathbf{u}_0\|\mathbf{u}_1\hat{\mathbf{H}}_{s_f\to r'}\hat{\mathbf{H}}_{r,r'})\mathbf{k}| > q/4$. Similar to CAB-PRE, the noise bound will be changed as $q/4\cdot\delta$ for the transformed ciphertext where δ is a coefficient factor.
- $\mathsf{ReKeyGen}(sk_x, g)$: Let $\mathbf{v}' = \mathbf{v}\cdot\delta$, sample a short $\mathbf{d}$ such that $(\mathbf{B}||\mathbf{A}_{x,r})\mathbf{d} = \mathbf{v}'$ and set
$$\mathbf{Z} = \begin{pmatrix} \mathbf{r}_1\mathbf{H} + \mathbf{r}_2 & \mathbf{r}_1\mathbf{v} + \mathbf{r}_3 - \mathsf{P2}(\mathbf{d}) \\ \mathbf{0}_{1\times(m'+ml_g)} & \delta \end{pmatrix},$$
in which $\mathbf{r}_1 \in \chi^{(m'+m)\lceil\log q\rceil\times n}$, $\mathbf{r}_2 \in \chi^{(m'+m)\lceil\log q\rceil\times(m'+ml_g)}$, $\mathbf{r}_3 \in \chi^{(m'+m)\lceil\log q\rceil}$, $\delta \in \chi$. Output $rk_{x\to g} = (s_g, r, \delta, \mathbf{Z})$
- $\mathsf{ReEnc}(rk_{x\to g}, ct)$: Evaluate $ct_{f\to g} = (\mathsf{BD}(\mathbf{u}_0\|\mathbf{u}_1')\|u_2)\cdot\mathbf{Z}$. Output $ct' = (s_g, \delta, ct_{f\to g})$.

Correctness. When operating the decryption algorithm, we have

$$u_2 - (\mathbf{u}_0 \| \mathbf{u}_1 \hat{\mathbf{H}}_{s_f \to r'} \hat{\mathbf{H}}_{r,r'})\mathbf{k} = \mu \lfloor q/2 \rceil + e_2 - (\mathbf{e}_0^T \| \mathbf{e}_1')\mathbf{k}.$$

By our parameter choices in Section 4.1, the norm of error term is bounded by

$$|e_2 - (\mathbf{e}_0^T \| \mathbf{e}_1')\mathbf{k}| \le B \cdot \mathsf{poly}(n \lceil \log q \rceil) \cdot (2m)^{d_{\mathsf{ConEval}}+d+4} < q/4.$$

The proof of security remains mostly unaltered compared with CAB-PRE, except that in the reduction to LWE problem $\mathcal{B}$ receives an LWE instance as $(\mathbf{B} \| \mathbf{v}, \mathbf{u}_0 \| u_2') \in \mathbb{Z}_q^{n \times (m'+1)} \times \mathbb{Z}_q^{1 \times (m'+1)}$, enabling $\mathcal{B}$ to generate a challenge ciphertext $(\mathbf{u}_0^*, \mathbf{u}_1^*, u_2^*) = (\mathbf{u}_0, \mathbf{u}_0 \mathbf{R} \hat{\mathbf{H}}_{msk \to s_{f^*}} + \bar{\mathbf{e}}_1^T, u_2' + \mu_b \lfloor q/2 \rceil)$.

AB-CPRE. Recall that the definition of (KP)AB-CPRE in [16], our CAB-PRE primitive could also be constructed as (CP)AB-CPRE. Concretely, since the underlying structure of AB-CPRE is public-key encryption where the access policy is used to control conditions in the re-encryption phase, thus we comply with [16]'s definition and give a concrete (CP)AB-CPRE scheme as follows:

