Revocable policy-based chameleon hash using lattices

Klamti, Jean Belo; Hasan, Mohammed Anwarul

Publisher's version / Version de l'éditeur:
https://doi.org/10.1515/jmc-2023-0012
Journal of Mathematical Cryptology, 18, 1, pp. 1-27, 2024-11-28

Research Article

Jean Belo Klamti* and Mohammed Anwarul Hasan

Revocable policy-based chameleon hash using lattices

https://doi.org/10.1515/jmc-2023-0012
received May 15, 2023; accepted September 16, 2024

Abstract: A chameleon hash function is a type of hash function that involves a trapdoor to help find collisions, i.e., it allows the rewriting of a message without modifying the hash. For some applications, it is important to have the feature of revoking the rewriting privilege of the trapdoor holder. In this paper, using lattice-based hard problems that are considered quantum-safe, we first introduce a lattice-based chameleon hash with an ephemeral trapdoor (CHET) and then a revocable attribute-based encryption (RABE) scheme that is adaptively indistinguishable. We also give security analyses of our schemes and compare our RABE scheme to two relevant schemes proposed recently. Furthermore, we combine our CHET and RABE to design a new revocable policy-based chameleon hash.

Keywords: lattice-based cryptography, hash function, chameleon hash function, attribute-based encryption

MSC 2020: 94A60

1 Introduction

Chameleon hash (CH) function, introduced by Krawczyk and Rabin [1], is a collision-resistant hash function that involves a trapdoor, without which, it has all the properties of the standard cryptographic hash function. Specifically, CH schemes are parameterized by public keys such that a trapdoor holder can compute a second pre-image of a given hash. As an extended variant, a chameleon hash with an ephemeral trapdoor (CHET) is a CH scheme in which in addition to the main trapdoor, an ephemeral trapdoor is necessary to compute a second pre-image of the given hash [2]. The ephemeral trapdoor is usually chosen by the party computing the hash value. Therefore, a holder of the main trapdoor is not able to compute a second pre-image of the hash unless also provided with the ephemeral trapdoor. Since their introduction, CH and CHET schemes have received considerable attention and have now become a part of the construction of many cryptographic primitives. They are used, for instance, in the design of chameleon, sanitizable, and online/offline signature schemes [3–7].

One of the recent and important applications of CH is in the design of a mutable blockchain. It was the work of Ateniese et al. [8] who first proposed a way to allow blockchain rewriting. However, one issue of the Ateniese et al. scheme is that any trapdoor holder can rewrite a block of its choice. To address this issue, Derler et al. [9] recently introduced the concept of policy-based chameleon (PCH), which is a type of CH where for computing a second pre-image of a hash, only the trapdoor holders satisfying its access policy are authorized to do it. PCH schemes are designed by combining CH and attribute-based encryption (ABE) schemes[1].

1 In an ABE scheme, a user's secret key is generated by a key generation center using a master secret key, and only users with attributes satisfying the access policy can have the decryption privilege.

* **Corresponding author: Jean Belo Klamti,** National Research Council Canada, Ottawa, Ontario, Canada,
e-mail: jeanbelo.klamti@nrc-cnrc.gc.ca
Mohammed Anwarul Hasan: Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada,
e-mail: ahasan@uwaterloo.ca

Although access policies allow the controlling of computation of the second pre-image in PCH schemes, a compromised trapdoor or malicious behaviour can constitute a serious security issue. To address this issue, Xu et al. have recently introduced the concept of revocable policy-based chameleon hash (RPCH) [10], which allows the revoking of the rewriting privilege of trapdoor holders. Similar to PCH, RPCH schemes are designed by combining revocable attribute-based encryption (RABE) and CHET schemes.

In this paper, we construct a lattice-based CHET by applying Derler et al.'s approach to a modified version of Cash et al.'s CH [11]. Furthermore and still using lattices, we introduce an RABE scheme that is adaptively indistinguishable. In addition, we combine our CHET and RABE schemes to design a new RPCH.

1.1 Related works

After being introduced in [1], several new variants of CH have been published. These include CH with ephemeral trapdoor [2] and identity-based [12–15], policy-based, tag-based [16], and RPCH functions.

RPCH is a combination of an RABE and CHET. Thus, designing an efficient RPCH using lattice requires designing an efficient lattice-based RABE and CHET. For the construction of CHET, two frameworks were introduced by Derler et al. [2]. The first one is based on zero-knowledge proof and a commitment scheme. The second technique consists of transforming a secure and unique CH into a CHET where the hash of a message $\boldsymbol{m}$ will be a pair $\boldsymbol{h} = (\boldsymbol{h}_1, \boldsymbol{h}_2)$. $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ correspond to the hash of $\boldsymbol{m}$ computed using the public key and an ephemeral public key, respectively. The CHET designed in this paper is based on the second technique using Cash et al.'s CH [11]. In Cash et al.'s scheme, the public key is a pair of matrices $(\mathbf{A}_1||\mathbf{A}_2)$, and the hash of a message $\boldsymbol{m}$ is given by $\boldsymbol{h} = \mathbf{A}_1\boldsymbol{m} + \mathbf{A}_2\boldsymbol{r}$, where $\boldsymbol{r}$ is a random vector chosen from a Gaussian distribution. Using a CHET designed from Cash et al.'s CH to build a RPCH would require a slight modification. The ephemeral trapdoor has to be encrypted using an ABE scheme. Therefore, the ephemeral trapdoor needs to be a binary string instead of a matrix.

For the design of the proposed RPCH, an adaptive secure RABE scheme is required. However, most of the revocable attribute-based schemes using lattice in the literature are selectively secure. For this purpose, we introduce a new adaptive secure RABE scheme on top of Tsabary's attribute-based encryption [17], which is one of the efficient and adaptive secure ABEs. The Tsabary scheme is a ciphertext-policy ABE scheme that uses as policies the conjunctive normal form policies.

1.2 Motivation

Our motivation is mainly two-fold. First, there exist lattice-based hard problems that are considered resistant to attacks using future quantum computers, but to our knowledge, there is no work in the literature on the design of adaptive indistinguishable RABE based on such hard problems. Although there is a recent article on adaptive indistinguishable RABE introduced by Xiong et al. [18], the security of their scheme relies on the decisional linear (DLIN) assumption, which is not quantum secure either. Second, and with regard to RPCH, it appears that Xu et al.'s work [10] is the only one published in the literature so far and the security of their scheme relies on the DLIN assumption, and hence, it is not quantum secure. With the continued progress in the development of quantum computers, it is therefore of interest to design RPCH with post-quantum security assumptions.

1.3 Our contributions

In this paper, we propose a slight modification of Micciancio and Peikert's trapdoor generation algorithm [19, Algorithm 1] using a random oracle (RO) to reduce the size of trapdoors. We use the modified trapdoor generation algorithm to introduce a lattice-based CHET. Our scheme is designed using a modified version of Cash et al.'s CH [11] and Derler et al.'s framework [9]. We show that the security of the proposed lattice-based CHET relies on short integer solution (SIS) and inhomogeneous short integer solution (ISIS) problems. We design an adaptive indistinguishable RABE from lattice assumptions. We combine our proposed lattice-based

RABE and CHET schemes to design a lattice-based RPCH scheme from Xu et al.'s construction [10]. Finally, we show that our schemes are secure and compare our RABE with relevant lattice-based schemes proposed recently.

1.4 Organization of this article

In this article, we recall the necessary prerequisites in Section 2. In Section 3, we introduce a modified version of Micciancio and Peikert's trapdoor generator algorithm, construct a lattice-based CHET scheme, and give a security analysis of the proposed CHET. Section 4 contains the description of our RABE scheme and its security analysis. In Section 5, we describe our RPCH scheme using lattice and provide its security analysis. Finally, we conclude in Section 6.

2 Preliminaries

In this paper, we consider row vectors and denote vectors by bold lowercase and matrices by bold uppercase letters. In addition, we consider the following definition of norms: $||\boldsymbol{a}|| = \sqrt{\sum_{i=1}^{m} a_i^2}$, $||\boldsymbol{a}||_1 = \sum_{i=1}^{m} |a_i|$, $||\boldsymbol{a}||_\infty = \max_i |a_i|$, $||\mathbf{A}||_1 = \max_j \sum_{i=1}^{m} |a_{ij}|$ and $||\mathbf{A}||_\infty = \max_i \sum_{j=1}^{m} |a_{ij}|$.

2.1 Lattices

Let m be a nonzero positive integer $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m\}$ be a set of m linearly independent vectors of $\mathbb{R}^m$. A (full-rank) m-dimensional lattice generated by $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m\}$ is the set denoted by $\Lambda(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$ and defined by $\Lambda(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m) = \{\sum_{i=1}^{m} x_i \boldsymbol{b}_i : x_i \in \mathbb{Z}\}$. Equivalently, if $\mathbf{B}$ is a matrix where its rows are vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m$, the lattice $\Lambda(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$ can be denoted by $\Lambda(\mathcal{B})$ and described by $\Lambda(\mathcal{B}) = \{\boldsymbol{x}\mathbf{B} : \boldsymbol{x} \in \mathbb{Z}^m\}$.

For a prime number q, a lattice Λ is called a q-ary lattice if $q\mathbb{Z}^m \subseteq \Lambda \subseteq \mathbb{Z}^m$. With this condition, it is easy to verify that for all $\boldsymbol{y} \in \mathbb{Z}^m$, $\boldsymbol{y}$ is an element of a q-ary lattice Λ if and only if $\boldsymbol{y} \bmod q \in \Lambda$.

Let $0 < n \leq m$ and $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ be a full rank matrix. We denote by $\Lambda_q(\mathbf{A})$ the set defined by

$$\Lambda_q(\mathbf{A}) = \{\boldsymbol{y} \in \mathbb{Z}^m : \boldsymbol{y} \bmod q = \boldsymbol{x}\mathbf{A} \bmod q \text{ for some } \boldsymbol{x} \in \mathbb{Z}^n\}.$$

Similarly, $\Lambda_q^\perp(\mathbf{A})$ is defined by

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

$\Lambda_q(\mathbf{A})$ and $\Lambda_q^\perp(\mathbf{A})$ are q-ary lattices. In addition, we can verify that $\Lambda_q(\mathbf{A})$ satisfies $\Lambda_q(\mathbf{A}) = \mathbb{Z}^n\mathbf{A} + q\mathbb{Z}^m$. For all $\boldsymbol{u} \in \mathbb{Z}_q^n$, we denote by $\Lambda_q^{\boldsymbol{u}}(\mathbf{A})$ the set

$$\Lambda_q^{\boldsymbol{u}}(\mathbf{A}) = \{\boldsymbol{y} \in \mathbb{Z}^m : \mathbf{A}\boldsymbol{y}^T = \boldsymbol{u} \bmod q\}.$$

2.2 Discrete Gaussian, bounded distributions, lattice trapdoors, and hard problems

2.2.1 Discrete Gaussian

For a positive real number $\sigma > 0$, the Gaussian function with parameter σ and center $\boldsymbol{c} \in \mathbb{R}^n$ is defined by

$$\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x}) = \exp\left(-\frac{||\boldsymbol{x}-\boldsymbol{c}||^2}{\sigma^2}\pi\right), \quad \text{for all } \boldsymbol{x} \in \mathbb{R}^n.$$

When $\boldsymbol{c}$ is the origin, the corresponding Gaussian function is said to be centered. Later, in this article, we simply say the Gaussian function to refer to the centered Gaussian function and denote it by ρ_σ. For an n-dimensional lattice Λ, the discrete Gaussian distribution on Λ is the function defined by

$$\mathcal{D}_{\Lambda,\sigma}(\boldsymbol{x}) = \frac{\rho_\sigma(\boldsymbol{x})}{\rho_\sigma(\Lambda)}, \quad \text{for all } \boldsymbol{x} \in \Lambda,$$

where $\rho_\sigma(\Lambda) = \sum_{\boldsymbol{x}\in\Lambda}\rho_\sigma(\boldsymbol{x})$. We denote the discrete Gaussian distribution supported over $\mathbb{Z}$ (resp. $\mathbb{Z}_q$) with parameter σ by $\mathcal{D}_\sigma$ (resp. $\mathcal{D}_{q,\sigma}$). For a given lattice Λ, we denote this distribution by $\mathcal{D}_{\Lambda,\sigma}$. We now state the following lemma:

Lemma 1. [20] *Let n, m, and q be the positive integers such that $m \geq 2n\log(q)$, where q is a prime number. Let σ be any positive real number such that $\sigma \geq \sqrt{n + \log(m)}$. Then, for $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ and $\boldsymbol{e} \xleftarrow{\$} \mathcal{D}_\sigma^m$, the distribution of $\boldsymbol{u} = \mathbf{A}\boldsymbol{e}^T \bmod q$ is $2^{-\Omega(n)}$-statistically close to uniform over $\mathbb{Z}_q^n$. Furthermore, for a fixed $\boldsymbol{u} \in \mathbb{Z}_q^n$, the conditional distribution of $\boldsymbol{e} \xleftarrow{\$} \mathcal{D}_\sigma^m$, given $\mathbf{A}\boldsymbol{e}^T = \boldsymbol{u} \bmod q$ for a uniformly random $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, is $\mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(\mathbf{A}),\sigma}$.*

2.2.2 Bounded distribution [17]

Let χ be a distribution supported over $\mathbb{Z}$. If there exists $B \in \mathbb{R}_+$ and $0 < \varepsilon < 1$ such that $\Pr_{x\xleftarrow{\$}\chi}[|x| > B] < \varepsilon$, χ is said to be (B, ε)-bounded. When ε is a negligible function, for brevity, χ is the said B-bounded distribution instead of (B, ε)-bounded. χ is said to be (B, ε)-swallowing if for all $y \in [-B, B] \cap \mathbb{Z}$, the statistical distance between χ and $y + \chi$ is equal to ε. A second distribution $\tilde{\chi}$ supported over $\mathbb{Z}$ is said to be (χ, ε)-swallowing if the statistical distance between χ and $\chi + \tilde{\chi}$ is equal to ε.

2.2.3 Lattice trapdoor

Let $n, q \in \mathbb{Z}$, $\boldsymbol{g} = (1, 2, 4, \ldots, 2^{\lceil \log(q)\rceil - 1})$ and $m_0 = n\lceil \log(q)\rceil$. The *gadget* matrix $\mathbf{G}$ is defined as the diagonal matrix given by $\mathbf{G} = \mathbf{I}_n \otimes \boldsymbol{g} \in \mathbb{Z}_q^{n\times m_0}$, where $\otimes$ is the Kronecker product [17]. For all $t \in \mathbb{Z}$, let $\mathbf{G}^{-1} : \mathbb{Z}_q^{n\times t} \longrightarrow \{0,1\}^{m_0\times t}$ be the function that converts each entry $a_{ij} \in \mathbb{Z}_q$ of a matrix in a binary column vector of size $\lceil \log(q)\rceil$ corresponding to the binary representation of a_{ij}. It holds that for all $\mathbf{A} \in \mathbb{Z}_q^{n\times m_0}$, we have $\mathbf{G}\mathbf{G}^{-1}(\mathbf{A}) = \mathbf{A}$ [17].

Let $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ be a matrix. For all $\boldsymbol{v} \in \mathbb{Z}_q^n$ (resp. $\mathbf{V} \in \mathbb{Z}_q^{n\times m}$), let $\mathbf{A}_\sigma^{-1}(\boldsymbol{v})$ (resp. $\mathbf{A}_\sigma^{-1}(\mathbf{V})$) be an output distribution of $\mathcal{D}_\sigma^{m_0}$ (resp. of $\mathcal{D}_\sigma^{m_0\times m}$).

A σ-trapdoor for a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m_0}$ is a trapdoor that enables to sample from the distribution $\mathbf{A}_\sigma^{-1}(\boldsymbol{v})$ (resp. $\mathbf{A}_\sigma^{-1}(\mathbf{V})$) in time poly$(n, m_0, \log(q))$ (resp. poly$(n, m_0, m, \log(q))$) for all $\boldsymbol{v} \in \mathbb{Z}_q^n$ (resp. $\mathbf{V} \in \mathbb{Z}_q^{n\times m}$). We denote a σ-trapdoor of matrix $\mathbf{A}$ by $\mathbf{A}_\sigma^{-1}$. In the following proposition, we recall some properties of trapdoors [11,17,19,21–23].

Proposition 1.

(1) *There exist an efficiently computable value* $m_0 = O(n \log(q))$ *and an efficient procedure* GenTrap *such that for all* $m \geq m_0$, GenTrap *outputs* $(\mathbf{A}, \mathbf{A}_\sigma^{-1}) = \mathsf{GenTrap}(1^n, m, q)$, *where* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ *is* 2^{-n}*-uniform and* $\sigma = O(\sqrt{n \log(q) \log(n)})$.

(2) *Let* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ *be a matrix with a trapdoor* $\mathbf{A}_\tau^{-1}$. *Then, it is efficient to generate a trapdoor* $[\mathbf{A}||\mathbf{B}]_{\tau'}^{-1}$, *for all* $\mathbf{B} \in \mathbb{Z}_q^{n \times m'}$, *any* $m' \in \mathbb{N}$, *and* $\tau' \geq \tau$.

(3) *Let* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ *and* $\mathbf{R} \in \mathbb{Z}_q^{m \times m_0}$ *with* $m_0 = n\lceil \log(q) \rceil$. *Then, it is efficient to compute* $[\mathbf{A}||\mathbf{AR} + \mathbf{G}]_\tau^{-1}$ for $\tau = O(\sqrt{m_0 m}\, ||\mathbf{R}||_\infty)$.

Note that if $\mathbf{A}$ is chosen uniformly and $\mathbf{R}$ uniformly in $\{-1, 0, 1\}^{m \times m_0}$, then part 3 of Proposition 1 is a special case of the leftover hash lemma [17].

2.2.4 Hard lattice problems

Now, we briefly recall some hard lattice problems that we use as security assumptions in our schemes. For more details, the reader is referred to [21,24–29].

- Short integer solution ($\mathsf{SIS}_{q,n,m,\gamma}$): Given a random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, find a vector $\boldsymbol{x} \in \mathbb{Z}^m \backslash \{\mathbf{0}\}$ such that $\mathbf{A}\boldsymbol{x}^T = \mathbf{0}$ and $||\boldsymbol{x}|| \leq \gamma$.
- Inhomogeneous short integer solution ($\mathsf{ISIS}_{q,n,m,\gamma}$): Given a random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a vector $\boldsymbol{y} \in \mathbf{Z}_q^n$, find a vector $\boldsymbol{x} \in \mathbb{Z}^m \backslash \{\mathbf{0}\}$ such that $\mathbf{A}\boldsymbol{x}^T = \boldsymbol{y}$ and $||\boldsymbol{x}|| \leq \gamma$
- Decisional learning with errors ($\mathsf{DLWE}_{q,n,m,\chi}$) [28]: Let n, m, and q be positive integers such that q is prime. Let χ be a noise distribution over $\mathbb{Z}$. Distinguish the following two distributions:

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

where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_q^m$, and $\boldsymbol{e} \xleftarrow{\$} \chi^m$ are independently sampled.

We are interested in the case where $q < 2^n$; therefore, we have the following corollary:

Corollary 1. [19,28] *For all* $\varepsilon > 0$, *there exist functions* $q = q(n) \leq 2^n$, *and* $\chi = \chi(n)$ *such that* χ *is B-bounded for some* $B = B(n)$, $q/B \geq 2^{n^\varepsilon}$, *and* $\mathsf{DLWE}_{q,n,m,\chi}$ *is at least as hard as the classical hardness of the gap shortest vector problem* (GapSVP_γ) *and the quantum hardness of the shortest independent vector problem* (SIVP_γ) *for* $\gamma = 2^{\Omega(n^\varepsilon)}$ *and all* $m = \mathrm{poly}(n)$.

2.3 Pseudo-random function (PRF) and conjunctive normal form policy

2.3.1 PRF

PRF defined over a key space $\mathcal{K}$, a domain $\mathcal{X}$, and a range $\mathcal{Y}$, is a function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ that can be computed by a deterministic polynomial time algorithm. On input (K, x), the algorithm outputs $F(\mathsf{K}, x) = y \in \mathcal{Y}$. Moreover, for a given key K, the function $F(\mathsf{K}, .)$ is efficiently evaluated at all points $x \in \mathcal{X}$. Thus, a PRF is defined by a set of two algorithms $F = (\mathsf{F.\,Setup}, \mathsf{F.\,Eval})$ such that [17,30]:

- $(\mathsf{PP}, \mathsf{K}) \leftarrow \mathsf{F.\,Setup}(1^\lambda)$ is the setup algorithm that takes as input a security parameter λ and outputs a key $\mathsf{K} \in \{0,1\}^\lambda$ and public parameters PP.
- $\boldsymbol{y} \leftarrow \mathsf{F.\,Eval}(\mathsf{K}, \boldsymbol{x})$ is the evaluation algorithm that for a given secret key K, takes as input a binary string $\boldsymbol{x} \in \{0,1\}^\ell$ and outputs $\boldsymbol{y} \in \{0,1\}^n$, where $n = n(\lambda)$ and $\ell = \ell(\lambda)$.

It is important to note that there are different types of PRFs, however, in this paper, we are interested in the constrained pseudo-random functions (cPRF) introduced by Boneh and Waters [31]. Indeed, for a given $\mathcal{F}$, a class of functions f defined from $\{0,1\}^{\ell}$ onto $\{0,1\}$, cPRF with policies in $\mathcal{F}$, is a tuple of algorithms F = (F. Setup, F. Eval, F. Constrain, F. ConstrainEval) such that [32]:

- (PP, msk) ← F. Setup(1^{λ}) takes as input the security parameter λ and outputs PP and a master secret key msk.
- $\boldsymbol{r}_{\boldsymbol{x}}$ ← F. Eval(msk, $\boldsymbol{x}$) is a deterministic algorithm that outputs a bit-string $\boldsymbol{r}_{\boldsymbol{x}}$ of length $n = n(\lambda)$ by taking as input a master secret key msk and a bit-string $\boldsymbol{x}$ of length ℓ.
- csk_f ← F. Constrain(msk, f) takes as input a master secret key msk and a function $f \in \mathcal{F}$ and returns a constrained key csk_f.
- $\boldsymbol{r}'_{\boldsymbol{x}}$ ← F. ConstrainEval(csk_f, $\boldsymbol{x}$) is a deterministic algorithm that takes as input a constrained key csk_f and a bit-string of length ℓ. It returns a bit-string $\boldsymbol{r}'_{\boldsymbol{x}}$ of length n.

A cPRF is said to be correct when for all $\boldsymbol{x} \in \{0,1\}^{\ell}$ and $f \in \mathcal{F}$, if $f(\boldsymbol{x}) = 1$, we have F. Eval(msk, $\boldsymbol{x}$) = F. ConstrainEval(csk_f, $\boldsymbol{x}$) where (PP, msk) = F. Setup(1^{λ}) and csk_f =F. Constrain(msk, f). However, there are two notions of pseudo-randomness for cPRFs namely, adaptive pseudo-randomness and selective pseudo-randomness. A cPRF is said to be adaptive pseudo-random if the probability of a PPT adversary $\mathcal{A}$, to win the adaptive pseudo-randomness game is negligible.

In this paper, for the design of our RABE, we are interested in a particular type of cPRF introduced by Tsabary called conforming cPRF. These functions, in addition to being pseudo-random, satisfy the following two properties [17]:

- *Gradual evaluation:* for all $f \in \mathcal{F}$ and $\boldsymbol{x} \in \{0,1\}^{\ell}$, if $f(\boldsymbol{x}) = 1$, it holds that the circuit descriptions of the algorithms F. Eval($\cdot$, $\boldsymbol{x}$) and F. ConstrainEval(F. Constrain($\cdot$, f), $\boldsymbol{x}$) are identical.
- *Key simulation:* it means that there exists an additional public algorithm F. KeySim(f) ← σ_f that allows a simulation of constrained keys. Moreover, simulated keys should be indistinguishable from real constrained keys to any distinguisher with no access to evaluations on points $\boldsymbol{x}$, where $f(\boldsymbol{x}) = 1$.

The adaptive pseudo-random game for cPRFs [17,32]:

cPRF. PseudoRandGame

1. (PP, F.msk) ← Setup(1^{λ}),
2. $\boldsymbol{x}^*$ ← $\mathcal{A}^{O_{\mathsf{Eval,Constrain}}}$(PP),
3. $\boldsymbol{r}_b^*$ ← Eval(F.msk, $\boldsymbol{x}^*$),
4. $\tilde{b}$ ← $\mathcal{A}^{O_{\mathsf{Eval,Constrain}}}(\boldsymbol{r}_b^*)$.

An adversary $\mathcal{A}$ wins in the adaptive pseudo-random game when the guessed value $\tilde{b}$ is equal to b. The advantage of an adversary $\mathcal{A}$ in the adaptive pseudo-random game is given by $\mathsf{Adv}^{\mathsf{PseudoRand}}_{\mathcal{A},\mathsf{cPRF}} = |\Pr[\tilde{b} = b] - 1/2|$. Thus, a cPRF is adaptive pseudo-random if the advantage of any PPT adversary $\mathcal{A}$ in the adaptive pseudo-random game is negligible in λ, i.e.,

$$\mathsf{Adv}^{\mathsf{PseudoRand}}_{\mathcal{A},\mathsf{cPRF}} \leq \varepsilon(\lambda).$$

2.3.2 Conjunctive normal form policy [17]

Let ℓ and t be nonzero positive integers such that $t \leq \ell$. A t-conjunctive normal form (t-CNF) policy $f : \{0,1\}^{\ell} \longrightarrow \{0,1\}$ is a set of clauses $f = \{(T_i, f_i)\}_i$ such that for all $\boldsymbol{x} \in \{0,1\}^{\ell}$, $f(\boldsymbol{x}) = \wedge_i f_i(\boldsymbol{x}_{T_i})$, and for all i, $T_i \subseteq \{1, \ldots, \ell\}, |T_i| = t$, $f_i : \{0,1\}^t \longrightarrow \{0,1\}$ and $\boldsymbol{x}_{T_i} = (x_j)_{j \in T_i}$ is a vector formed by components of $\boldsymbol{x}$ indexed by T_i. A class of functions $\mathcal{F}$ is said to be t-CNF when it consists of only t-CNF policies for some fixed $\ell \in \mathbb{N}$ and $t \leq \ell$.

2.4 Revocable attribute-based encryption

ABE scheme is a generalization of cryptographic primitives such as identity-based encryption scheme that allows us to target the recipients of a message according to access policies. It was introduced by Sahai and Waters [33] and developed later by Goyal et al. [34]. There are two categories of ABE schemes: ciphertext policy (CP − ABE) and key policy (KP − ABE) ABE schemes. In CP − ABE schemes, the ciphertexts are associated with access policies, i.e., senders can decide who can decrypt the sent ciphertexts. On the other hand, in KP − ABE, secret keys are associated with access structures that specify which ciphertexts the user will be able to decrypt. In this article, we are interested in the ciphertext policy revocable ABE scheme (CP − RABE), which is a generalization of the ciphertext policy ABE scheme where, it is possible to revoke a malicious secret holder. For brevity, when the context is clear, we sometimes denote CP − RABE simply as RABE.

2.4.1 Definition

Let $\mathcal{F}$ be a class of functions defined from $\{0,1\}^{\ell}$ onto $\{0,1\}$. Let $\mathcal{T}$ be a time space, $\mathcal{M}$ a message space, and $\mathcal{ID}$ an identity space. A CP − RABE scheme associated with $\mathcal{T}$, $\mathcal{M}$ for policies $f \in \mathcal{F}$ is a tuple of algorithms $(\mathsf{Setup}_{\mathsf{RABE}}, \mathsf{SKGen}_{\mathsf{RABE}}, \mathsf{DKGen}_{\mathsf{RABE}}, \mathsf{Enc}_{\mathsf{RABE}}, \mathsf{Dec}_{\mathsf{RABE}}, \mathsf{Rev}_{\mathsf{RABE}}, \mathsf{KUpd}_{\mathsf{RABE}})$ defined as follows:

- $(\mathsf{msk}_{\mathsf{RABE}}, \mathsf{mpk}_{\mathsf{RABE}}, \mathsf{st}, \mathsf{RL}) \leftarrow \mathsf{Setup}_{\mathsf{RABE}}(1^{\lambda})$ takes as input a security parameter λ. It returns a master secret key $\mathsf{msk}_{\mathsf{RABE}}$, a master public key $\mathsf{mpk}_{\mathsf{RABE}}$, a state st, and a revocation list RL. We assume that the master public key contains all information about the plaintext message space $\mathcal{M}$, time period space $\mathcal{T} \in \{1, \ldots, \mathsf{T}_{\max}\}$, and identity space $\mathcal{ID}$, where $\mathsf{T}_{\max}$ is polynomial in λ. We also assume that the revocation list RL is initialized to $\mathsf{RL} = \varnothing$.
- $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{SKGen}_{\mathsf{RABE}}(\mathsf{msk}_{\mathsf{RABE}}, \mathsf{st}, \mathsf{id}, \boldsymbol{x})$ takes as input a master secret key $\mathsf{msk}_{\mathsf{RABE}}$, a state st, an identity id with its corresponding attribute $\boldsymbol{x} \in \{0,1\}^{\ell}$, and outputs a secret key $\mathsf{sk}_{\mathsf{id}}$.
- $\mathsf{ku}_{\mathsf{T}} \leftarrow \mathsf{KUpd}_{\mathsf{RABE}}(\mathsf{msk}_{\mathsf{RABE}}, \mathsf{T}, \mathsf{RL})$ takes as input a master secret key $\mathsf{msk}_{\mathsf{RABE}}$, a time period T, and a revocation list RL. It outputs an updated key material ku_{T}.
- $\mathsf{dk}_{\mathsf{T},\mathsf{id}} \leftarrow \mathsf{DKGen}_{\mathsf{RABE}}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ku}_{\mathsf{T}})$ takes as input a secret key $\mathsf{sk}_{\mathsf{id}}$ and a key update material ku_{T} corresponding to a time period T. It outputs a decryption key $\mathsf{dk}_{\mathsf{id},\mathsf{T}}$.
- $\mathsf{ct} \leftarrow \mathsf{Enc}_{\mathsf{RABE}}(\mathsf{mpk}_{\mathsf{RABE}}, \boldsymbol{m}, f, \mathsf{T})$ outputs a ciphertext ct. It takes as input a master public key $\mathsf{mpk}_{\mathsf{RABE}}$, a time period $\mathsf{T} \in \mathcal{T}$, a plaintext message $\boldsymbol{m} \in \mathcal{M}$, and an access policy $f \in \mathcal{F}$.
- $\boldsymbol{m} \leftarrow \mathsf{Dec}_{\mathsf{RABE}}(\mathsf{dk}_{\mathsf{id},\mathsf{T}}, \mathsf{ct})$ takes as input a decryption key $\mathsf{dk}_{\mathsf{id},\mathsf{T}}$ and a ciphertext ct and outputs a plaintext $\boldsymbol{m} \in \mathcal{M}$.
- $\mathsf{RL} \leftarrow \mathsf{Rev}_{\mathsf{RABE}}(\mathsf{id}, \mathsf{RL}, \mathsf{T})$ outputs an updated revocation list RL. It takes as input an identity id, a time period T, and a revocation list RL.

2.4.2 Correctness

The correctness requires that the decryption of all ciphertext $\mathsf{ct} = \mathsf{Enc}_{\mathsf{RABE}}(\mathsf{mpk}_{\mathsf{RABE}}, \boldsymbol{m}, f, \mathsf{T})$ by an honest user with non-revoked identity id always gives the message $\boldsymbol{m}$ with high probability if the user attribute $\boldsymbol{x}$ satisfies the ciphertext-policy f. Otherwise, an RABE is correct when the probability of a failed decryption of an honest computed ciphertext by an honest and non-revoked user satisfying the access policy is negligible. Indeed, for a chosen security parameter $\lambda \in \mathbb{N}$ and a fixed pair of master public/secret keys $(\mathsf{mpk}_{\mathsf{RABE}}, \mathsf{msk}_{\mathsf{RABE}})$, the following inequality is satisfied:

$$\Pr\left[\mathsf{Dec}_{\mathsf{RABE}}(\mathsf{dk}_{\mathsf{id},\mathsf{T}}, \mathsf{ct}) \neq \boldsymbol{m} \;\middle|\; \begin{array}{c} \mathsf{ct} = \mathsf{Enc}_{\mathsf{RABE}}(\mathsf{mpk}_{\mathsf{RABE}}, \boldsymbol{m}, f, \mathsf{T}) \\ \mathsf{sk}_{\mathsf{id}} = \mathsf{SKGen}_{\mathsf{RABE}}(\mathsf{msk}_{\mathsf{RABE}}, \mathsf{st}, \mathsf{id}, \boldsymbol{x}) \\ \mathsf{id} \notin \mathsf{RL} \text{ and } f(\boldsymbol{x}) = 1 \\ \mathsf{ku}_{\mathsf{T}} = \mathsf{KUpd}_{\mathsf{RABE}}(\mathsf{msk}_{\mathsf{RABE}}, \mathsf{T}, \mathsf{RL}) \\ \mathsf{dk}_{\mathsf{id},\mathsf{T}} = \mathsf{DKGen}_{\mathsf{RABE}}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ku}_{\mathsf{T}}) \end{array}\right] \leq \varepsilon(\lambda),$$

where ε is a negligible function in λ.

2.4.3 Security model

Note that, as in ABE schemes, there are two kinds of IND − CPA security notions in RABE: the selective and the adaptive IND − CPA security. The main difference between the two is that in the selective IND − CPA game, the adversary must declare to the challenger the access policy and a revocation list RL on a time period T before receiving the secret key and public parameters from the challenger. However, in the adaptive IND − CPA game, it should declare the policy after receiving the secret key and public parameters from the challenger. Hence, different phases in the adaptive IND − CPA game of RABE can be described as follows:
Setup:
During the setup phase of the RABE adaptive IND-CPA game, the challenger C computes $(\mathsf{msk}_{\mathsf{RABE}}, \mathsf{mpk}_{\mathsf{RABE}}, \mathsf{st}, \mathsf{RL}) = \mathsf{Setup}(1^{\lambda})$ and prepares a list SKList in which all tuples $(\mathsf{id}, \boldsymbol{x}, \mathsf{sk}_{\mathsf{id}})$ generated during the game will be stored. Indeed, when a new secret key $\mathsf{sk}_{\mathsf{id}}$ is generated from a pair $(\mathsf{id}, \boldsymbol{x})$ of identity id and attribute $\boldsymbol{x}$ due to the execution of SKGen, the challenger C will store $(\mathsf{id}, \boldsymbol{x}, \mathsf{sk}_{\mathsf{id}})$ in SKList. Then, the challenger computes the updated key material $\mathsf{ku}_{\mathsf{T}_{cu}} = \mathsf{KUpd}(\mathsf{msk}_{\mathsf{RABE}}, \mathsf{RL}_{\mathsf{T}_{cu}}, \mathsf{T}_{cu} = 1)$ for initial time period $\mathsf{T}_{cu} = 1$, where T_{cu} corresponds to *current time period*. Finally, the challenger C sends $\mathsf{mpk}_{\mathsf{RABE}}$ and $\mathsf{ku}_{\mathsf{T}_{cu}}$ to the adversary $\mathcal{A}$.