- $\mathsf{Setup}(1^\lambda)$: Suppose there are n users in the system. For $i \in [n]$, sample matrices $\mathbf{A}_i \xleftarrow{\$} \mathbb{Z}_q^{n \times m\lambda}$ as public parameters, namely $pp = \{\mathbf{A}_i\}_{i \in [n]}$.
- $\mathsf{KeyGen}(pp, \alpha)$: Run $(P.pp_\alpha, P.msk) \leftarrow \mathsf{P.Setup}(1^\lambda)$ and set $\sigma_\alpha = P.msk$. Invoke $(\mathbf{B}_\alpha, \mathbf{T}_{\mathbf{B}_\alpha}) \leftarrow \mathsf{TrapGen}(1^n, m', q)$. Sample a vector $\mathbf{v}_\alpha \xleftarrow{\$} \mathbb{Z}_q^n$ and compute $\mathbf{k}_\alpha \leftarrow \mathsf{SamplePre}(\mathbf{B}_\alpha, \mathbf{T}_{\mathbf{B}_\alpha}, \mathbf{v}_\alpha, \tau)$ such that $\mathbf{B}_\alpha \mathbf{k}_\alpha = \mathbf{v}_\alpha$. Output $pk_\alpha = (\mathbf{B}_\alpha, \mathbf{v}_\alpha, P.pp_\alpha)$ and $sk_\alpha = (\mathbf{T}_{\mathbf{B}_\alpha}, \mathbf{k}_\alpha, \sigma_\alpha)$.
- $\mathsf{Enc}(pk_\alpha, f, \mu)$: Compute $s_f \leftarrow \mathsf{P.KeySim}(P.pp_\alpha, f)$. Choose $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$,$\mathbf{e}_0 \xleftarrow{\$} \chi^{m'}$,$\mathbf{e}_1 \xleftarrow{\$} \tilde{\chi}^{ml_f}$,$e_2 \xleftarrow{\$} \chi$, and evaluate

$$\mathbf{u}_0 = \mathbf{s}^T \mathbf{B}_\alpha + \mathbf{e}_0^T,$$

$$\mathbf{u}_1 = \mathbf{s}^T (\mathbf{A}_f - s_f \otimes \mathbf{G}) + \mathbf{e}_1^T,$$

$$u_2 = \mathbf{s}^T \mathbf{v}_\alpha + e_2 + \mu \lfloor q/2 \rceil,$$

  where $\mathbf{A}_f = \mathbf{A}_\alpha \mathbf{H}_{\sigma_\alpha \to f}$, $\mathbf{H}_{\sigma_\alpha \to f} \leftarrow \mathsf{EvalF}(U_{\sigma_\alpha \to f}, \mathbf{A}_\alpha)$. In particularly, $\mathbf{u}_1$ denotes a transition condition and does not participate in decryption. Output $ct_\alpha = (s_f, \mathbf{u}_0, \mathbf{u}_1, u_2)$.
- $\mathsf{Dec}(sk_\alpha, ct_\alpha)$: Parse $sk_\alpha = (\mathbf{T}_{\mathbf{B}_\alpha}, \mathbf{k}_\alpha, \sigma_\alpha)$ and $ct_\alpha = (s_f, \mathbf{u}_0, \mathbf{u}_1, u_2)$, compute

$$u = u_2 - \mathbf{u}_0 \mathbf{k}_\alpha.$$

  Output 1 iff $|u| > q/4$, otherwise output 0.
- $\mathsf{ReKeyGen}(sk_\alpha, pk_\beta, x)$: Parse $sk_\alpha = (\mathbf{T}_{\mathbf{B}_\alpha}, \mathbf{k}_\alpha, \sigma_\alpha)$ and $pk_\beta = (\mathbf{B}_\beta, \mathbf{v}_\beta, P.pp_\beta)$. Compute $\mathbf{H}_{\sigma_\alpha \to x} \leftarrow \mathsf{EvalF}(U_{\sigma_\alpha \to x}, \mathbf{A}_\alpha)$, let $\mathbf{A}_x = \mathbf{A}_\alpha \mathbf{H}_{\sigma_\alpha \to x}$. Evaluate $r \leftarrow \mathsf{P.Eval}(\sigma_\alpha, x)$ and $\mathbf{H}_r \leftarrow \mathsf{EvalF}(I_r, \mathbf{A}_x)$, set $\mathbf{A}_{x,r} = \mathbf{A}_x \mathbf{H}_r$. Sample $\mathbf{d} \leftarrow \mathsf{SampleLeft}(\mathbf{B}_\alpha, \mathbf{A}_{x,r}, \mathbf{T}_{\mathbf{B}_\alpha}, \mathbf{v}_\alpha, \tau)$ such that $(\mathbf{B}_\alpha \| \mathbf{A}_{x,r})\mathbf{d} = \mathbf{v}_\alpha$. Choose

randomly $\mathbf{r}_1 \in \chi^{(m'+m)\lceil \log q \rceil \times n}, \mathbf{r}_2 \in \chi^{(m'+m)\lceil \log q \rceil \times m'}, \mathbf{r}_3 \in \chi^{(m'+m)\lceil \log q \rceil}$, set

$$\mathbf{Z} = \begin{pmatrix} \mathbf{r}_1\mathbf{B}_\beta + \mathbf{r}_2 & \mathbf{r}_1\mathbf{v}_\beta + \mathbf{r}_3 - \mathsf{P2}(\mathbf{d}) \\ \mathbf{0}_{1\times m'} & 1 \end{pmatrix}.$$

Output $rk^x_{\alpha\to\beta} = (r, \mathbf{Z})$.

– $\mathsf{ReEnc}(rk^x_{\alpha\to\beta}, ct_\alpha)$: Parse $rk^x_{\alpha\to\beta} = (r, \mathbf{Z})$ and $ct_\alpha = (s_f, \mathbf{u}_0, \mathbf{u}_1, u_2)$. Compute $r' \leftarrow U_{f\to x}(s_f)$, abort if $r' = r$. Otherwise, compute $\mathbf{A}_x, \mathbf{A}_f$ as in $\mathsf{ReKeyGen}, \mathsf{Enc}$ algorithms and

$$\hat{\mathbf{H}}_{r,r'} \leftarrow \mathsf{EvalFX}(I_r, r', \mathbf{A}_x),$$

$$\hat{\mathbf{H}}_{s_f\to r'} \leftarrow \mathsf{EvalFX}(U_{f\to x}, s_f, \mathbf{A}_f).$$

Let $\mathbf{u}'_1 = \mathbf{u}_1\hat{\mathbf{H}}_{s_f\to r'}\hat{\mathbf{H}}_{r,r'}$, evaluate

$$ct_{\alpha\to\beta} = (\mathsf{BD}(\mathbf{u}_0\|\mathbf{u}'_1)\|u_2)\cdot\mathbf{Z}.$$

Output $ct_{\alpha\to\beta}$.

Correctness. When the decryption algorithm operates as specified, we have

$$u_2 - \mathbf{u}_0\mathbf{k}_\alpha = \mu\lfloor q/2\rceil + e_2 - \mathbf{e}_0^T\mathbf{k}_\alpha.$$

Based on our parameter choices in Section 4.1, the norm of error term is bounded by

$$|e_2 - \mathbf{e}_0^T\mathbf{k}_\alpha| \le B + m'B\cdot\tau\sqrt{m'} < q/4.$$

The following theorem proves the HRA security of (CP)AB-CPRE.

Theorem 5.1. For a class family $\mathcal{F}$, if P be a conforming cPRF, then the above unidirectional, single-hop (CP)AB-CPRE is a HRA-secure scheme under the hardness of $\mathrm{DLWE}_{n,q,\chi}$ problem.

Proof. This security proof is a sequence of interactive games between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. In each game, we define S_i to be the event that $\mathcal{A}$ wins in $\mathbf{Game}_i$.

$\mathbf{Game}_0$: This is the original security game.

Phase 1 (Setup): At the beginning of the game, the challenger chooses $\mathbf{A}_i \xleftarrow{\$} \mathbb{Z}_q^{n\times m\lambda}(i \in [n])$ as public parameters and give them to $\mathcal{A}$. Besides $\mathsf{numCt}, \mathsf{C}, \mathsf{Deriv}$, it initializes two empty sets $\Phi_H := \emptyset$ and $\Phi_C := \emptyset$.

Phase 2 (Oracle query): These queries are handled as follows.