Queries:
During the query phases, the adversary $\mathcal{A}$ may adaptively (possibly many times) make the following queries:

- *Secret key generation query:* Until the challenge query, upon a query $(\mathsf{id}, \boldsymbol{x})$ from the adversary $\mathcal{A}$, the challenger C first checks if $(\mathsf{id}, *) \notin \mathsf{RL}_{\mathsf{T}_{cu}}$ and $(\mathsf{id}, *, *) \notin \mathsf{SKList}_{\mathsf{T}_{cu}}$. If the condition is not satisfied, the challenger returns $\perp$. Otherwise, it computes $(\mathsf{sk}_{\mathsf{id}}, \mathsf{st}) = \mathsf{SKGen}(\mathsf{msk}_{\mathsf{RABE}}, \mathsf{st}, \mathsf{id}, \boldsymbol{x})$ and then stores $(\mathsf{id}, \boldsymbol{x}, \mathsf{sk}_{\mathsf{id}})$ in $\mathsf{SKList}_{\mathsf{T}_{cu}}$. Finally, it returns $\mathsf{sk}_{\mathsf{id}}$ to the adversary $\mathcal{A}$.
- *Revoke and key update query:* Until the challenge query, upon a query RL* from the adversary $\mathcal{A}$, where RL* is a set of identities that are going to be revoked in the next time period, the challenger C checks if the following condition is satisfied:

$$\mathsf{RL}_{\mathsf{T}_{cu}} \subseteq \mathsf{RL}^*.$$

  This is for ensuring that identities that have already been revoked will remain revoked in the next period. After the challenge query, C also checks if $\mathsf{T}_{cu} = \mathsf{T}^* - 1$ and all $\mathsf{id} \in \mathcal{ID}$ such that $(\mathsf{id}, *, *) \in \mathsf{SKList}_{\mathsf{T}_{cu}}$ are revoked, i.e., $(\mathsf{id}, *) \in \mathsf{RL}^*$. If the conditions are not satisfied, the challenger returns $\perp$ to the adversary $\mathcal{A}$. Otherwise, the challenger computes $\mathsf{T}_{cu} = \mathsf{T}_{cu} + 1$ and sets $\mathsf{RL}_{\mathsf{T}_{cu}} = \mathsf{RL}^*$. Then, it computes $\mathsf{ku}_{\mathsf{T}_{cu}} = \mathsf{KUpd}(\mathsf{msk}_{\mathsf{RABE}}, \mathsf{RL}_{\mathsf{T}_{cu}}, \mathsf{T}_{cu})$. Finally, it returns $\mathsf{ku}_{\mathsf{T}_{cu}}$ to the adversary $\mathcal{A}$.

The adversary $\mathcal{A}$ is allowed to query only during increased time $\mathsf{T} > \mathsf{T}_{cu}$.

Challenge:
For the challenge query, the adversary $\mathcal{A}$ is allowed to make the query only once. Upon a query $(\boldsymbol{m}^*, \boldsymbol{x}^* \in \{0,1\}^{\ell}, f^*, \mathsf{T}^*)$, the challenger verifies that $\mathsf{T}^* \leq \mathsf{T}_{cu}$ and for all $\mathsf{id} \in \mathcal{ID}$ such that $(\mathsf{id}, *, *) \in \mathsf{SKList}^*$, $(\mathsf{id}, *) \in \mathsf{RL}^*$. Then, the challenger picks $b \in \{0,1\}$. If $b = 0$, the challenger computes $\mathsf{ct}_b^* = \mathsf{Enc}(\mathsf{mpk}_{\mathsf{RABE}}, \boldsymbol{m}^*, f^*, \mathsf{T}^*)$; otherwise, it samples ct_b^* from the ciphertext space uniformly at random. Finally, it returns ct* to the adversary $\mathcal{A}$.

Guess:
During this phase, the adversary guesses $\tilde{b} \in \{0,1\}$ and terminates the game.

The aforementioned description of the RABE adaptive IND − CPA game can be summed up in four steps as follows.

RABE.IND − aCPAGame:

1. $(\mathsf{msk}_{\mathsf{RABE}}, \mathsf{mpk}_{\mathsf{RABE}}, \mathsf{st}, \mathsf{RL}) \xleftarrow{\$} \mathsf{Setup}(1^{\lambda})$,
2. $(\boldsymbol{m}^*, \boldsymbol{x}^* \in \{0,1\}^{\ell}, f^*, \mathsf{T}^*) \xleftarrow{\$} \mathcal{A}^{O}(\mathsf{mpk}_{\mathsf{RABE}})$,

3. $\mathsf{ct}_b^* \leftarrow \mathsf{Enc}(\mathsf{mpk}_{\mathsf{RABE}}, \boldsymbol{m}^*, f^*, \mathsf{T}^*)$,
4. $\tilde{b} \leftarrow \mathcal{A}^{O}(\mathsf{ct}_b^*)$,

where $\mathsf{IND-aCPA}$ denotes the adaptive $\mathsf{IND-CPA}$. At the end of the $\mathsf{IND-aCPA}$ game, the adversary $\mathcal{A}$ is successful with the attack when $\tilde{b} = b$ and $(f^*(\boldsymbol{x}^*) = 0 \text{ or } \mathsf{id} \in \mathsf{RL}_{\mathsf{T}^*})$. Moreover, its advantage in $\mathsf{RABE.IND-aCPAGame}$ is given by $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{RABE}}(\lambda) = |\mathsf{Pr}[\tilde{b} = b] - 1/2|$. Therefore, an RABE is said to be adaptively indistinguishable if for all PPT adversary $\mathcal{A}$, the following inequality holds:

$$|\mathsf{Pr}[\tilde{b} = b] - 1/2| \leq \varepsilon(\lambda),$$

where ε is a negligible function in λ.

2.5 CHET

In this section, we briefly recall the formal definition and security model of CHET and highlight some of the differences among CHET, PCH, and RPCH. For more details CH schemes, the reader is referred to [2,9,10,35,36].

2.5.1 Definition

A CH (CHET) is a set of five algorithms ($\mathsf{Setup}_{\mathsf{CHET}}, \mathsf{KGen}_{\mathsf{CHET}}, \mathsf{Hash}_{\mathsf{CHET}}, \mathsf{Verif}_{\mathsf{CHET}}$, and $\mathsf{Adapt}_{\mathsf{CHET}}$) described as follows:

- $\mathsf{PP}_{\mathsf{CHET}} \leftarrow \mathsf{Setup}_{\mathsf{CHET}}(1^\lambda)$ generates public parameters $\mathsf{PP}_{\mathsf{CH}}$ of the scheme; it takes as input the security parameter λ.
- $(\mathsf{sk}_{\mathsf{CHET}}, \mathsf{pk}_{\mathsf{CHET}}) \leftarrow \mathsf{KGen}_{\mathsf{CHET}}(\mathsf{PP}_{\mathsf{CHET}})$ takes as input the public parameters $\mathsf{PP}_{\mathsf{CHET}}$ and outputs a pair of secret and public keys $(\mathsf{sk}_{\mathsf{CHET}}, \mathsf{pk}_{\mathsf{CHET}})$.
- $(\boldsymbol{h}, \boldsymbol{r}, \mathsf{etd}) \leftarrow \mathsf{Hash}_{\mathsf{CHET}}(\mathsf{pk}_{\mathsf{CHET}}, \boldsymbol{m})$ takes as input the message $\boldsymbol{m}$ to hash and the public key $\mathsf{pk}_{\mathsf{CHET}}$; it outputs a hash $\boldsymbol{h}$, a random value $\boldsymbol{r}$, and an ephemeral trapdoor etd.
- $b \in \{0,1\} \leftarrow \mathsf{Verif}_{\mathsf{CHET}}(\mathsf{pk}_{\mathsf{CHET}}, \boldsymbol{h}, \boldsymbol{r})$ is a deterministic algorithm that outputs $b \in \{0,1\}$; it takes as input the public key $\mathsf{pk}_{\mathsf{CHET}}$, a message $\boldsymbol{m}$, a hash $\boldsymbol{h}$ and a random value $\boldsymbol{r}$.
- $\boldsymbol{r}' \leftarrow \mathsf{Adapt}_{\mathsf{CHET}}(\mathsf{sk}_{\mathsf{CHET}}, \mathsf{etd}, \boldsymbol{m}, \boldsymbol{m}', \boldsymbol{h}, \boldsymbol{r})$ takes as input the secret key $\mathsf{sk}_{\mathsf{CHET}}$, an ephemeral trapdoor etd, the original message $\boldsymbol{m}$, a hash $\boldsymbol{h}$ of $\boldsymbol{m}$ with its corresponding random value $\boldsymbol{r}$, and a new message $\boldsymbol{m}'$. It returns a new random value $\boldsymbol{r}'$ such that $\boldsymbol{h}$ remains a CH of $\boldsymbol{m}'$.

The ephemeral trapdoor etd in CHET is chosen by the party that computes the hash value such that holders of the main trapdoor are unable to compute a collision of the hash unless they are provided with an ephemeral trapdoor. Note that CHET can be constructed from a CH using from However, PCH scheme is a combination of ABE and CH schemes. An editor or a trapdoor holder will be able to rewrite data if its attribute satisfies the access policy. Unlike PCH, RPCH allows a revocation of a malicious trapdoor holder. Compared to PCH, RPCH is designed by combining RABE and CH scheme.

2.5.2 Security model

In the following, we briefly recall the CHET security model. Note that the security requirements of CHET are indistinguishability, public collision-resistance, private collision-resistance, and uniqueness. For more information, the reader is referred to [2,9].

2.5.2.1. Indistinguishability

The indistinguishability in CHET requires that it should be intractable for an adversary to distinguish whether a given random value $\boldsymbol{r}$ is returned by $\mathsf{Hash}_{\mathsf{CHET}}$ or $\mathsf{Adapt}_{\mathsf{CHET}}$. However, the level of indistinguishability, which is of our interest, is the full indistinguishability. We denote the corresponding game by $\mathsf{CHET.FIndGame}$.

CHET.FIndGame:

1. $\mathsf{PP}_{\mathsf{CHET}} \xleftarrow{\$} \mathsf{Setup}_{\mathsf{CHET}}(1^{\lambda})$
2. $b \xleftarrow{\$} \{0,1\}$
3. $b^* \leftarrow \mathcal{A}^{O_{\mathsf{CHashOrCAdapt}_{\mathsf{CHET}}}(\cdot,\cdot,\cdot,b)}(\mathsf{PP}_{\mathsf{CHET}})$
4. Return $b = b^*$

$O_{\mathsf{CHashOrCAdapt}_{\mathsf{CHET}}}(\cdot,\cdot,\cdot,b)$:
Input: $\mathsf{sk}_{\mathsf{CHET}}$, $\mathsf{pk}_{\mathsf{CHET}}$, $\boldsymbol{m}$ and $\boldsymbol{m}'$.
Output: $(\boldsymbol{h}_b, \boldsymbol{r}_b, \mathsf{etd}_b)$
1. $(\boldsymbol{h}_0, \boldsymbol{r}_0, \mathsf{etd}_0) \leftarrow \mathsf{HashCHET}(\mathsf{pk}_{\mathsf{CHET}}, \boldsymbol{m}')$
2. $(\boldsymbol{h}_1, \boldsymbol{r}_1, \mathsf{etd}_1) \leftarrow \mathsf{HashCHET}(\mathsf{pk}_{\mathsf{CHET}}, \boldsymbol{m})$
3. $\boldsymbol{r}_1 \leftarrow \mathsf{Adapt}(\mathsf{sk}_{\mathsf{CHET}}, \mathsf{etd}_1, \boldsymbol{m}, \boldsymbol{m}', \boldsymbol{r}_1, \boldsymbol{h}_1)$
4. Return $(\boldsymbol{h}_b, \boldsymbol{r}_b, \mathsf{etd}_b)$.

Let $\mathcal{A}$ be an adversary in the CHET full indistinguishability game. $\mathcal{A}$ wins the game when it guesses $\tilde{b}$ such that $\tilde{b} = b$. Its advantage denoted by $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{CH.FInd}}$ is given by

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{CHET.FInd}} = |\Pr[\mathsf{CHET.FIndGame}(\lambda) = 1] - 1/2|.$$

A CHET scheme is said to be fully indistinguishable if the advantage of any polynomial time adversary $\mathcal{A}$ in the CHET.FIndGame is negligible.

2.5.2.2. Public collision-resistance

In the public collision-resistance game of CHET, the adversary is authorized to have access to a collision oracle $O_{\mathsf{Adapt}'_{\mathsf{CHET}}}$. The public collision resistance requires that it should be infeasible to produce collisions, other than the ones produced by $O_{\mathsf{Adapt}'_{\mathsf{CHET}}}$.

CHET.PubColResGame:

1. $\mathsf{PP}_{\mathsf{CHET}} \xleftarrow{\$} \mathsf{Setup}_{\mathsf{CHET}}(1^{\lambda})$
2. $(\mathsf{sk}_{\mathsf{CHET}}, \mathsf{pk}_{\mathsf{CHET}}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathsf{CHET}}(\mathsf{PP}_{\mathsf{CHET}})$
3. $Q \leftarrow \varnothing$
4. $(\tilde{\boldsymbol{m}}, \tilde{\boldsymbol{r}}, \tilde{\boldsymbol{m}}', \tilde{\boldsymbol{r}}', \tilde{\boldsymbol{h}}) \leftarrow \mathcal{A}^{O_{\mathsf{Adapt}'_{\mathsf{CHET}}}(\mathsf{sk}_{\mathsf{CHET}},\cdot,\cdot,\cdot,\cdot,\cdot)}(\mathsf{pk}_{\mathsf{CHET}})$
5. If $\mathsf{Verif}_{\mathsf{CHET}}(\mathsf{pk}_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \tilde{\boldsymbol{r}}, \tilde{\boldsymbol{h}}) = 1 \wedge \mathsf{Verif}_{\mathsf{CHET}}(\mathsf{pk}_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}', \tilde{\boldsymbol{r}}', \tilde{\boldsymbol{h}}) = 1 \wedge \tilde{\boldsymbol{m}} \neq \tilde{\boldsymbol{m}}' \wedge \tilde{\boldsymbol{m}}' \notin Q$:
   Return 1
5. Return 0.

$O_{\mathsf{Adapt}'_{\mathsf{CHET}}}(\mathsf{sk}_{\mathsf{CHET}},\cdot,\cdot,\cdot,\cdot,\cdot)$:
Input: etd, $\boldsymbol{m}$, $\boldsymbol{r}$, $\boldsymbol{m}'$, and $\boldsymbol{h}$.
Output: $\boldsymbol{r}'$.
1. If $\mathsf{Verif}_{\mathsf{CHET}}(\mathsf{pk}_{\mathsf{CHET}}, \boldsymbol{m}, \boldsymbol{h}, \boldsymbol{r})$:
   Return $\perp$
2. $\boldsymbol{r}' \leftarrow \mathsf{Adapt}_{\mathsf{CHET}}(\mathsf{sk}_{\mathsf{CHET}}, \mathsf{etd}, \boldsymbol{m}, \boldsymbol{m}', \boldsymbol{h}, \boldsymbol{r})$
3. If $\boldsymbol{r}' = \perp$:
   Return $\perp$
4. $Q \leftarrow Q \cup \{\boldsymbol{m}, \boldsymbol{m}'\}$
5. Return $\boldsymbol{r}'$

The advantage of an adversary in the public collision-resistance game of CHET is the probability that the value 1 is returned. If we denote it by $\mathsf{Adv}_{\mathcal{A},\mathsf{CHET}}^{\mathsf{CHET.PubColRes}}$, we have:

$$\mathsf{Adv}_{\mathcal{A},\mathsf{CHET}}^{\mathsf{CHET.PubColRes}} = \Pr[\mathsf{CHET.PubColResGame}(\lambda) = 1].$$

Thus, a CHET scheme is public collision secure if the advantage of any probabilistic polynomial time (PPT) adversary $\mathcal{A}$ in CHET.PubColResGame is negligible. Note that since our security proof is based on [9, Theorem 4] RPCH, we are not concerned with the public collision-resistance property. For more detail on the public collision-resistance in CHET, see [2,9].

2.5.2.3. Private collision-resistance

The aim of the private collision-resistance in CHET is to ensure that even if the adversary holds one of the trapdoors, it should not be able to find collisions unless it holds both the secret key and the ephemeral trapdoor etd. For the formal definition of private collision-resistance in CHET, see [2]. However, in this paper we are interested in the strong private collision-resistance in CHET introduced in [9]. The strong private collision-resistance is similar to the private collision-resistance game but with a small difference. Indeed, a trapdoor holder adversary $\mathcal{A}$ should not be able to provide a collision of a hash without the knowledge of etd even if it is allowed to request arbitrarily many collisions. The strong private collision-resistance game CHET.SPrivColResGame in CHET is described as follows [9]:

CHET.SPrivColResGame:

1. $\mathsf{PP}_{\mathsf{CHET}} \xleftarrow{\$} \mathsf{Setup}_{\mathsf{CHET}}(1^{\lambda})$
2. $Q \leftarrow \varnothing$
3. $i \leftarrow 0$
4. $(\tilde{\mathsf{pk}}_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \tilde{\boldsymbol{r}}, \tilde{\boldsymbol{m}}', \tilde{\boldsymbol{r}}', \tilde{\boldsymbol{h}}) \leftarrow \mathcal{A}^{O_{\mathsf{Hash}'_{\mathsf{CHET}}}(\cdot,\cdot), O_{\mathsf{Adapt}'_{\mathsf{CHET}}}(\cdot,\cdot,\cdot,\cdot,\cdot)}(\mathsf{PP}_{\mathsf{CHET}})$
5. If $\mathsf{Verif}_{\mathsf{CHET}}(\tilde{\mathsf{pk}}_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \tilde{\boldsymbol{r}}, \tilde{\boldsymbol{h}}) = 1 \wedge \mathsf{Verif}_{\mathsf{CHET}}(\tilde{\mathsf{pk}}_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}', \tilde{\boldsymbol{r}}', \tilde{\boldsymbol{h}}) = 1 \wedge \tilde{\boldsymbol{m}} \neq \tilde{\boldsymbol{m}}' \wedge (\tilde{\mathsf{pk}}_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \tilde{\boldsymbol{h}}, \cdot, \cdot) \notin Q \wedge (\tilde{\mathsf{pk}}_{\mathsf{CHET}}, \cdot, \tilde{\boldsymbol{h}}, \cdot, \cdot) \notin Q$:
   Return 1
6. Return 0.

$O_{\mathsf{Hash}'_{\mathsf{CHET}}}(\cdot,\cdot)$:
Input: pk, $\boldsymbol{m}$.
Output: $(\boldsymbol{h}, \boldsymbol{r})$.
1. $(\boldsymbol{h}, \boldsymbol{r}, \mathsf{etd}) \leftarrow \mathsf{Hash}_{\mathsf{CHET}}(\mathsf{pk}, \boldsymbol{m})$
2. If $\boldsymbol{r} = \perp$:
   Return $\perp$
3. $i = i + 1$
4. $Q \leftarrow Q \cup \{(\mathsf{pk}, \boldsymbol{h}, \boldsymbol{m}, \mathsf{etd}, i)\}$
5. Return $(\boldsymbol{h}, \boldsymbol{r})$.

$O_{\mathsf{Adapt}'_{\mathsf{CHET}}}(\cdot,\cdot,\cdot,\cdot,\cdot)$:
Input: sk, $\boldsymbol{h}$, $\boldsymbol{m}$, $\boldsymbol{r}$, $\boldsymbol{m}'$, i.
Output: $\boldsymbol{r}'$.
1. If $(\mathsf{pk}, \boldsymbol{h}', \boldsymbol{m}'', \mathsf{etd}, i) \notin Q$ for some pk $\boldsymbol{h}'$, $\boldsymbol{m}''$, etd:
   Return $\perp$.
3. $\boldsymbol{r}' \leftarrow \mathsf{Adapt}_{\mathsf{CHET}}(\mathsf{sk}, \mathsf{etd}, \boldsymbol{m}, \boldsymbol{m}', \boldsymbol{h}, \boldsymbol{r})$
4. If $\boldsymbol{r}' \neq \perp$:
   $Q \leftarrow Q \cup \{(\mathsf{pk}, \boldsymbol{h}', \boldsymbol{m}, \mathsf{etd}, i), (\mathsf{pk}, \boldsymbol{h}', \boldsymbol{m}', \mathsf{etd}, i)\}$
5. Return $\boldsymbol{r}'$.

An adversary $\mathcal{A}$ wins in the strong private collision-resistance game CHET.SPrivColResGame when 1 is returned at the end of the game. Therefore, the advantage of an adversary in CHET.SPrivColResGame corresponds to the probability that the value 1 is returned as the of CHET.SPrivColResGame, i.e.,

$$\mathsf{Adv}^{\mathsf{CHET.SPrivColRes}}_{\mathcal{A},\mathsf{CHET}} = \Pr[\mathsf{CHET.SPrivColResGame}(\lambda) = 1].$$

A CHET scheme is said to be strongly private collision-resistant when this advantage is negligible for any PPT adversary $\mathcal{A}$.

2.5.2.4. Uniqueness

Uniqueness in CHET requires that for an adversarially given public key $\tilde{\mathsf{pk}}$, it will be difficult for the adversary to find two different random values $\tilde{\boldsymbol{r}}$ and $\tilde{\boldsymbol{r}}'$ for the same message $\tilde{\boldsymbol{m}}$ such that the hashes are equal.

CHET.UniGame:

1. $\mathsf{PP}_{\mathsf{CHET}} \xleftarrow{\$} \mathsf{Setup}_{\mathsf{CH}}(1^{\lambda})$
2. $(\tilde{\mathsf{pk}}, \tilde{\boldsymbol{m}}, \tilde{\boldsymbol{r}}, \tilde{\boldsymbol{r}}', \tilde{\boldsymbol{h}}) \xleftarrow{\$} \mathcal{A}(\mathsf{PP}_{\mathsf{CHET}})$
3. If $\mathsf{Verif}_{\mathsf{CH}}(\tilde{\mathsf{pk}}, \tilde{\boldsymbol{m}}, \tilde{\boldsymbol{r}}, \tilde{\boldsymbol{h}}) = 1 \wedge \mathsf{Verif}_{\mathsf{CH}}(\tilde{\mathsf{pk}}, \tilde{\boldsymbol{m}}', \tilde{\boldsymbol{r}}', \tilde{\boldsymbol{h}}) = 1 \wedge \tilde{\boldsymbol{r}} \neq \tilde{\boldsymbol{r}}'$:
   Return 1
4. Return 0.

The advantage of an adversary in the uniqueness game corresponds to the probability that 1 is returned at the end of CHET.UniGame. If we denote this advantage by $\mathsf{Adv}^{\mathsf{CHET.Uni}}_{\mathcal{A},\mathsf{CHET}}$, we have:

$$\mathsf{Adv}^{\mathsf{CHET.Uni}}_{\mathcal{A},\mathsf{CHET}} = \Pr[\mathsf{CHET.UniGame}(\lambda) = 1].$$

A CHET scheme is said to be unique when the advantage of any PPT adversary is negligible.

3 Lattice-based CHET scheme

In this section, based on the work of Derler et al. [9], we introduce a CH function with an ephemeral trapdoor. To this end, we first give a modified version of the trapdoor generation algorithm of Micciancio and Peikert [19, Algorithm 1] by including RO. Then, we describe our CHET and finally show that our CHET is secure under lattice problems, namely, the SIS and ISIS problems.

3.1 Trapdoor generator algorithm

In the Micciancio and Peikert trapdoor generator algorithm, the trapdoor is usually a matrix. However, for the purpose of our design, we need to generate a trapdoor that we would be able to transmit easily through a secure channel. To this end, we propose to generate the trapdoor matrix using an RO with a random seed as input. The seed is a random binary string of length λ. Therefore, in addition to the public matrix, the algorithm returns the seed as the trapdoor instead of a matrix. We denote the modified algorithm by $\mathsf{GenTrap}^{\mathsf{RO}}$ and describe it as follows:

$\mathsf{GenTrap}^{\mathsf{RO}}$:
Input: Integers n and q where q is a prime number.
Output: Public matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and the seed for the trapdoor $\mathbf{A}_\sigma^{-1}$ where $m = n(\lceil \log(q) \rceil + 2)$.

1. Choose randomly $\overline{\mathbf{A}} \xleftarrow{\$} \mathbb{Z}_q^{n \times \overline{m}}$ with $\overline{m} = 2n$.
2. $\mathsf{seed} \xleftarrow{\$} \{0,1\}^{\lambda}$.
3. Compute $\mathbf{R} \leftarrow \mathsf{RO}(\mathsf{seed})$, where $\mathbf{R} \in \{-1, 0, 1\}^{\overline{m} \times m_0}$ and $m_0 = n\lceil \log(q) \rceil$.
4. Compute $\mathbf{A} = [\overline{\mathbf{A}} \| \mathbf{G} - \overline{\mathbf{A}}\mathbf{R}]$, where $\mathbf{G} \in \mathbb{Z}_q^{n \times m_0}$.
5. Output seed and $\mathbf{A}$.

Note that in the modified trapdoor generator algorithm $\mathsf{GenTrap}^{\mathsf{RO}}$, the matrix $\mathbf{G}$ corresponds to the gadget matrix defined in Section 2.2. The trapdoor $\mathbf{A}_\sigma^{-1}$ is computed by using the RO and the secret seed $\mathbf{A}_\sigma^{-1} = \mathsf{RO}(\mathsf{seed})$. As the seed is a binary string, it can be encrypted and transmitted through a communication channel in a relatively efficient manner.

3.2 Construction

In this section, we discuss the design of a CHET based on our CH scheme. The public key and secret key in our lattice-based CHET scheme are computed using the Micciancio and Peikert trapdoor generator algorithm $(\mathsf{sk}_{\mathsf{CHET}}, \mathsf{pk}_{\mathsf{CHET}}) = (\mathbf{A}_{\sigma_1}^{-1}, \mathbf{A}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathsf{CHET}}(n_1, q_1, \mathcal{D}_{\sigma_1})$. The scheme is designed using a slightly modified version of Cash et al's CH [11] and Derler et al.'s framework [2,9]. In fact, in Cash et al.'s CH two public matrices $\mathbf{A}_0$ and $\mathbf{A}_1 \in \mathbb{Z}_q^{n\times m}$ are used. The CH of a message $\boldsymbol{m}$ is computed by $\boldsymbol{h} = \mathbf{A}_0\boldsymbol{m}^T + \mathbf{A}_1\boldsymbol{r}^T$, where $\boldsymbol{r}$ is a random vector. However, we propose to use only one public matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ instead of two matrices. The CH of a message $\boldsymbol{m}$ will be computed by $\boldsymbol{h} = \mathbf{A}\boldsymbol{r}_2 + \mathcal{H}_q(\mathsf{pk}_{\mathsf{CHET}}, \boldsymbol{m}, \boldsymbol{r}_1)$, where $\mathcal{H}_q$ is a cryptographic hash function defined from $\mathbb{Z}_q^*$ onto $\mathbb{Z}_q^m$. The random value $\boldsymbol{r}$ is given by $\boldsymbol{r} = (\boldsymbol{r}_1, \boldsymbol{r}_2)$, where $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are randomly chosen for computing $\boldsymbol{h}$.

For hashing a message $\boldsymbol{m}$ using a public matrix $\mathbf{A} \in \mathbb{Z}_{q_1}^{n_1\times m_1}$, we first generate an ephemeral key pair $(\mathsf{seed}, \tilde{\mathbf{A}}) \xleftarrow{\$} \mathsf{GenTrap}^{\mathsf{RO}}(n_2, q_2)$ by executing the modified trapdoor generator algorithm. Then, we set $\tilde{\boldsymbol{m}} = (\mathsf{pk}_{\mathsf{CHET}}, \mathsf{pk}'_{\mathsf{CHET}}, \boldsymbol{m})$ and choose randomly $\boldsymbol{r}_1 = (\boldsymbol{r}_{1,1}, \boldsymbol{r}_{1,2}) \xleftarrow{\$} \mathcal{D}_{\sigma_1}^{m_1} \times \mathbb{Z}_{q_1}^{\kappa}$ and $\boldsymbol{r}_2 = (\boldsymbol{r}_{2,1}, \boldsymbol{r}_{2,2}) \xleftarrow{\$} \mathcal{D}_{\sigma_2}^{m_2} \times \mathbb{Z}_{q_2}^{\kappa}$. We compute $\boldsymbol{h}_1 = \mathbf{A}\boldsymbol{r}_{1,2} + \mathcal{H}_{q_1}(\mathsf{pk}_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \boldsymbol{r}_{1,1})$ and $\boldsymbol{h}_2 = \tilde{\mathbf{A}}\boldsymbol{r}_{2,2} + \mathcal{H}_{q_2}(\mathsf{pk}'_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \boldsymbol{r}_{2,1})$, where $\mathcal{H}_{q_1}$ and $\mathcal{H}_{q_2}$ are two hash functions defined from $\mathbb{Z}_{q_1}^*$ onto $\mathbb{Z}_{q_1}^{m_1}$ and $\mathbb{Z}_{q_2}^*$ onto $\mathbb{Z}_{q_2}^{m_2}$, respectively. Finally, we return $(\boldsymbol{h}, \boldsymbol{r}, \mathsf{etd})$, where $\boldsymbol{h} = (\boldsymbol{h}_1, \boldsymbol{h}_2)$, $\boldsymbol{r} = (\boldsymbol{r}_1, \boldsymbol{r}_2)$, and $\mathsf{etd} = \mathsf{seed}$.

$\underline{\mathsf{KeyGen}_{\mathsf{CHET}}}$:
Input: Integers n_1 and q_1 where q_1 is a prime number. A Gaussian distribution $\mathcal{D}_{\sigma_1}$.
Output: $(\mathsf{sk}_{\mathsf{CHET}}, \mathsf{pk}_{\mathsf{CHET}}) = (\mathbf{A}_{\sigma_1}^{-1}, \mathbf{A})$

1. $(\mathbf{A}, \mathbf{A}_{\sigma_1}^{-1}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathsf{CH}}(n_1, q_1, \mathcal{D}_{\sigma_1})$
2. Set $\mathsf{sk}_{\mathsf{CHET}} = \mathbf{A}_{\sigma_1}^{-1}$ and $\mathsf{pk}_{\mathsf{CHET}} = \mathbf{A} \in \mathbb{Z}_{q_1}^{n_1\times m_1}$ where $m_1 = n_1(\lceil \log(q_1) \rceil + 2)$.
3. Return $(\mathsf{sk}_{\mathsf{CHET}}, \mathsf{pk}_{\mathsf{CHET}})$.

$\underline{\mathsf{Hash}_{\mathsf{CHET}}}$:
Input: Public parameters $\mathsf{PP}_{\mathsf{CHET}}$, public key $\mathsf{pk}_{\mathsf{CHET}} = \mathbf{A}$, message to hash $\boldsymbol{m}$.
Output: A hash $\boldsymbol{h}$, a random value $\boldsymbol{r}$ and an ephemeral trapdoor etd.

1. Generate a pair $(\mathsf{sk}'_{\mathsf{CHET}}, \mathsf{pk}'_{\mathsf{CHET}}) = (\mathsf{etd}, \tilde{\mathbf{A}}) \xleftarrow{\$} \mathsf{GenTrap}^{\mathsf{RO}}(n_2, q_2)$ of an ephemeral secret/public keys and set the ephemeral trapdoor $\mathsf{sk}'_{\mathsf{CHET}} = \mathsf{etd}$.
2. Set $\tilde{\boldsymbol{m}} = (\mathsf{pk}_{\mathsf{CHET}}, \mathsf{pk}'_{\mathsf{CHET}}, \boldsymbol{m})$
3. Choose randomly and uniformly $(\boldsymbol{r}_{1,1}, \boldsymbol{r}_{2,1}) \xleftarrow{\$} \mathbb{Z}_{q_1}^{\kappa} \times \mathbb{Z}_{q_2}^{\kappa}$
4. Compute $\boldsymbol{y}_1 = \mathcal{H}_q(\mathsf{pk}_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \boldsymbol{r}_{1,1})$ and $\boldsymbol{y}_2 = \mathcal{H}_q(\mathsf{pk}'_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \boldsymbol{r}_{2,1})$
5. Choose uniformly and randomly $(\boldsymbol{r}_{1,2}, \boldsymbol{r}_{2,2}) \xleftarrow{\$} \mathcal{D}_{\sigma_1}^{m_1} \times \mathcal{D}_{\sigma_2}^{m_2}$
6. Compute $\boldsymbol{h}_1 = \mathbf{A}\boldsymbol{r}_{1,2}^T + \boldsymbol{y}_1$ and $\boldsymbol{h}_2 = \tilde{\mathbf{A}}\boldsymbol{r}_{2,2}^T + \boldsymbol{y}_2$
7. Set $\boldsymbol{r}_1 = (\boldsymbol{r}_{1,1}, \boldsymbol{r}_{1,2})$ and $\boldsymbol{r}_2 = (\boldsymbol{r}_{2,1}, \boldsymbol{r}_{2,2})$
8. Set $\boldsymbol{h}' = (\boldsymbol{h}_1, \boldsymbol{h}_2)$ and $\boldsymbol{r} = (\boldsymbol{r}_1, \boldsymbol{r}_2)$
9. Return $\boldsymbol{h}$, $\boldsymbol{r}$ and etd where $\boldsymbol{h} = (\boldsymbol{h}', \mathsf{pk}'_{\mathsf{CHET}})$.

$\underline{\mathsf{Verif}_{\mathsf{CHET}}}$:
Input: $\mathsf{pk}_{\mathsf{CHET}} = \mathbf{A}$, message $\boldsymbol{m}$, hash $\boldsymbol{h}$ and random value $\boldsymbol{r}$.
Output: $b \in \{0, 1\}$

1. Parse $\boldsymbol{h}$ as $(\boldsymbol{h}', \mathsf{pk}'_{\mathsf{CHET}})$, $\boldsymbol{h}'$ as $(\boldsymbol{h}_1, \boldsymbol{h}_2)$ and $\boldsymbol{r}$ as $(\boldsymbol{r}_1, \boldsymbol{r}_2)$ where $\mathsf{pk}'_{\mathsf{CHET}} = \tilde{\mathbf{A}}$
2. Parse $\boldsymbol{r}_1$ as $(\boldsymbol{r}_{1,1}, \boldsymbol{r}_{1,2})$ and $\boldsymbol{r}_1$ as $(\boldsymbol{r}_{2,1}, \boldsymbol{r}_{2,2})$.
3. Set $\tilde{\boldsymbol{m}} = (\mathsf{pk}_{\mathsf{CHET}}, \mathsf{pk}'_{\mathsf{CHET}}, \boldsymbol{m})$
4. Compute $\boldsymbol{z}_1 = \mathbf{A}\boldsymbol{r}_{1,2}^T + \mathcal{H}_{q_1}(\mathsf{pk}_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \boldsymbol{r}_{1,1})$, and $\boldsymbol{z}_2 = \tilde{\mathbf{A}}\boldsymbol{r}_{2,2}^T + \mathcal{H}_{q_2}(\mathsf{pk}'_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \boldsymbol{r}_{2,1})$
5. If $\boldsymbol{h}_1 = \boldsymbol{z}_1$ and $\boldsymbol{h}_2 = \boldsymbol{z}_2$

Return 1.
6. Return 0.

For adaptiveness, we take as input a pair of secret/public keys $(\mathsf{sk}_{\mathsf{CHET}}, \mathsf{pk}_{\mathsf{CHET}})$, an ephemeral trapdoor etd, where $\tilde{\mathbf{A}}_{\sigma}^{-1} = \mathsf{RO}(\mathsf{etd})$, an original message $\boldsymbol{m}$, a hash $\boldsymbol{h} = (\boldsymbol{h}', \mathsf{pk}'_{\mathsf{CHET}})$ of $\boldsymbol{m}$ with the corresponding random value $\boldsymbol{r} = (\boldsymbol{r}_1, \boldsymbol{r}_2)$, and a modified message $\boldsymbol{m}'$. We set $\tilde{\boldsymbol{m}} = (\mathsf{pk}_{\mathsf{CHET}}, \mathsf{pk}'_{\mathsf{CHET}}, \boldsymbol{m})$, $\tilde{\boldsymbol{m}}' = (\mathsf{pk}_{\mathsf{CHET}}, \mathsf{pk}'_{\mathsf{CHET}}, \boldsymbol{m}')$ and randomly choose $(\boldsymbol{r}'_{1,1}, \boldsymbol{r}'_{2,1}) \xleftarrow{\$} \mathbb{Z}_{q_1}^{\kappa} \times \mathbb{Z}_{q_2}^{\kappa}$. Then, we compute $\boldsymbol{r}'_{1,2} = \mathbf{A}_{\sigma}^{-1}(\boldsymbol{h}_1 - \mathcal{H}_{q_1}(\mathsf{pk}_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}', \boldsymbol{r}'_{1,1}))$ and $\boldsymbol{r}'_{2,2} = \tilde{\mathbf{A}}_{\sigma}^{-1}(\boldsymbol{h}_2 - \mathcal{H}_{q_2}(\mathsf{pk}_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}', \boldsymbol{r}'_{2,1}))$. We output the random value $\boldsymbol{r}' = (\boldsymbol{r}'_1, \boldsymbol{r}'_2)$ where $\boldsymbol{r}'_1 = (\boldsymbol{r}'_{1,1}, \boldsymbol{r}'_{1,2})$ and $\boldsymbol{r}'_2 = (\boldsymbol{r}'_{2,1}, \boldsymbol{r}'_{2,2})$.