– $\mathcal{O}^H_{\mathsf{KeyGen}}(pp, \eta)$: For an honest key query η, the challenger generates $(pk_\eta, sk_\eta) \leftarrow \mathsf{KeyGen}(pp, \eta)$ and adds it to Φ_H. $\mathcal{C}$ gives pk_η to $\mathcal{A}$.

– $\mathcal{O}^C_{\mathsf{KeyGen}}(pp, \eta)$: For a corrupt key query η, the challenger generates $(pk_\eta, sk_\eta) \leftarrow \mathsf{KeyGen}(pp, \eta)$ and adds it to Φ_C. $\mathcal{C}$ gives the keypair to $\mathcal{A}$.

– $\mathcal{O}_{\mathsf{Enc}}(pk_\alpha, f, \mu)$: For an encryption query (pk_α, f, μ) if $f \ne \mathsf{Null}$, the challenger computes $ct = \mathsf{Enc}(pk_\alpha, f, \mu)$, sets $\mathsf{numCt} := \mathsf{numCt} + 1$, adds ct in C with $(pk_\alpha, f, \mathsf{numCt})$, and gives (numCt, ct) to $\mathcal{A}$.

– $\mathcal{O}_{\mathsf{ReKey}}(\alpha, \beta, x)$: For a re-encryption key query (α, β, x), there are two cases:
  1. $pk_\beta \in \Phi_H$, the challenger chooses a random $\mathbf{Z}$.
  2. $pk_\beta \in \Phi_C$, return $\perp$ if $pk_\alpha \in \Phi_H$ and $f(x) = 1$ where $f \in \mathcal{O}_{\mathsf{Enc}}$. Otherwise, the challenger produces $rk^x_{\alpha\to\beta} \leftarrow \mathsf{ReKeyGen}(sk_\alpha, pk_\beta, x)$.

  After that, $\mathcal{C}$ gives $rk^x_{\alpha\to\beta}$ to $\mathcal{A}$.
– $\mathcal{O}_{\mathsf{Cha}}(\theta, f^*, (\mu_0^*, \mu_1^*))$: This oracle can only be invoked once. For a challenge query $(\theta, f^*, (\mu_0^*, \mu_1^*))$ (if $f^* \neq \mathsf{Null}$), it requires $pk_\theta \in \Phi_H$. The challenger flips a bit $b \in \{0, 1\}$, generates $ct^* \leftarrow \mathsf{Enc}(pk_\theta, f^*, \mu_b^*)$, sets $\mathsf{numCt} := \mathsf{numCt} + 1$ and $\mathsf{Deriv} := \mathsf{Deriv} \cup \mathsf{numCt}$. It adds ct^* in C with $(pk_\theta, f^*, \mathsf{numCt})$ and gives (numCt, ct^*) to $\mathcal{A}$.
– $\mathcal{O}_{\mathsf{ReEnc}}(\alpha, \beta, x, (pk_\alpha, f, k))$: For a re-encryption query $(\alpha, \beta, x, (pk_\alpha, f, k))$ where $k \leq \mathsf{numCt}$, the challenger does the following operations.
  1. If there is no value in C with policy $(pk_\alpha, f \neq \mathsf{Null}, k)$, return $\perp$.
  2. If $f(x) = 0$, return $\perp$.
  3. If $pk_\beta \in \Phi_C \cap k \in \mathsf{Deriv}$, return $\perp$.
  4. Otherwise, let ct_α be that value in C. The challenger first captures sk_α by checking Φ_H or Φ_C, which depends on $pk_\alpha \in \Phi_H$ or $pk_\alpha \in \Phi_C$. Then, it computes $ct_{\alpha\to\beta} \leftarrow \mathsf{ReEnc}(rk^x_{\alpha\to\beta}, ct_\alpha)$ where $rk^x_{\alpha\to\beta} \leftarrow \mathsf{ReKeyGen}(sk_\alpha, pk_\beta, x)$, sets $\mathsf{numCt} := \mathsf{numCt} + 1$, adds $ct_{\alpha\to\beta}$ in C with $(pk_\beta, \mathsf{Null}, \mathsf{numCt})$. If $k \in \mathsf{Deriv}$, set $\mathsf{Deriv} := \mathsf{Deriv} \cup \mathsf{numCt}$. Finally, it gives $(\mathsf{numCt}, ct_{\alpha\to\beta})$ to $\mathcal{A}$.