$\mathsf{Adapt}_{\mathsf{CHET}}$:
Input: A secret key $\mathsf{sk}_{\mathsf{CHET}} = \mathbf{A}_{\sigma}^{-1}$, an ephemeral trapdoor etd, a message $\boldsymbol{m}'$, a hash $\boldsymbol{h}$ with its corresponding random value $\boldsymbol{r}$, the modified message $\boldsymbol{m}'$.
Output: A new random value $\boldsymbol{r}'$.
1. If $\mathsf{Verif}_{\mathsf{CH}}(\mathsf{pk}_{\mathsf{CHET}}, \boldsymbol{m}, \boldsymbol{h}, \boldsymbol{r}) = 0$
   Return $\perp$.
2. Compute $\tilde{\mathbf{A}}_{\sigma}^{-1} = \mathsf{RO}(\mathsf{etd})$.
3. Parse $\boldsymbol{h}$ as $(\boldsymbol{h}', \mathsf{pk}'_{\mathsf{CHET}})$, and $\boldsymbol{h}'$ as $(\boldsymbol{h}_1, \boldsymbol{h}_2)$.
4. Set $\tilde{\boldsymbol{m}}' = (\mathsf{pk}_{\mathsf{CHET}}, \mathsf{pk}'_{\mathsf{CHET}}, \boldsymbol{m}')$.
5. Choose randomly $(\boldsymbol{r}'_{1,1}, \boldsymbol{r}'_{2,1}) \xleftarrow{\$} \mathbb{Z}_{q_1}^{\kappa} \times \mathbb{Z}_{q_2}^{\kappa}$.
6. Compute $\boldsymbol{r}'_{1,2} = \mathbf{A}_{\sigma_1}^{-1}(\boldsymbol{h}_1 - \mathcal{H}_{q_1}(\mathsf{pk}_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}', \boldsymbol{r}'_{1,1}))$ and $\boldsymbol{r}'_{2,2} = \tilde{\mathbf{A}}_{\sigma_2}^{-1}(\boldsymbol{h}_2 - \mathcal{H}_{q_2}(\mathsf{pk}'_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}', \boldsymbol{r}'_{2,1}))$.
7. Set $\boldsymbol{r}' = (\boldsymbol{r}'_1, \boldsymbol{r}'_2)$ where $\boldsymbol{r}'_1 = (\boldsymbol{r}'_{1,1}, \boldsymbol{r}'_{1,2})$ and $\boldsymbol{r}'_2 = (\boldsymbol{r}'_{2,1}, \boldsymbol{r}'_{2,2})$.
8. Return $\boldsymbol{r}'$.

For the correctness of our proposed lattice-based CHET, we have the following proposition:

Proposition 2. *The proposed lattice-based* CHET *is correct.*

Proof. Let $\boldsymbol{m}$ be an original message and the pair $(\boldsymbol{h}, \boldsymbol{r})$ its CH with a random value where $\boldsymbol{h} = (\boldsymbol{h}', \mathsf{pk}'_{\mathsf{CHET}})$ and $\boldsymbol{h}' = (\boldsymbol{h}_1, \boldsymbol{h}_2)$. Let $\boldsymbol{m}'$ be a modification of $\boldsymbol{m}$. Suppose that the CH is computed honestly and we are in the presence of an honest trapdoor owner. For forging a hash, the trapdoor holder first chooses randomly and uniformly $(\boldsymbol{r}'_{1,1}, \boldsymbol{r}'_{2,1}) \in \mathbb{Z}_{q_1}^{\kappa} \times \mathbb{Z}_{q_2}^{\kappa}$ and computes $\boldsymbol{u}_1 = \boldsymbol{h}_1 - \mathcal{H}_{q_1}(\mathsf{pk}_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}', \boldsymbol{r}'_{1,1})$ and $\boldsymbol{u}_2 = \boldsymbol{h}_2 - \mathcal{H}_{q_2}(\mathsf{pk}'_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}', \boldsymbol{r}'_{2,1})$. Then, it uses the trapdoor $\mathbf{A}_{\sigma_1}^{-1}$ and $\tilde{\mathbf{A}}_{\sigma_2}^{-1} = \mathsf{RO}(\mathsf{etd})$ for sampling $\boldsymbol{r}'_{1,2}$ and $\boldsymbol{r}'_{2,2}$, respectively, such that $\mathbf{A}\boldsymbol{r}'^{T}_{1,2} = \boldsymbol{u}_1$ and $\tilde{\mathbf{A}}\boldsymbol{r}'^{T}_{2,2} = \boldsymbol{u}_2$. The probability that $\|\boldsymbol{r}'_{1,2}\| > \eta_1\sigma_1\sqrt{m_1}$ or $\|\boldsymbol{r}'_{2,2}\| > \eta_2\sigma_2\sqrt{m_2}$ is negligible [29], and moreover, we have

$$\mathbf{A}\boldsymbol{r}'^{T}_{1,2} = \boldsymbol{h}_1 - \mathcal{H}_{q_1}(\mathsf{pk}_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}', \boldsymbol{r}'_{1,1}) \Rightarrow \mathbf{A}\boldsymbol{r}'^{T}_{1,2} + \mathcal{H}_{q_1}(\mathsf{pk}_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}', \boldsymbol{r}'_{1,1}) = \boldsymbol{h}_1, \tag{1}$$

$$\mathbf{A}\boldsymbol{r}'^{T}_{2,2} = \boldsymbol{h}_2 - \mathcal{H}_{q_2}(\mathsf{pk}'_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}', \boldsymbol{r}'_{2,1}) \Rightarrow \mathbf{A}\boldsymbol{r}'^{T}_{2,2} + \mathcal{H}_{q_2}(\mathsf{pk}'_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}', \boldsymbol{r}'_{2,1}) = \boldsymbol{h}_2. \tag{2}$$

Thus, we have

$$\mathbf{A}\boldsymbol{r}'^{T}_{1,2} + \mathcal{H}_{q_1}(\mathsf{pk}_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}', \boldsymbol{r}'_{1,1}) = \boldsymbol{h}_1 = \mathbf{A}\boldsymbol{r}^{T}_{1,2} + \mathcal{H}_{q_1}(\mathsf{pk}_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \boldsymbol{r}_{1,1}), \tag{3}$$

$$\mathbf{A}\boldsymbol{r}'^{T}_{2,2} + \mathcal{H}_{q_2}(\mathsf{pk}'_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}', \boldsymbol{r}'_{2,1}) = \boldsymbol{h}_2 = \mathbf{A}\boldsymbol{r}^{T}_{2,2} + \mathcal{H}_{q_2}(\mathsf{pk}'_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \boldsymbol{r}_{2,1}). \tag{4}$$

From (3) and (4), we have $\mathsf{Verif}_{\mathsf{CHET}}(\mathsf{pk}_{\mathsf{CHET}}, \boldsymbol{m}', \boldsymbol{h}, \boldsymbol{r}') = \mathsf{Verif}_{\mathsf{CHET}}(\mathsf{pk}_{\mathsf{CHET}}, \boldsymbol{m}, \boldsymbol{h}, \boldsymbol{r}) = 1$. □

3.3 Security analysis

For the security analysis of our CHET scheme, it is important to note that in addition to the correctness, there are four required security properties for a CHET: indistinguishability, public collision-resistance, private collision-resistance, and uniqueness.

Proposition 3. *The proposed lattice-based* CHET *is fully indistinguishable.*

Proof. Let $\mathcal{A}$ be an adversary against our scheme in the full indistinguishability game. As mentioned in Section 2.5, in the full indistinguishability game, $\mathcal{A}$ is allowed to generate the secret and the public keys. Suppose that $\mathcal{A}$ is able to win the full indistinguishability game with a probability more than $\frac{1}{2}$, i.e., with a probability equal to $\frac{1}{2} + \nu$, where ν is not negligible. That means that for $\boldsymbol{h}_b$ and $\boldsymbol{r}_b$ as given in Subsection 2.5, the adversary is able to distinguish at least from which message $\boldsymbol{h}_b$ is computed or from which algorithm $\boldsymbol{r}_b$ is computed with a probability equal to $\frac{1}{2} + \nu$ with ν non-negligible. We have:

(i) For $b \in \{0,1\}$, we have $\boldsymbol{r}_b = (\boldsymbol{r}_{b,1}, \boldsymbol{r}_{b,2})$, where $\boldsymbol{r}_{b,1} = (\boldsymbol{r}_{b,1,1}, \boldsymbol{r}_{b,1,2}) \in \mathbb{Z}_{q_1}^{\kappa} \times \mathcal{D}_{\sigma_1}^{m_1}$, and $\boldsymbol{r}_{b,2} = (\boldsymbol{r}_{b,2,1}, \boldsymbol{r}_{b,2,2}) \in \mathbb{Z}_{q_2}^{\kappa} \times \mathcal{D}_{\sigma_2}^{m_2}$. Hence, the distribution of the products $\mathbf{A}\boldsymbol{r}_{b,1,2}^T$ and $\tilde{\mathbf{A}}\boldsymbol{r}_{b,2,2}^T$ are is statistically close to uniform over $\mathbb{Z}_{q_1}^{n_1}$ and $\mathbb{Z}_{q_2}^{n_2}$, respectively [20]; the distribution of $\boldsymbol{h}_{b,1} = \mathbf{A}\boldsymbol{r}_{b,1,2}^T + \mathcal{H}_q(\mathsf{pk}_{\mathsf{CHET}}, \boldsymbol{m}_b, \boldsymbol{r}_{b,1,1})$ and $\boldsymbol{h}_{b,2} = \tilde{\mathbf{A}}\boldsymbol{r}_{b,2,2}^T + \mathcal{H}_q(\mathsf{pk}'_{\mathsf{CHET}}, \boldsymbol{m}_b, \boldsymbol{r}_{b,2,1})$ are statistically close to uniform over $\mathbb{Z}_{q_1}^{n_1}$ and $\mathbb{Z}_{q_2}^{n_2}$ respectively. Therefore, the distribution of $\boldsymbol{h}' = (\boldsymbol{h}_{b,1}, \boldsymbol{h}_{b,2})$ is statistically close to uniform over $\mathbb{Z}_{q_1}^{n_1} \times \mathbb{Z}_{q_2}^{n_2}$. Moreover, pk'_{CHET} is independent from the message $\boldsymbol{m}$; thus, $\mathcal{A}$ cannot distinguish $\boldsymbol{h}_b = (\boldsymbol{h}', \mathsf{pk}'_{\mathsf{CHET}})$ with a probability $\frac{1}{2} + \nu$ with ν being non-negligible.

(ii) For $b \in \{0,1\}$, the distributions of $\boldsymbol{h}_{b,1}$ and $\boldsymbol{h}_{b,2}$ are statistically close to uniform over $\mathbb{Z}_{q_1}^{n_1}$ and $\mathbb{Z}_{q_2}^{n_2}$ respectively. Moreover, $\boldsymbol{r}_{b,1,1}$ and $\boldsymbol{r}_{b,2,1}$ are chosen randomly, uniformly, and independently from $\boldsymbol{r}_{b,1,2}$ and $\boldsymbol{r}_{b,2,2}$, respectively. Thus, the distributions of $\boldsymbol{u}_1 = \boldsymbol{h}_{b,1} - \mathcal{H}_{q_1}(\mathsf{pk}_{\mathsf{CHET}}, \boldsymbol{m}_b, \boldsymbol{r}_{b,1,1})$ and $\boldsymbol{u}_2 = \boldsymbol{h}_{b,2} - \mathcal{H}_{q_2}(\mathsf{pk}'_{\mathsf{CHET}}, \boldsymbol{m}_b, \boldsymbol{r}_{b,2,1})$ are also statistically close to uniform over $\mathbb{Z}_{q_1}^{n_1}$ and $\mathbb{Z}_{q_2}^{n_2}$, respectively. Thus, $\boldsymbol{r}_{b,1,2}$ and $\boldsymbol{r}_{b,2,2}$ are the sampled solutions from Gaussian distributions of random equations $\mathbf{A}\boldsymbol{x}_2^T = \boldsymbol{u}_1$ and $\tilde{\mathbf{A}}\boldsymbol{x}_2^T = \boldsymbol{u}_2$, respectively. Hence, $\mathcal{A}$ cannot distinguish $\boldsymbol{r}_{b,1} = (\boldsymbol{r}_{b,1,1}, \boldsymbol{r}_{b,1,2})$ or $\boldsymbol{r}_{b,1} = (\boldsymbol{r}_{b,1,1}, \boldsymbol{r}_{b,1,2})$ with a probability $\frac{1}{2} + \nu'$ with ν' being non-negligible, and consequently, it will not be able to distinguish $\boldsymbol{r}_b = (\boldsymbol{r}_{b,1}, \boldsymbol{r}_{b,2})$ with a probability $\frac{1}{2} + \nu$ with ν being non-negligible.

From (*i*) and (*ii*), we conclude that our scheme is fully indistinguishable. □

Proposition 4. *If the* $\mathsf{ISIS}_{q,n,m,\beta}$ *and* $\mathsf{SIS}_{q,n,m,\delta}$ *assumptions hold, the proposed lattice-based* CHET *is unique.*

Proof. Let $\mathcal{A}$ be an adversary against the proposed CHET in the uniqueness game. The goal of $\mathcal{A}$ is to find a random value a public key $\mathsf{pk}_{\mathsf{CHET}}$, a message $\boldsymbol{m}$, a hash $\boldsymbol{h} = (\boldsymbol{h}', \mathsf{pk}'_{\mathsf{CHET}})$, and two random values $\boldsymbol{r}$ and $\boldsymbol{r}'$ such that $\boldsymbol{r} \neq \boldsymbol{r}'$ and $\mathsf{Verif}_{\mathsf{CHET}}(\mathsf{pk}_{\mathsf{CHET}}, \boldsymbol{m}, \boldsymbol{h}, \boldsymbol{r}) = \mathsf{Verif}_{\mathsf{CHET}}(\mathsf{pk}_{\mathsf{CHET}}, \boldsymbol{m}, \boldsymbol{h}, \boldsymbol{r}') = 1$. This means that the following holds:

$$\mathbf{A}\boldsymbol{r}_{1,2}^T + \mathcal{H}_{q_1}(\mathsf{pk}_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \boldsymbol{r}_{1,1}) = \mathbf{A}\boldsymbol{r}_{1,2}'^T + \mathcal{H}_{q_1}(\mathsf{pk}_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \boldsymbol{r}'_{1,1}) = \boldsymbol{h}_1, \tag{5}$$

$$\tilde{\mathbf{A}}\boldsymbol{r}_{2,2}^T + \mathcal{H}_{q_2}(\mathsf{pk}'_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \boldsymbol{r}_{2,1}) = \tilde{\mathbf{A}}\boldsymbol{r}_{2,2}'^T + \mathcal{H}_{q_2}(\mathsf{pk}'_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \boldsymbol{r}'_{2,1}) = \boldsymbol{h}_2, \tag{6}$$

where $\mathsf{pk}'_{\mathsf{CHET}} = \tilde{\mathbf{A}}$. The adversary $\mathcal{A}$ can proceed from different ways as follows:

(1) $\mathcal{A}$ decides to keep $(\boldsymbol{r}'_{1,2}, \boldsymbol{r}'_{2,2}) = (\boldsymbol{r}_{1,2}, \boldsymbol{r}_{2,2})$, and thus, it should find $\boldsymbol{r}'_{1,1}$ and $\boldsymbol{r}'_{2,1}$ such that $(\boldsymbol{r}'_{1,1}, \boldsymbol{r}'_{2,1}) \neq (\boldsymbol{r}_{1,1}, \boldsymbol{r}_{2,1})$. $\mathcal{A}$ will try to proceed by finding collisions $\boldsymbol{r}'_{1,1}$ and $\boldsymbol{r}'_{2,1}$ of $\boldsymbol{r}_{1,1}$ and $\boldsymbol{r}_{2,1}$, respectively. In fact, it will try to find $(\boldsymbol{r}'_{1,1}, \boldsymbol{r}'_{2,1}) \neq (\boldsymbol{r}_{1,1}, \boldsymbol{r}_{2,1})$ such that $\mathcal{H}_{q_1}(\mathsf{pk}_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \boldsymbol{r}'_{1,1}) = \mathcal{H}_{q_1}(\mathsf{pk}_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \boldsymbol{r}_{1,1})$ and $\mathcal{H}_{q_2}(\mathsf{pk}'_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \boldsymbol{r}'_{2,1}) = \mathcal{H}_{q_2}(\mathsf{pk}'_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \boldsymbol{r}_{2,1})$. Finally, it will return $\boldsymbol{r}' = (\boldsymbol{r}'_1, \boldsymbol{r}'_2)$ with $\boldsymbol{r}'_1 = (\boldsymbol{r}'_{1,1}, \boldsymbol{r}_{1,2})$ and

$\boldsymbol{r}'_2 = (\boldsymbol{r}'_{2,1}, \boldsymbol{r}_{2,2})$. As the underlying hash functions $\mathcal{H}_{q_1}$ and $\mathcal{H}_{q_2}$ are collision-resistant, the probability that $\mathcal{A}$ succeeds is negligible and less than $\frac{2q_{\mathcal{H}_q}}{q^\kappa} + \left(\frac{q_{\mathcal{H}_q}}{q^\kappa}\right)^2$, where $q_{\mathcal{H}_q}$ is the maximum of number of queries that the adversary is authorized to make to the hash oracles and $q = \min(q_1, q_2)$.