Phase 3 (Decision): This is the decision phase. $\mathcal{A}$ outputs a bit b' for b.
By definition, we have $|\Pr[S_0] - 1/2| \leq \mathsf{negl}(\lambda)$.

Game$_1$: In this game, we change $r \xleftarrow{\$} \{0, 1\}^k$ to replace $r \leftarrow \mathsf{P.Eval}(\sigma_\alpha, x)$ when generating $rk^x_{\alpha\to\beta}$, if it happens to the case of $k \in \mathsf{Deriv}$ in re-encryption queries, which is identical to **Game**$_1$ in Section 4.2. Since the pseudorandomness security of P, we have $|\Pr[S_0] - \Pr[S_1]| \leq \mathsf{negl}(\lambda)$.

Game$_2$: In this game, we change the way s_{f^*} for a target function is created, which is identical to **Game**$_2$ in Section 4.2 if $f^* \neq \mathsf{Null}$. Since P has a property of key simulation, we have $|\Pr[S_1] - \Pr[S_2]| = \epsilon_{\mathsf{KeySim}} \leq \mathsf{negl}(\lambda)$.

Game$_3$: In this game, we change the way the matrices $\mathbf{A}_i$ are generated, which are identical to **Game**$_3$ in Section 4.2. Precisely, the challenger first runs $(P.pp_i, P.msk_i) \leftarrow \mathsf{P.Setup}(1^\lambda)$ for system users and sets $\sigma_i = P.msk_i$. It then samples matrices $\mathbf{R}_i \xleftarrow{\$} \{0, 1\}^{m' \times m\lambda}$ and let $\mathbf{A}_i = \mathbf{B}_i\mathbf{R}_i + \sigma_i \otimes \mathbf{G}$ for $i \in [n]$. Since $m' \geq (n + 1)\lceil \log q \rceil + 2\lambda$ and the generalized leftover hash lemma (Definition 2.9), we have $|\Pr[S_2] - \Pr[S_3]| \leq \mathsf{negl}(\lambda)$.

Game$_4$: In this game, we change the way challenge query f^* is answered and the way of generating $\mathbf{u}_1^*$, which is identical to **Game**$_4$ in Section 4.2 if $f^* \neq \mathsf{Null}$. By parameter settings and Definition 2.6, we have $|\Pr[S_3] - \Pr[S_4]| \leq \mathsf{negl}(\lambda)$.

Game$_5$: In this game, we change the way re-encryption key queries are answered if $pk_\beta \in \Phi_C$, which is similar to **Game**$_5$ in Section 4.2. Based on the computations of **Game**$_5$ in Section 4.2, we have

$$\mathbf{B}_\alpha \| \mathbf{A}_{x,r} = \mathbf{B}_\alpha \| (\mathbf{B}_\alpha \mathbf{R}_\alpha \hat{\mathbf{H}}_{\sigma_\alpha \to r} \hat{\mathbf{H}}_{r,r} + \mathbf{G}).$$

Let $\mathbf{R} = \mathbf{R}_\alpha \hat{\mathbf{H}}_{\sigma_\alpha \to r} \hat{\mathbf{H}}_{r,r}$, the challenger samples

$$\mathbf{d} \leftarrow \mathsf{SampleRight}(\mathbf{B}_\alpha, \mathbf{G}, \mathbf{R}, \mathbf{T}_\mathbf{G}, \mathbf{v}_\alpha, \tau).$$

In the meanwhile, the challenger could answer re-encryption queries on the case of $pk_\beta \in \Phi_C$ and $k \notin \mathsf{Deriv}$, instead of obtaining $\mathbf{T}_{\mathbf{B}_\alpha}$. By Lemma 2.3 and Lemma 2.4, we conclude that $|\Pr[S_4] - \Pr[S_5]| \le \mathsf{negl}(\lambda)$.

Game$_6$: In this game, we change the way the matrix $\mathbf{B}_i$ are generated, which are identical to **Game**$_6$ in Section 4.2. Note that for all key queries of honest users, the challenger chooses $\mathbf{B}_\eta \overset{\$}{\leftarrow} \mathbb{Z}_q^{n \times m'}$ and $\mathbf{k}_\eta \overset{\$}{\leftarrow} \chi^{m'}$, computes $\mathbf{B}_\eta \mathbf{k}_\eta = \mathbf{v}_\eta$. Due to the properties of $\mathsf{TrapGen}$ and $\mathsf{SamplePre}$ algorithms (Lemma 2.1 and Lemma 2.2), it is statistically close to the previous game. Thus, we have $|\Pr[S_5] - \Pr[S_6]| \le \mathsf{negl}(\lambda)$.

Game$_7$: In this game, we change the way the challenge ciphertext is created, which is identical to **Game**$_7$ in Section 4.2. Since the challenge ciphertext completely hides b, thus $\mathcal{A}$ has no advantage in this game. We claim that $|\Pr[S_6] - \Pr[S_7]|$ is negligible for a PPT adversary assuming the hardness of the DLWE problem. This reduction proceeds as same as the above AB-PRE scheme.

Therefore, by integrating the conclusions above, the theorem is proven. ■

6 Conclusion

In this work, we addressed the problem of how to simultaneously realize the underlying construction and the delegation in PRE at a fine-grained level. We presented the concept of conditional attribute-based proxy encryption (CAB-PRE). We formalized its definition and adaptive HRA security model and proposed a lattice-based CAB-PRE scheme. Besides, we further show that CAB-PRE implies AB-PRE and AB-CPRE and gave their HRA-secure constructions. The downside is that the dimension will expand in re-encryption key generation due to the key switching technique. Therefore, it remains an open problem how to control it and construct CCA-secure CAB-PRE over lattices.

References

1. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer (2010), https://doi.org/10.1007/978-3-642-13190-5_28

2. Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 595–618. Springer (2009), `https://doi.org/10.1007/978-3-642-03356-8_35`
3. Ateniese, G., Benson, K., Hohenberger, S.: Key-private proxy re-encryption. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 279–294. Springer (2009), `https://doi.org/10.1007/978-3-642-00862-7_19`
4. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer (1998), `https://doi.org/10.1007/BFb0054122`
5. Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 533–556. Springer (2014), `https://doi.org/10.1007/978-3-642-55220-5_30`
6. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) ITCS 2012. pp. 309–325. ACM (2012), `https://doi.org/10.1145/2090236.2090262`
7. Brakerski, Z., Vaikuntanathan, V.: Circuit-abe from LWE: unbounded attributes and semi-adaptive security. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9816, pp. 363–384. Springer (2016), `https://doi.org/10.1007/978-3-662-53015-3_13`
8. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer (2010), `https://doi.org/10.1007/978-3-642-13190-5_27`
9. Cohen, A.: What about bob? the inadequacy of CPA security for proxy reencryption. In: Lin, D., Sako, K. (eds.) PKC 2019. LNCS, vol. 11443, pp. 287–316. Springer (2019), `https://doi.org/10.1007/978-3-030-17259-6_10`
10. Davidson, A., Katsumata, S., Nishimaki, R., Yamada, S., Yamakawa, T.: Adaptively secure constrained pseudorandom functions in the standard model. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. LNCS, vol. 12170, pp. 559–589. Springer (2020), `https://doi.org/10.1007/978-3-030-56784-2_19`
11. Döttling, N., Nishimaki, R.: Universal proxy re-encryption. In: Garay, J.A. (ed.) PKC 2021. LNCS, vol. 12710, pp. 512–542. Springer (2021), `https://doi.org/10.1007/978-3-030-75245-3_19`
12. Fuchsbauer, G., Kamath, C., Klein, K., Pietrzak, K.: Adaptively secure proxy reencryption. In: Lin, D., Sako, K. (eds.) PKC 2019. LNCS, vol. 11443, pp. 317–346. Springer (2019), `https://doi.org/10.1007/978-3-030-17259-6_11`
13. Ge, C., Susilo, W., Fang, L., Wang, J., Shi, Y.: A cca-secure key-policy attribute-based proxy re-encryption in the adaptive corruption model for dropbox data sharing system. Des. Codes Cryptogr. **86**(11), 2587–2603 (2018)
14. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Dwork, C. (ed.) STOC 2008. pp. 197–206. ACM (2008), `https://doi.org/10.1145/1374376.1374407`
15. Liang, X., Cao, Z., Lin, H., Shao, J.: Attribute based proxy re-encryption with delegating capabilities. In: Li, W., Susilo, W., Tupakula, U.K., Safavi-Naini, R., Varadharajan, V. (eds.) ASIACCS 2009. pp. 276–286. ACM (2009), `https://doi.org/10.1145/1533057.1533094`
16. Liang, X., Weng, J., Yang, A., Yao, L., Jiang, Z., Wu, Z.: Attribute-based conditional proxy re-encryption in the standard model under LWE. In: Bertino, E.,