(2) $\mathcal{A}$ decides to have $(\boldsymbol{r}'_{1,1}, \boldsymbol{r}'_{2,1}) \neq (\boldsymbol{r}_{1,1}, \boldsymbol{r}_{2,1})$ and $(\boldsymbol{r}'_{1,2}, \boldsymbol{r}'_{2,2}) \neq (\boldsymbol{r}_{1,2}, \boldsymbol{r}_{2,2})$. This means that $\mathcal{A}$ is able to solve at least one of the equations $\mathbf{A}\boldsymbol{x}_1^T = \boldsymbol{s}_1$ and $\tilde{\mathbf{A}}\boldsymbol{x}_2^T = \boldsymbol{s}_2$ where $\boldsymbol{s}_1 = \boldsymbol{h} - \mathcal{H}_{q_1}(\mathsf{pk}_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \boldsymbol{r}'_{1,1})$ and $\boldsymbol{s}_2 = \boldsymbol{h} - \mathcal{H}_{q_2}(\mathsf{pk}'_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \boldsymbol{r}'_{2,1})$ by choosing randomly $(\boldsymbol{r}'_{1,1}, \boldsymbol{r}'_{2,1}) \in \mathbb{Z}_{q_1}^\kappa \times \mathbb{Z}_{q_2}^\kappa$. Thus, the probability that $\mathcal{A}$ will be able to produce the random value in this case is less than $2\varepsilon_{\mathsf{ISIS}_{q,n,m,\beta}} + \varepsilon^2_{\mathsf{ISIS}_{q,n,m,\beta}}$, where $\varepsilon_{\mathsf{ISIS}_{q,n,m,\beta}} = \max(\varepsilon_{\mathsf{ISIS}_{q_1,n_1,m_1,\beta_1}}, \varepsilon_{\mathsf{ISIS}_{q_2,n_2,m_2,\beta_2}})$. $\varepsilon_{\mathsf{ISIS}_{q_1,n_1,m_1,\beta_1}}$ and $\varepsilon_{\mathsf{ISIS}_{q_2,n_2,m_2,\beta_2}}$ are the advantage of an adversary to solve an instance of $\mathsf{ISIS}_{q_1,n_1,m_1,\beta_1}$ and $\mathsf{ISIS}_{q_2,n_2,m_2,\beta_2}$, respectively, where $\beta_1 = \eta_1\sigma_1\sqrt{m_1}$ and $\beta_2 = \eta_2\sigma_2\sqrt{m_2}$ for a well-chosen value of η_1 and η_2.

(3) $\mathcal{A}$ decides to keep $(\boldsymbol{r}'_{1,1}, \boldsymbol{r}'_{2,1}) = (\boldsymbol{r}_{1,1}, \boldsymbol{r}_{2,1})$. According to the fact that $\boldsymbol{r} \neq \boldsymbol{r}'$, $\mathcal{A}$ should find $(\boldsymbol{r}'_{1,2}, \boldsymbol{r}'_{2,2}) \neq (\boldsymbol{r}_{1,2}, \boldsymbol{r}_{2,2})$ such that $\mathbf{A}\boldsymbol{r}_{1,2}^T = \mathbf{A}\boldsymbol{r}'^T_{1,2}$ and $\tilde{\mathbf{A}}\boldsymbol{r}_{2,2}^T = \tilde{\mathbf{A}}\boldsymbol{r}'^T_{2,2}$. This means that $\mathcal{A}$ is able to solve at least an instance of $\mathsf{SIS}_{q_1,n_1,m_1,\delta_1}$ or $\mathsf{SIS}_{q_2,n_2,m_2,\delta_2}$ with $\delta_1 = 2\beta_1$ and $\delta_2 = 2\beta_2$, i.e., we have

$$\mathbf{A}\boldsymbol{r}_{1,2}^T = \mathbf{A}\boldsymbol{r}'^T_{1,2} \Rightarrow \mathbf{A}(\boldsymbol{r}_{1,2} - \boldsymbol{r}'_{1,2})^T = 0, \tag{7}$$

$$\tilde{\mathbf{A}}\boldsymbol{r}_{2,2}^T = \tilde{\mathbf{A}}\boldsymbol{r}'^T_{2,2} \Rightarrow \tilde{\mathbf{A}}(\boldsymbol{r}_{2,2} - \boldsymbol{r}'_{2,2})^T = 0. \tag{8}$$

Thus, the probability that $\mathcal{A}$ will be able to produce the random value in this case is less than $2\varepsilon_{\mathsf{SIS}_{q,n,m,\delta}} + \varepsilon^2_{\mathsf{SIS}_{q,n,m,\delta}}$, where $\varepsilon_{\mathsf{SIS}_{q,n,m,\delta}} = \max(\varepsilon_{\mathsf{SIS}_{q_1,n_1,m_1,\delta_1}}, \varepsilon_{\mathsf{SIS}_{q_2,n_2,m_2,\delta_2}})$. $\varepsilon_{\mathsf{SIS}_{q_1,n_1,m_1,\delta_1}}$ and $\varepsilon_{\mathsf{SIS}_{q_2,n_2,m_2,\delta_2}}$ are the advantage of an adversary to solve an instance of $\mathsf{SIS}_{q_1,n_1,m_1,\delta_1}$ and $\mathsf{SIS}_{q_2,n_2,m_2,\delta_2}$, respectively.

To sum up, if $\mathsf{Adv}^{\mathsf{CHET.Uni}}_{\mathcal{A}}$ is the advantage of an adversary $\mathcal{A}$ in the uniqueness game, the following holds:

$$\mathsf{Adv}^{\mathsf{CHET.Uni}}_{\mathcal{A}}(\lambda) \leq 2\left(\varepsilon_{\mathsf{SIS}_{q,n,m,\delta}}(\lambda) + \varepsilon_{\mathsf{ISIS}_{q,n,m,\beta}}(\lambda) + \frac{q_{\mathcal{H}_q}}{q^\kappa}\right) + \varepsilon^2_{\mathsf{ISIS}_{q,n,m,\beta}}(\lambda) + \varepsilon^2_{\mathsf{SIS}_{q,n,m,\delta}}(\lambda) + \left(\frac{q_{\mathcal{H}_q}}{q^\kappa}\right)^2.$$

This concludes the proof. □

Proposition 5. *If* $\mathsf{ISIS}_{q,n,m,\beta}$ *assumption holds, the proposed lattice-based* CHET *scheme is strongly private collision-resistant.*

Proof. Let $(\boldsymbol{m}, \boldsymbol{r}) \neq (\boldsymbol{m}', \boldsymbol{r}')$ be collision for the proposed CHET. It immediately yields to a solution of the system:

$$\mathbf{A}(\boldsymbol{r}_{1,2} - \boldsymbol{r}'_{1,2})^T = \mathcal{H}_{q_1}(\mathsf{pk}_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \boldsymbol{r}'_{1,1}) - \mathcal{H}_{q_1}(\mathsf{pk}_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \boldsymbol{r}_{1,1}), \tag{9}$$

$$\tilde{\mathbf{A}}(\boldsymbol{r}_{2,2} - \boldsymbol{r}'_{2,2})^T = \mathcal{H}_{q_2}(\mathsf{pk}'_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \boldsymbol{r}'_{2,1}) - \mathcal{H}_{q_2}(\mathsf{pk}'_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \boldsymbol{r}_{2,1}). \tag{10}$$

As in the strong private collision-resistance game, the adversary $\mathcal{A}$ has access to the secret key $\mathsf{sk}_{\mathsf{CHET}} = \mathbf{A}_\sigma^{-1}$ but not to the ephemeral trapdoor etd; thus, winning means that it is able to solve $\tilde{\mathbf{A}}(\boldsymbol{r}_{2,2} - \boldsymbol{r}'_{2,2})^T = \mathcal{H}_{q_2}(\mathsf{pk}'_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \boldsymbol{r}'_{2,1}) - \mathcal{H}_{q_2}(\mathsf{pk}'_{\mathsf{CHET}}, \tilde{\boldsymbol{m}}, \boldsymbol{r}_{2,1})$. $\tilde{\mathbf{A}} \in \mathbb{Z}_{q_2}^{n_2 \times m_2}$ is an ephemeral public key, and $\boldsymbol{r}'_{2,1} \in \mathbb{Z}_{q_2}^\kappa$ is a random vector. Consequently, $\mathcal{A}$ is able to solve an instance of $\mathsf{ISIS}_{q,n,m,\beta}$. □

4 New lattice-based RABE scheme

In this section, we introduce an adaptive secure lattice-based ciphertext policy RABE scheme. We design it using Tsabary's lattice-based ABE [17] for which the secure assumption relies on decisional learning with

errors. Unlike similar other lattice-based RABE, our scheme is based on a single attribute per user instead of a set of attributes. Attributes are represented by a binary string of length ℓ corresponding to the length of input points of t-CNF policies. For the purpose of managing the revocation, we use a binary tree where the leaves correspond to the user identities in the system. In the following, we first briefly recall the use of a binary tree in this context where we consider user identities id_i to be binary strings of length k_{id}. Then, we describe our scheme and show that it is correct. Near the end of this section, we give a security analysis of our scheme followed by a comparison of our scheme with two other lattice-based RABE schemes in terms of the size of the master secret key, the master public key, and the user secret key of our scheme.

4.1 Binary tree structure

For the design of our RABE, we consider user identities id as a binary string of length k_{id}. We use a binary tree with $2^{k_{\mathsf{id}}}$ leaves labeled by a key $j \in [2^{k_{\mathsf{id}}}, 2^{k_{\mathsf{id}}+1} - 1]$ and an identity id_i, where k_{id} corresponds to the binary size of identities id_i. We denote a user leaf path from the root by $\mathsf{Path}(\mathsf{id}_i)$, where id_i is the user identity. When a user with identity id_i is revoked at time T, we have $\mathsf{Path}(\mathsf{id}_i) \cap \mathsf{Node}_\mathsf{T} = \emptyset$, where Node_T is an updated set of nodes obtained by executing the key updated nodes algorithm KUNode as described in the paper by Naor et al. [37]. Otherwise, there is an unique integer $i' \in [1, 2^{k_{\mathsf{id}}+1} - 1]$ such that $\mathsf{Path}(\mathsf{id}_i) \cap \mathsf{Node}_\mathsf{T} = \{i'\}$.

For instance, in Figure 1, leaves 11, 14, and 15 are unassigned. Thus, if there is a new user in the system, it should be randomly assigned to one of these leaves. Moreover, each node contains a list of user identities that are under its control. The updated set of nodes will only contain the root when the revocation list is empty, i.e., $\mathsf{Node}_\mathsf{T} = \{1\}$ if $\mathsf{RL} = \emptyset$. When a user is revoked, none of the nodes from its identity path will be included in the updated set of nodes Node_T. For instance, if the users with identity id_1 and id_5 are revoked, i.e, $\mathsf{RL} = \{\mathsf{id}_1, \mathsf{id}_5\}$, the updated set of nodes Node_T will become $\mathsf{Node}_\mathsf{T} = \{9, 5, 7, 12\}$. Thus, we have $\mathsf{Node}_\mathsf{T} \cap \mathsf{Path}(\mathsf{id}_1) = \emptyset$ and $\mathsf{Node}_\mathsf{T} \cap \mathsf{Path}(\mathsf{id}_5) = \emptyset$. However, the user identity id_2 and id_3 are not revoked; thus, we have $\mathsf{Node}_\mathsf{T} \cap \{\mathsf{id}_5\} = \{9\}$ and $\mathsf{Node}_\mathsf{T} \cap \{\mathsf{id}_5\} = \{5\}$. For all unassigned leaves or leaves with non-revoked identity, there is only one node from their path hat is included in Node_T. We can summarize the key updated nodes algorithm as follows:

KUNode
Input: st, $\mathsf{RL} = \{(i, \mathsf{id}, \mathsf{T}_i)\}$, T
Output: Node_T
1. Initialize $S = \emptyset$ and $\mathsf{Node}_\mathsf{T} = \emptyset$
2. For all $(i, \mathsf{id}, \mathsf{T}_i) \in \mathsf{RL}$:
   If $\mathsf{T}_i \leq \mathsf{T}$:
      Compute $S = S \cup \mathsf{Path}(\mathsf{id})$
3. For all $\theta \in S$:
   If $\theta_{left} \notin S$:

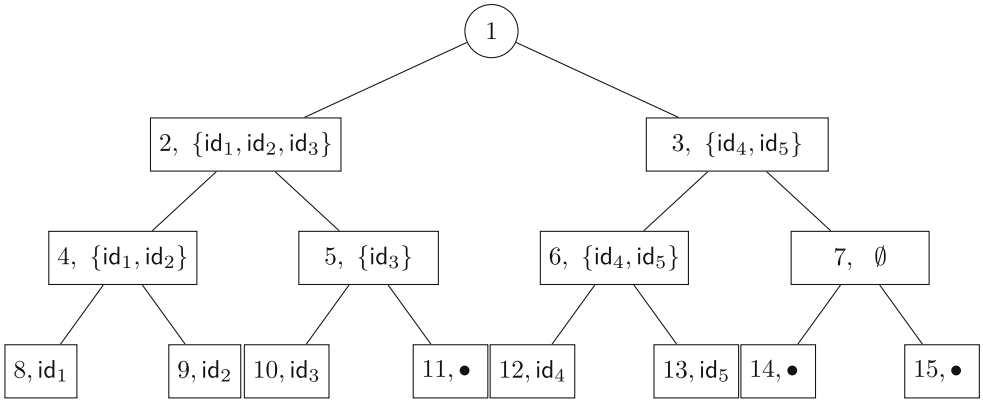

Figure 1: Binary tree.

$\mathsf{Node}_\mathsf{T} = \mathsf{Node}_\mathsf{T} \cup \{\theta_{letf}\}$

If $\theta_{right} \notin S$:

$\mathsf{Node}_\mathsf{T} = \mathsf{Node}_\mathsf{T} \cup \{\theta_{right}\}$

4. Return Node_T

4.2 Construction

The main tool used in the design of our adaptive secure RABE scheme is the conforming cPRF. Let $\mathsf{F} = (\mathsf{F.Setup}, \mathsf{F.Eval}, \mathsf{F.Constrain}, \mathsf{F.ConstrainEval})$ be a conforming cPRF for a class of functions $\mathcal{F}$ with input length ℓ and output length k. Let $\mathsf{F.msk}$ be a fixed master secret key of F and $\mathsf{F.PP}$ its public parameters. For all $f \in \mathcal{F}$ and $\boldsymbol{x} \in \{0,1\}^\ell$, let us consider the following circuits:

- $U_{\sigma \to \boldsymbol{x}} : \{0,1\}^\lambda \longrightarrow \{0,1\}^k$,
  $\mathsf{F.msk} \longmapsto \mathsf{Eval}(\mathsf{F.msk}, \boldsymbol{x})$
- $U_{\sigma \to f} : \{0,1\}^\lambda \longrightarrow \{0,1\}^{\ell_f}$,
  $\mathsf{F.msk} \longmapsto \mathsf{Constrain}(\mathsf{F.msk}, f)$
- $U_{f \to \boldsymbol{x}} : \{0,1\}^{\ell_f} \longrightarrow \{0,1\}^k$
  $\mathsf{csk}_f \longmapsto \mathsf{ConstrainEval}(\mathsf{csk}_f, \boldsymbol{x})$.

where ℓ_f is the length of constrained key csk_f of f. Note that F is a conforming cPRF, and hence, the gradual evaluation requirement is held by definition. However, for all $f \in \mathcal{F}$ and for all $\boldsymbol{x} \in \{0,1\}^\ell$, if $f(\boldsymbol{x}) = 1$, we have $U_{\sigma \to \boldsymbol{x}} \equiv U_{f \to \boldsymbol{x}} \circ U_{\sigma \to f}$ (see Section 2.3).

For the design of RABE, we need to consider the master secret key size λ of the conforming cPRF satisfying $\lambda = k$. However, instead of taking only the security parameter λ as input, our algorithm Setup takes as input the security parameter λ and the maximum number $|\mathcal{ID}| = 2^{k_{\mathsf{id}}} = N$ of users in the system. In our algorithm Setup, we first execute the algorithm P.Setup for generating the master secret key F.msk and public parameters F.PP for F. Then, we execute the trapdoor generator matrix algorithm GenTrap for sampling $(\mathbf{B}, \mathbf{B}_{\tau_0}^{-1})$ with $\mathbf{B} \in \mathbb{Z}_q^{n \times m'}$, and then, we sample uniformly a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m_0 \lambda}$ and a vector $\boldsymbol{y} \in \mathbb{Z}_q^n$. To finish, we set $\mathsf{RL} = \emptyset$, $\mathsf{st} = \mathsf{BT}$, $\mathsf{mpk} = (\mathbf{A}, \mathbf{B}, \boldsymbol{y}, \mathsf{F.PP})$ and $\mathsf{msk} = (\mathbf{B}_{\tau_0}^{-1}, \mathsf{F.msk})$. We sum up the description of the algorithm Setup of our scheme as follows:

Setup:

Input: Security parameter 1^λ, number $N = 2^{k_{\mathsf{id}}} = |\mathcal{ID}|$ of users in the system.
Output: Master public key mpk, master secret key msk, state st and revocation list RL with additional public parameter PP.

1. Generate the master key and public parameters $(\mathsf{F.msk}, \mathsf{F.PP}) = \mathsf{P.Setup}(1^\lambda)$ of F and set $\sigma = \mathsf{F.msk}$.
2. Choose randomly $\overline{\mathbf{B}} \xleftarrow{\$} \mathbb{Z}_q^{n \times w}$ with $w = \lceil \log(q) \rceil + 2\lambda$
3. Compute $(\mathbf{B}, \mathbf{B}_{\tau_0}^{-1}) \leftarrow \mathsf{GenTrap}^{\mathcal{D}_{\tau_0}}(\overline{\mathbf{B}}, \mathbf{H})$ where $\mathbf{B} \in \mathbb{Z}_q^{n \times m'}$, $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$ is a random invertible matrix of size $n \times n$ and $m' = (n+1)\lceil \log(q) \rceil + 2\lambda$.
4. Choose two hash functions $\mathcal{H}_q$ and $\mathcal{H}'_q$, where $\mathcal{H}_q$ takes as input a vector over $\mathbb{Z}_q$ with arbitrary length and outputs a vector of $\mathbb{Z}_q^n$ and $\mathcal{H}'_q$ takes as input a vector over $\mathbb{Z}_q$ with arbitrary length and outputs a binary vector of length m'.
5. Sample uniformly a matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m_0 \lambda}$ and sample randomly $\boldsymbol{y} \xleftarrow{\$} \mathbb{Z}_q^n$.
6. Set $\mathsf{st} = \mathsf{BT}$ and $\mathsf{RL} = \emptyset$.
7. Set $\mathsf{mpk} = (\mathbf{A}, \mathbf{B}, \boldsymbol{y}, \mathsf{F.PP}, \mathcal{H}_q, \mathcal{H}'_q)$ and $\mathsf{msk} = (\mathbf{B}_{\tau_0}^{-1}, \sigma)$.

Note that in our scheme, we consider that the users in the system are identified by an identity $\mathsf{id} \in \{0,1\}^{k_{\mathsf{id}}}$, where k_{id} is a well-chosen positive integer while taking into account the scalability of the system. The secret key generation algorithm SKGen takes as input a master key msk, an identity id, a state st, and an attribute $\boldsymbol{x}$. It outputs a secret key $\mathsf{sk}_{\mathsf{id}}$ for the user with identity id. Now, we recall the following theorem, which is necessary for the description of our secret key generation algorithm SKGen and encryption algorithm Enc.

Theorem 1. [17] *There exist efficient deterministic algorithms* EvalF *and* EvalFX *such that for all* $n, q, \ell \in \mathbb{N}$, *and* $m_0 = n\lceil \log(q) \rceil$, *for any depth* d *Boolean circuit* $f : \{0,1\}^{\ell} \longrightarrow \{0,1\}^{k}$, *for every* $\boldsymbol{x} \in \{0,1\}^{\ell}$, *and for any matrix* $\mathbf{A} \in \mathbb{Z}_q^{n \times m_0 \ell}$, *the outputs* $\mathbf{H} \leftarrow \mathsf{EvalF}(f, \mathbf{A})$ *and* $\widehat{\mathbf{H}} \leftarrow \mathsf{EvalFX}(f, \boldsymbol{x}, \mathbf{A})$ *are both in* $\mathbb{Z}_q^{m_0 \ell \times m_0 k}$, *and it holds that* $\|\mathbf{H}\|_{\infty}, \|\widehat{\mathbf{H}}\|_{\infty} \leq (2m_0)^d$ *and* $[\mathbf{A} - \boldsymbol{x} \otimes \mathbf{G}]\widehat{\mathbf{H}} = \mathbf{A}\mathbf{H} - f(\boldsymbol{x}) \otimes \mathbf{G} \pmod{q}^2$, *where* $\otimes$ *denotes the tensor product.*

Moreover, for any $f : \{0,1\}^{\ell} \longrightarrow \{0,1\}^{k}$, $g : \{0,1\}^{k} \longrightarrow \{0,1\}^{t}$, *for any matrix* $\mathbf{A} \in \mathbb{Z}_q^{n \times m_0 \ell}$, *the outputs* $\mathbf{H}_f \leftarrow \mathsf{EvalF}(f, \mathbf{A})$, $\mathbf{H}_g \leftarrow \mathsf{EvalF}(g, \mathbf{A}\mathbf{H}_f)$ *and* $\mathbf{H}_{f \circ g} \leftarrow \mathsf{EvalF}(g \circ f, \mathbf{A})$ *satisfy* $\mathbf{H}_f \mathbf{H}_g = \mathbf{H}_{g \circ f}$.

We apply Theorem 1 to circuits $U_{\sigma \to f}, U_{\sigma \to \boldsymbol{x}}$ and $U_{\boldsymbol{x} \to f}$ for designing our secret key generation algorithm SKGen as follows:

SKGen:
Input: Master secret key msk, state st, identity id and attribute $\boldsymbol{x} \in \mathsf{A}$
Output: A secret key $\mathsf{sk}_{\mathsf{id}}$ for the user with identity id and an updated state st.
1. Sample an unassigned leaf node in st and label it by the identity id.
2. Compute $\mathbf{H}_{\sigma \to \boldsymbol{x}} = \mathsf{EvalF}(U_{\sigma \to \boldsymbol{x}}, \mathbf{A})$
3. Compute $\mathbf{A}_{\boldsymbol{x}} = \mathbf{A}\mathbf{H}_{\sigma \to \boldsymbol{x}}$
4. Compute $\boldsymbol{r}_{\boldsymbol{x}} = \mathsf{P.Eval}_{\sigma}(\boldsymbol{x})$
5. Compute $\mathbf{H}_{\boldsymbol{r}_{\boldsymbol{x}}} = \mathsf{EvalF}(I_{\boldsymbol{r}_{\boldsymbol{x}}}, \mathbf{A}_{\boldsymbol{x}})$ where $I_{\boldsymbol{r}_{\boldsymbol{x}}} : \{0,1\}^{k} \longrightarrow \{0,1\}$ is a function such that $I_{\boldsymbol{r}_{\boldsymbol{x}}}(\boldsymbol{r}) = 1$ if $\boldsymbol{r}_{\boldsymbol{x}} = \boldsymbol{r}$
6. Compute $[\mathbf{B} \| \mathbf{A}_{\boldsymbol{x}, \boldsymbol{r}_{\boldsymbol{x}}}]_{\tau}^{-1}$ by using $\mathbf{B}_{\tau_0}^{-1}$ where $\mathbf{A}_{\boldsymbol{x}, \boldsymbol{r}_{\boldsymbol{x}}} = \mathbf{A}_{\boldsymbol{x}} \mathbf{H}_{\boldsymbol{r}_{\boldsymbol{x}}}$
7. Compute $\boldsymbol{e}_{\mathsf{id}} = \mathcal{H}'_q(\mathbf{A}, \mathbf{B}, q, \mathsf{id}, \boldsymbol{z}) \in \{0,1\}^{m'}$ and $\boldsymbol{y}_2 = \boldsymbol{y} + \boldsymbol{y}_1$, where $\boldsymbol{z} \xleftarrow{\$} \mathbb{Z}_q^{\lambda_0}$ and $\boldsymbol{y}_1 = \mathbf{B}\boldsymbol{e}_{\mathsf{id}}$ with $2\lambda = \lambda_0 \lfloor \log(q) \rfloor$.
8. Sample $\boldsymbol{s}_{\mathsf{id}} = [\mathbf{B} \| \mathbf{A}_{\boldsymbol{x}, \boldsymbol{r}_{\boldsymbol{x}}}]_{\tau}^{-1}(\boldsymbol{y}_2)$
9. Set $\mathsf{sk}_{\mathsf{id}} = (\boldsymbol{x}, \boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{s}_{\mathsf{id}}, \boldsymbol{e}_{\mathsf{id}})$,

Note that, for a binary tree with $2^{k_{\mathsf{id}}}$ leaves, we have for each leaf $k_{\mathsf{id}} + 1$ nodes from the root. Therefore, as mentioned earlier if an assigned leaf is identified by (j, id), where $\mathsf{id} \in \{1, \ldots, 2^{k_{\mathsf{id}}}\}$ and $j \in [2^{k_{\mathsf{id}}}, 2^{k_{\mathsf{id}}+1} - 1]$, we can denote the path of each identity id in the tree by Path(id), which is a subset of $[1, 2^{k_{\mathsf{id}}+1} - 1]$ of size $k_{\mathsf{id}} + 1$ containing 1 and j. In the algorithm KUpd, we first execute the update node algorithm KUNode(st, RL, T) to compute the updated node set $\mathsf{Node}_{\mathsf{T}}$; then, for all node $\theta \in \mathsf{Node}_{\mathsf{T}}$, we sample $\boldsymbol{e}_{\theta,1} \xleftarrow{\$} \{0,1\}^{m'}$, and compute $\boldsymbol{z} = \mathbf{B}\boldsymbol{e}_{\theta,1}^{T}$, $\boldsymbol{u}_{\theta,\mathsf{T}} = \boldsymbol{z}' - \boldsymbol{z} \in \mathbb{Z}_q^{n}$ and $\boldsymbol{e}_{\theta,2} = \mathbf{B}_{\tau_0}^{-1}(\boldsymbol{u}_{\theta,\mathsf{T}})$, where $\boldsymbol{z}' = \sum_{i=1}^{2^{k_{\mathsf{id}}+1}-1} \mathcal{H}_q(\mathbf{A}, \mathbf{B}, q, i, \mathsf{T})$. Finally, we output the updated key material $\mathsf{ku}_{\mathsf{T}} = \{(\theta, \boldsymbol{e}_{\theta,\mathsf{T}}, \mathsf{T}) : \theta \in \mathsf{Node}_{\mathsf{T}}\}$ with $\boldsymbol{e}_{\theta,\mathsf{T}} = (\boldsymbol{e}_{\theta,1}, \boldsymbol{e}_{\theta,2})$.

KUpd:
Input: Master secret key msk, state st, time period T and revocation list RL.
Output: updated key material ku_{T} for the time period T.
1. Compute $\mathsf{Node}_{\mathsf{T}} = \mathsf{KUNode}(\mathsf{st}, \mathsf{RL}, \mathsf{T})$
2. For each $\theta \in \mathsf{Node}_{\mathsf{T}}$:
   (a) Sample $\boldsymbol{e}_{\theta,1} \xleftarrow{\$} \{0,1\}^{m'}$
   (b) Compute $\boldsymbol{z} = \mathbf{B}\boldsymbol{e}_{\theta,1}^{T}$, $\boldsymbol{u}_{\theta,\mathsf{T}} = \boldsymbol{z}' - \boldsymbol{z} \in \mathbb{Z}_q^{n}$ and $\boldsymbol{e}_{\theta,2} = \mathbf{B}_{\tau_0}^{-1}(\boldsymbol{u}_{\theta,\mathsf{T}})$ where $\boldsymbol{z}' = \sum_{i=1}^{2^{k_{\mathsf{id}}+1}-1} \mathcal{H}_q(\mathbf{A}, \mathbf{B}, q, i, \mathsf{T})$
3. Return $\mathsf{ku}_{\mathsf{T}} = \{(\theta, \boldsymbol{e}_{\theta,\mathsf{T}}, \mathsf{T}) : \theta \in \mathsf{Node}_{\mathsf{T}}\}$ with $\boldsymbol{e}_{\theta,\mathsf{T}} = (\boldsymbol{e}_{\theta,1}, \boldsymbol{e}_{\theta,2})$

Rev:
Input: Identity $\mathsf{id} \in \mathcal{ID}$, revocation list RL and time period $\mathsf{T} \in \mathcal{T}$.
Output: Updated revocation list RL
Return $\mathsf{RL} = \mathsf{RL} \cup \{(i, \mathsf{id}, \mathsf{T})\}$, where i is the key of the leaf assigned to the identity id.

As a conforming cPRF F for access policies $\mathcal{F}$ is used, a key simulation algorithm P.KeySim is required for generating the constrained key csk_f of the access policy $f \in \mathcal{F}$ (see [17, Definition 9]). Therefore, for encrypting a message $\boldsymbol{m}$, we compute $\boldsymbol{u}_0 = \boldsymbol{s}\mathbf{B} + \boldsymbol{e}_0$ and $\boldsymbol{u}_1 = \boldsymbol{s}[\mathbf{A}_f - \mathsf{csk}_f \otimes \mathbf{G}] + \boldsymbol{e}_1$, where $\boldsymbol{s}$ is chosen uniformly and randomly in $\mathbb{Z}_q^n$, $\boldsymbol{e}_0$ and $\boldsymbol{e}_1$ are chosen from the distributions $\chi^{m'}$ and $\tilde{\chi}^{m_0 \ell_f}$, respectively. The matrix $\mathbf{A}_f$ is defined by $\mathbf{A}_f = \mathbf{A}\mathbf{H}_{\sigma \to f}$, where $\mathbf{H}_{\sigma \to f} = \mathsf{EvalF}(U_{\sigma \to f}, \mathbf{A})$. Then, we compute $u_2 = \boldsymbol{s}\boldsymbol{y}'^T + \boldsymbol{s}\boldsymbol{y}^T + \boldsymbol{e}_2 + \lfloor q/2 \rceil \boldsymbol{m}$, where $\boldsymbol{y}' = \sum_{i=1}^{2^{k_{\mathsf{id}}+1}-1} \mathcal{H}_q(\mathbf{A}, \mathbf{B}, q, i, \mathsf{T})$ and $\boldsymbol{e}_2 \xleftarrow{\$} \chi$. Finally, we output $\mathsf{ct} = (\mathsf{csk}_f, \boldsymbol{u}_0, \boldsymbol{u}_1, u_2)$ as a ciphertext of $\boldsymbol{m}$.

Enc:
Input: Master public key mpk, message $\boldsymbol{m} \in \{0,1\}$ and access policy f and time period T.
Output: Ciphertext ct

1. Sample $\mathsf{csk}_f \xleftarrow{\$} \mathsf{P.KeySim}(\mathsf{F.PP}, f)$
2. Sample uniformly $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\boldsymbol{e}_0 \xleftarrow{\$} \chi^{m'}$, $\boldsymbol{e}_1 \xleftarrow{\$} \tilde{\chi}^{m_o \ell_f}$ and $\boldsymbol{e}_2 \xleftarrow{\$} \chi$
3. Compute $\boldsymbol{u}_0 = \boldsymbol{s}\mathbf{B} + \boldsymbol{e}_0$ and $\boldsymbol{u}_1 = \boldsymbol{s}[\mathbf{A}_f - \mathsf{csk}_f \otimes \mathbf{G}] + \boldsymbol{e}_1$
4. Compute $\boldsymbol{y}' = \sum_{i=1}^{2^{k_{\mathsf{id}}+1}-1} \mathcal{H}_q(\mathbf{A}, \mathbf{B}, q, i, \mathsf{T})$
5. Compute and $u_2 = \boldsymbol{s}\boldsymbol{y}'^T + \boldsymbol{s}\boldsymbol{y}^T + \boldsymbol{e}_2 + \lfloor q/2 \rceil \boldsymbol{m}$
6. Set $\mathsf{ct} = (\mathsf{csk}_f, \boldsymbol{u}_0, \boldsymbol{u}_1, u_2)$.

Before presenting the decryption algorithm Dec, we describe the decoding-key generation algorithm DKGen. In our algorithm DKGen, if $\mathsf{Path(id)} \cap \mathsf{Node}_\mathsf{T} = \varnothing$, we return the fail symbol $\perp$. Otherwise, note that there is only one node $\theta \in \mathsf{Path(id)} \cap \mathsf{Node}_\mathsf{T}$ and the output is decoding-key $\mathsf{dk}_{\mathsf{id},\mathsf{T}} = (\boldsymbol{s}_{\theta,\mathsf{T}}, \boldsymbol{e}_\mathsf{T})$, where $\boldsymbol{s}_{\theta,\mathsf{T}} = \boldsymbol{e}_{\theta,2}$ and $\boldsymbol{e}_\mathsf{T} = \boldsymbol{e}_{\theta,1} - \boldsymbol{e}_{\mathsf{id}}$ with $(\theta, \boldsymbol{e}_{\theta,\mathsf{T}}, \mathsf{T}) \in \mathsf{ku}_\mathsf{T}$ and $\boldsymbol{e}_{\theta,\mathsf{T}} = (\boldsymbol{e}_{\theta,1}, \boldsymbol{e}_{\theta,2})$.

For the decryption algorithm Dec, we first parse ct as $(\mathsf{csk}_f, \boldsymbol{u}_0, \boldsymbol{u}_1, u_2)$. Then, we compute $\boldsymbol{c} = u_2 - [\boldsymbol{u}_0 || \boldsymbol{u}_1 \widehat{\mathbf{H}}_{\mathsf{csk}_f \to \boldsymbol{r}'} \widehat{\mathbf{H}}_{\boldsymbol{r}_\boldsymbol{x}, \boldsymbol{r}'}] \boldsymbol{s}_{\mathsf{id}}^T - \boldsymbol{u}_0 \boldsymbol{s}_{\theta,\mathsf{T}}^T - \boldsymbol{u}_0 \boldsymbol{e}_\mathsf{T}^T$, where $\boldsymbol{r}' = U_{f \to \boldsymbol{x}}(\mathsf{csk}_f)$, $\widehat{\mathbf{H}}_{\mathsf{csk}_f \to \boldsymbol{r}'} = \mathsf{EvalFX}(U_{f \to \boldsymbol{x}}, \mathsf{csk}_f, \mathbf{A}_f)$ and $\widehat{\mathbf{H}}_{\boldsymbol{r}_\boldsymbol{x}, \boldsymbol{r}'} = \mathsf{EvalFX}(I_\boldsymbol{x}, \boldsymbol{r}', \mathbf{A}_\boldsymbol{x})$. Finally, if $|\boldsymbol{c}| > q/4$, we output $\boldsymbol{m} = 1$ as the plaintext; else, we output $\boldsymbol{m} = 0$.

DKGen:
Input: Secret key sk_id and updated key material ku_T
Output: Decryption key $\mathsf{dk}_{\mathsf{id},\mathsf{T}}$ or $\perp$

1. Find node $\theta \in \mathsf{Path(id)} \cap \mathsf{Node}_\mathsf{T}$.
2. If $\mathsf{Path(id)} \cap \mathsf{Node}_\mathsf{T} = \varnothing$ return $\perp$.
3. Otherwise, find $(\theta, *, *)$ in ku_T and output $\mathsf{dk}_{\mathsf{id},\mathsf{T}} = (\boldsymbol{s}_{\theta,\mathsf{T}}, \boldsymbol{e}_\mathsf{T})$ with $\boldsymbol{e}_\mathsf{T} = \boldsymbol{e}_{\theta,1} - \boldsymbol{e}_{\mathsf{id}}$ and $\boldsymbol{s}_{\theta,\mathsf{T}} = \boldsymbol{e}_{\theta,2}$

Dec:
Input: Decryption key $\mathsf{dk}_{\mathsf{id},\mathsf{T}}$ and ciphertext ct
Output: Plaintext $\boldsymbol{m} \in \{0,1\}$

1. Parse $\mathsf{dk}_{\mathsf{id},\mathsf{T}}$ as $(\boldsymbol{x}, \boldsymbol{r}_\boldsymbol{x}, \boldsymbol{s}_{\mathsf{id}}, \boldsymbol{s}_{\theta,\mathsf{T}}, \boldsymbol{e}_\mathsf{T})$ and ct as $(\mathsf{csk}_f, \boldsymbol{u}_0, \boldsymbol{u}_1, \boldsymbol{u}_2)$.
2. Compute $\boldsymbol{r}' = U_{f \to \boldsymbol{x}}(\mathsf{csk}_f)$.
3. If $\boldsymbol{r}_\boldsymbol{x} = \boldsymbol{r}'$, then abort. Otherwise, compute $\mathbf{A}_f$ and $\mathbf{A}_\boldsymbol{x}$.
4. Compute $\widehat{\mathbf{H}}_{\mathsf{csk}_f \to \boldsymbol{r}'} = \mathsf{EvalFX}(U_{f \to \boldsymbol{x}}, \mathsf{csk}_f, \mathbf{A}_f)$ and $\widehat{\mathbf{H}}_{\boldsymbol{r}_\boldsymbol{x}, \boldsymbol{r}'} = \mathsf{EvalFX}(I_\boldsymbol{x}, \boldsymbol{r}', \mathbf{A}_\boldsymbol{x})$.
5. Compute $c = u_2 - [\boldsymbol{u}_0 || \boldsymbol{u}_1 \widehat{\mathbf{H}}_{\mathsf{csk}_f \to \boldsymbol{r}'} \widehat{\mathbf{H}}_{\boldsymbol{r}_\boldsymbol{x}, \boldsymbol{r}'}] \boldsymbol{s}_{\mathsf{id}}^T - \boldsymbol{u}_0 \boldsymbol{s}_{\theta,\mathsf{T}}^T - \boldsymbol{u}_0 \boldsymbol{e}_\mathsf{T}^T$
6. Output $\boldsymbol{m} = 0$ if $|c| \le q/4$. Otherwise, output $\boldsymbol{m} = 1$.