Shulman, H., Waidner, M. (eds.) ESORICS 2021. LNCS, vol. 12973, pp. 147–168. Springer (2021), https://doi.org/10.1007/978-3-030-88428-4_8

17. Luo, F., Al-Kuwari, S., Wang, F., Chen, K.: Attribute-based proxy re-encryption from standard lattices. Theor. Comput. Sci. **865**, 52–62 (2021)
18. Mao, X., Li, X., Wu, X., Wang, C., Lai, J.: Anonymous attribute-based conditional proxy re-encryption. In: Au, M.H., Yiu, S., Li, J., Luo, X., Wang, C., Castiglione, A., Kluczniak, K. (eds.) NSS 2018. LNCS, vol. 11058, pp. 95–110. Springer (2018), https://doi.org/10.1007/978-3-030-02744-5_7
19. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer (2012), https://doi.org/10.1007/978-3-642-29011-4_41
20. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on gaussian measures. In: FOCS 2004. pp. 372–381. IEEE Computer Society (2004), https://doi.org/10.1109/FOCS.2004.72
21. Paterson, K.G., Schuldt, J.C.N., Stam, M., Thomson, S.: On the joint security of encryption and signature, revisited. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 161–178. Springer (2011), https://doi.org/10.1007/978-3-642-25385-0_9
22. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In: Mitzenmacher, M. (ed.) STOC 2009. pp. 333–342. ACM (2009), https://doi.org/10.1145/1536414.1536461
23. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) STOC 2005. pp. 84–93. ACM (2005), https://doi.org/10.1145/1060590.1060603
24. Susilo, W., Dutta, P., Duong, D.H., Roy, P.S.: Lattice-based hra-secure attribute-based proxy re-encryption in standard model. In: Bertino, E., Shulman, H., Waidner, M. (eds.) ESORICS 2021. LNCS, vol. 12973, pp. 169–191. Springer (2021), https://doi.org/10.1007/978-3-030-88428-4_9
25. Tsabary, R.: Fully secure attribute-based encryption for t-cnf from LWE. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11692, pp. 62–85. Springer (2019), https://doi.org/10.1007/978-3-030-26948-7_3
26. Vasco, M.I.G., Hess, F., Steinwandt, R.: Combined schemes for signature and encryption: The public-key and the identity-based setting. Inf. Comput. **247**, 1–10 (2016)
27. Weng, J., Deng, R.H., Ding, X., Chu, C., Lai, J.: Conditional proxy re-encryption secure against chosen-ciphertext attack. In: Li, W., Susilo, W., Tupakula, U.K., Safavi-Naini, R., Varadharajan, V. (eds.) ASIACCS 2009. pp. 322–332. ACM (2009), https://doi.org/10.1145/1533057.1533100
28. Yang, Y., Lu, H., Weng, J., Zhang, Y., Sakurai, K.: Fine-grained conditional proxy re-encryption and application. In: Chow, S.S.M., Liu, J.K., Hui, L.C.K., Yiu, S. (eds.) ProvSec 2014. LNCS, vol. 8782, pp. 206–222. Springer (2014), https://doi.org/10.1007/978-3-319-12475-9_15