Correctness

As mentioned earlier, in our decryption algorithm, we compute $\boldsymbol{r}' = U_{f \to \boldsymbol{x}}(\boldsymbol{s}_f)$ and then abort the decoding if $\boldsymbol{r}' = \boldsymbol{r}_\boldsymbol{x}$. Otherwise, we compute $c = u_2 - [\boldsymbol{u}_0 || \boldsymbol{u}_1 \widehat{\mathbf{H}}_{\mathsf{csk}_f \to \boldsymbol{r}'} \widehat{\mathbf{H}}_{\boldsymbol{r}_\boldsymbol{x}, \boldsymbol{r}'}] \boldsymbol{s}_{\mathsf{id}}^T - \boldsymbol{u}_0 \boldsymbol{s}_{\theta,\mathsf{T}}^T - \boldsymbol{u}_0 \boldsymbol{e}_\mathsf{T}^T$. Note that based on the random value of the conforming cPRF, it was shown in [17] that $\boldsymbol{r}' \neq \boldsymbol{r}_\boldsymbol{x}$ with all but negligible probability. It was shown also that if $\boldsymbol{r}' \neq \boldsymbol{r}_\boldsymbol{x}$, we have:

$$\boldsymbol{u}_1 \widehat{\mathbf{H}}_{\mathsf{csk}_f \to \boldsymbol{r}'} \widehat{\mathbf{H}}_{\boldsymbol{r}_\boldsymbol{x}, \boldsymbol{r}'} = \boldsymbol{s}\mathbf{A}_{\boldsymbol{x}, \boldsymbol{r}_\boldsymbol{x}} + \boldsymbol{e}_1',$$

with $\boldsymbol{e}_1' = \boldsymbol{e}_1\widehat{\mathbf{H}}_{\mathsf{csk}_f \to \boldsymbol{r}'}\widehat{\mathbf{H}}_{\boldsymbol{r}_x,\boldsymbol{r}'}$. Therefore, we have

$$\begin{aligned} c &= u_2 - [\boldsymbol{u}_0 || \boldsymbol{u}_1\widehat{\mathbf{H}}_{\mathsf{csk}_f \to \boldsymbol{r}'}\widehat{\mathbf{H}}_{\boldsymbol{r}_x,\boldsymbol{r}'}]\boldsymbol{s}_{\mathsf{id}}^T - \boldsymbol{u}_0\boldsymbol{s}_{\theta,\mathsf{T}}^T - \boldsymbol{u}_0\boldsymbol{e}_{\mathsf{T}}^T \\ &= \boldsymbol{s}\boldsymbol{y}^T + \boldsymbol{s}\boldsymbol{y}'^T + \boldsymbol{e}_2 + \lfloor q/2 \rceil \boldsymbol{m} - [\boldsymbol{u}_0 || \boldsymbol{u}_1\widehat{\mathbf{H}}_{\mathsf{csk}_f \to \boldsymbol{r}'}\widehat{\mathbf{H}}_{\boldsymbol{r}_x,\boldsymbol{r}'}]\boldsymbol{s}_{\mathsf{id}}^T - \boldsymbol{u}_0\boldsymbol{s}_{\theta,\mathsf{T}}^T - \boldsymbol{u}_0\boldsymbol{e}_{\mathsf{T}}^T \\ &= \boldsymbol{e}_2 + \lfloor q/2 \rceil \boldsymbol{m} - [\boldsymbol{e}_0 || \boldsymbol{e}_1']\boldsymbol{s}_{\mathsf{id}}^T - \boldsymbol{e}_0\boldsymbol{s}_{\theta,\mathsf{T}}^T - \boldsymbol{e}_0\boldsymbol{e}_{\mathsf{T}}^T. \end{aligned}$$

Based on the properties of the discrete Gaussian distribution, we have $\|\boldsymbol{s}_{\mathsf{id}}\|_\infty \leq \tau\sqrt{m' + m_0}$ (respectively, $\|\boldsymbol{s}_{\theta,\mathsf{T}}\|_\infty \leq \tau_0\sqrt{m'}$) with all but $2^{-(m'+m_0)}$ (respectively, $2^{-m'}$) probability. In addition, due to the fact that $\|\boldsymbol{e}_1'\|_\infty \leq m_0^2\ell_f k\tilde{B}(2m_0)^{d_{ConEv}+1}$ (see [17]), if m', k, $\ell_f \in O(n, \lceil \log(q) \rceil)$, $\tilde{B} \in O(B, n)$, $\tau \in O(k, \lambda, (2m_0)^{d+3})$, and $\boldsymbol{u} = \boldsymbol{e}_2 - [\boldsymbol{e}_0 || \boldsymbol{e}_1']\boldsymbol{s}_{\mathsf{id}}^T - \boldsymbol{e}_0\boldsymbol{s}_{\theta,\mathsf{T}}^T - \boldsymbol{e}_0\boldsymbol{e}_{\mathsf{T}}^T$, we have with all but negligible probability that:

$$\begin{aligned} |\boldsymbol{u}| &\leq |\boldsymbol{e}_2| + (m'\|\boldsymbol{e}_0\|_\infty + m_0\|\boldsymbol{e}_1'\|_\infty)\|\boldsymbol{s}_{\mathsf{id}}\|_\infty + m'\|\boldsymbol{e}_0\|_\infty\|\boldsymbol{s}_{\theta,\mathsf{T}}\|_\infty + m'\|\boldsymbol{e}_0\|_\infty\|\boldsymbol{e}_{\mathsf{T}}\|_\infty \\ &\leq B + (m'B + m_0^3\ell_f k\tilde{B}(2m_0)^{d_{ConEv}+1})\tau\sqrt{m' + m_0} + m'B\tau_0\sqrt{m'} + m'B \\ &\leq B + (3m'B + m_0^3\ell_f k\tilde{B}(2m_0)^{d_{ConEv}+1})\tau\sqrt{m' + m_0} \\ &\leq B.\ poly(n, \lceil \log(q) \rceil)(2m_0)^{d_{ConEv}+d+4}. \end{aligned}$$

By denoting $E' = 4.\ poly(n, \lceil \log(q) \rceil)(2m_0)^{d_{ConEv}+d+4}$ and $E = B.\ poly(n, \lceil \log(q) \rceil)(2m_0)^{d_{ConEv}+d+4}$, if we choose E' to be bounded by q/B, then E is bounded by $q/4$. Therefore, if $\boldsymbol{m} = 0$, then $|\boldsymbol{c}| \leq q/4$; otherwise, $|\boldsymbol{c}| > q/4$. We now have the following proposition:

Proposition 6. *The proposed lattice-based* RABE *is correct.*

4.3 Security analysis and key size comparison

With regard to the security of the proposed RABE, we present the following theorem:

Theorem 2. *Given a conforming* cPRF *for a function class* $\mathcal{F}$*, the* RABE *designed in Section* 4.2 *is* IND − aCPA *secure with respect to* $\mathcal{F}$ *and under the hardness of* $\mathsf{DLWE}_{q,n,m,\chi}$.

Proof. The proof of Theorem 2 is similar to that of [17, Lemma 2]. It can be done by a sequence of hybrids; the only difference is that we need to take into account the way the challenger C should answer to updated key material queries from the adversary $\mathcal{A}$. We will proceed by considering a sequence of hybrids:

Hybrid 0: This game is the adaptive game described in Section 2.4.
Hybrid 1: This hybrid is the same as Hybrid 0 except the way that C answers the challenger query f^*. It computes $\mathsf{csk}_f \leftarrow \mathsf{F.Constrain}_\sigma(f^*)$ instead of $\mathsf{csk}_f \leftarrow \mathsf{P.KeySim}\sigma(f^*)$. Therefore, we have $\mathsf{csk}_f = U_\sigma \to_f(\sigma)$. However, based on a reduction to the key simulation game of conforming cPRF, it was shown in [17, Proof of Lemma 2] that the advantage for distinguishing Hybrid 0 and Hybrid 1 corresponds to the advantage of an adversary $\mathcal{A}_\mathsf{F}$ in the key simulation game of conforming cPRF. That means that Hybrid 0 and Hybrid 1 are indistinguishable.
Hybrid 2: The difference between Hybrid 1 and Hybrid 2 is the way that the challenger C generates the matrix $\mathbf{A}$. It computes the matrix $\mathbf{A}$ as follows: $\mathbf{A} = \mathbf{B}\mathbf{R} + \sigma \otimes \mathbf{G}$, where $\mathbf{R} \xleftarrow{\$} \{0,1\}^{m' \times m_0\lambda}$. From Proposition 1, $\mathbf{B}$ is statistically close to uniform, and therefore, from the extended leftover lemma, Hybrid 1 and Hybrid 2 are indistinguishable.
Hybrid 3: This hybrid is similar to Hybrid 2 but we change how the challenger computes $\boldsymbol{u}_1$ during the challenge query f^*. However, $\boldsymbol{u}_0$ and $\boldsymbol{u}_2$ are computed as described in the scheme. Indeed, the challenger computes $\boldsymbol{u}_1$ as $\boldsymbol{u}_1 = \boldsymbol{u}_0\mathbf{R}\widehat{\mathbf{H}}_{\mathsf{msk}\to\boldsymbol{r}_x} + \boldsymbol{e}_1$, where $\widehat{\mathbf{H}}_{\mathsf{msk}\to\boldsymbol{r}_x} \leftarrow \mathsf{EvalFX}(U_{\sigma\to\boldsymbol{x}}, \sigma, \mathbf{A})$, $\mathbf{A}_f - \mathsf{csk}_f \otimes \mathbf{G} = \mathbf{B}\mathbf{R}\widehat{\mathbf{H}}_{\mathsf{msk}\to\boldsymbol{r}_x}$ and $\boldsymbol{u}_0 = \boldsymbol{s}\mathbf{B} + \boldsymbol{e}_0$ with $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\boldsymbol{e}_0 \xleftarrow{\$} \chi^{m'}$, $\boldsymbol{e}_1 \xleftarrow{\$} \tilde{\chi}^{m\ell}$. We then have

$$\boldsymbol{u}_1 = \boldsymbol{u}_0\mathbf{R}\widehat{\mathbf{H}}_{\mathsf{msk}\to\boldsymbol{r}_x} + \boldsymbol{e}_1 = \boldsymbol{s}\mathbf{B}\mathbf{R}\widehat{\mathbf{H}}_{\mathsf{msk}\to\boldsymbol{r}_x} + \boldsymbol{e}_0\mathbf{R}\widehat{\mathbf{H}}_{\mathsf{msk}\to\boldsymbol{r}_x} + \boldsymbol{e}_1$$

and $B' = ||\mathbf{R}\widehat{\mathbf{H}}_{\mathsf{msk}\to \boldsymbol{r}_x}||_\infty \le (m_0 + m')\lambda||\boldsymbol{e}_0||_\infty||\mathbf{R}||_\infty||\widehat{\mathbf{H}}_{\mathsf{msk}\to \boldsymbol{r}_x}||_\infty \le (m_0 + m')\lambda B(2m_0)^{d_{con}}$, where d_{con} is the depth of $U_{\sigma\to f}$. Since $\tilde{\chi}$ is B'-swallowing, the difference between Hybrid 2 and Hybrid 3 is statistically indistinguishable.
Hybrid 4: The difference between this hybrid and Hybrid 3 is the way that the challenger answers to key queries. Indeed, if $(\mathsf{id}, \boldsymbol{x})$ is submitted, C fixes $\boldsymbol{r}_x \leftarrow \mathsf{P.Eval}_\sigma(\boldsymbol{x})$ and $\widehat{\mathbf{H}}_{\mathsf{msk}\to \boldsymbol{r}_x} \leftarrow \mathsf{EvalFX}(U_{\sigma\to \boldsymbol{x}}, \sigma, \mathbf{A})$. It also samples $\boldsymbol{y}_{\mathsf{id}} \xleftarrow{\$} \mathbb{Z}_q^n$. Due to the fact that $I_{\boldsymbol{r}_x} = 1$, it was shown in [17] that

$$[\mathbf{B}||\mathbf{A}_{\boldsymbol{x},\boldsymbol{r}_x}] = [\mathbf{B}||\mathbf{B}\mathbf{R}\widehat{\mathbf{H}}_{\mathsf{msk}\to \boldsymbol{r}_x}\widehat{\mathbf{H}}_{\boldsymbol{r}_x,\boldsymbol{r}_x} + \mathbf{G}] \quad \text{with } \widehat{\mathbf{H}}_{\boldsymbol{r}_x,\boldsymbol{r}_x} = \mathsf{EvalFX}(I_{\boldsymbol{r}_x}, \boldsymbol{r}_x, \mathbf{A}_x).$$

We now have

$$\begin{aligned}&||\mathbf{R}\widehat{\mathbf{H}}_{\mathsf{msk}\to \boldsymbol{r}_x}\widehat{\mathbf{H}}_{\boldsymbol{r}_x,\boldsymbol{r}_x}||_\infty \\ &\quad\le\ m_0^2 k\lambda||\mathbf{R}||_\infty||\mathbf{H}_{\mathsf{msk}\to \boldsymbol{r}_x}||_\infty||\mathbf{H}_{\boldsymbol{r}_x,\boldsymbol{r}_x}||_\infty \le m_0^2 k\lambda(2m_0)^{d+1}\end{aligned}$$

According to Proposition 1, given $\mathbf{B}$ and $\mathbf{R}\widehat{\mathbf{H}}_{\mathsf{msk}\to \boldsymbol{r}_x}\widehat{\mathbf{H}}_{\boldsymbol{r}_x,\boldsymbol{r}_x}$ the challenger can efficiently compute $[\mathbf{B}||\mathbf{A}_{\boldsymbol{x},\boldsymbol{r}_x}]_{\tau'}^{-1}$ for some $\tau' = O(k, \lambda, (2m_0)^{d+3})$. Thus, if $\tau \ge \tau'$, the challenger can sample $[\mathbf{B}||\mathbf{A}_{\boldsymbol{x},\boldsymbol{r}_x}]_{\tau'}^{-1}(\boldsymbol{y}_{\mathsf{id}} - (q-1)\boldsymbol{y}')$ without using the trapdoor $\mathbf{B}_{\tau_0}^{-1}$, where $\boldsymbol{y}' = \boldsymbol{y} - \boldsymbol{y}_{\mathsf{id}}$.

Hybrid 5: This game is similar to the previous game with the only difference being in the computing of the updated key material ku_T. Indeed, for all $\theta \in \mathsf{Node}_\mathsf{T}$, instead of computing $\boldsymbol{e}_{\theta,\mathsf{T}} = \mathbf{B}_{\tau_0}^{-1}(\mathcal{H}(\mathbf{A}, \mathbf{B}, q, \theta, \mathsf{T}))$, the challenger C first randomly chooses $\bar{\boldsymbol{e}}_{\theta,\mathsf{T}} \xleftarrow{\$} \chi^{m'}$, then computes $\boldsymbol{y}_{\theta,\mathsf{T}} = \mathbf{B}\bar{\boldsymbol{e}}_{\theta,\mathsf{T}}$, and finally set $\boldsymbol{e}_{\theta,\mathsf{T}} = \bar{\boldsymbol{e}}_{\theta,\mathsf{T}}$. It stores $(\theta, \boldsymbol{y}_{\theta,\mathsf{T}}, \boldsymbol{e}_{\theta,\mathsf{T}}, \mathsf{T})$ and configures an RO such that when the adversary queries with input (θ, T) for $\mathcal{H}(\mathbf{A}, \mathbf{B}, q, \theta, \mathsf{T})$, the challenger C returns $\boldsymbol{u}_{\theta,\mathsf{T}}$. Therefore, we can see that with a well-chosen set of parameters the change between Hybrid 4 and Hybrid 5 is statistically indistinguishable.
Hybrid 6: This game is similar to the previous game with the only difference being in the way that the adversary chooses the matrix $\mathbf{B}$. It chooses the matrix $\mathbf{B}$ randomly from $\mathbb{Z}_q^{n\times m'}$ instead of using the trapdoor generation algorithm. According to Proposition 1, this change between Hybrid 5 and Hybrid 6 is statistically indistinguishable.
Hybrid 7: The difference between Game 6 and Game 7 is in the way that the challenger computes ct* in the challenge query. Indeed, in the computation of ct*, the challenger chooses randomly $\boldsymbol{u}_0 \xleftarrow{\$} \mathbb{Z}_q^{m'}$ and $\boldsymbol{u}_2 \xleftarrow{\$} \mathbb{Z}_q$. We can see that this change is computationally indistinguishable under the $\mathsf{DLWE}_{q,n,m,\chi}$ assumption. Therefore, from this step, we can see that the challenger is able to hide completely b, and then, the adversary $\mathcal{A}$ has no advantage. □

In Table 1, we make a brief comparison of our scheme with that of the RABE schemes from previous studies [38,39], noting that the underlying hard problems that these three schemes rely on are considered quantum secure. Our comparison is based on the sizes of master secret key msk, master public key mpk, secret key sk, and the security model. In terms of access policy, similar to the proposed scheme, the scheme of Dong et al. and Luo et al. used boolean circuit as the access policy [39,40]. However, their schemes are key policy attribute-based schemes instead of ciphertext policy attribute-based schemes. In Dong et al.'s scheme [38], an arithmetic circuit is used as the access policy. In the scheme of Huang et al. and Yang et al. [41,42], linear secret sharing scheme (LSSS) access structures are used for access policy. We also note that the proposed scheme is adaptive secure compared to the other schemes in the table.

5 Lattice-based RPCH scheme

In this section, we describe our lattice-based RPCH. Its design follows Xu et al.'s [10] construction, and hence, it is a combination of our RABE and CHET schemes. Since we will use the modified version of the trapdoor generator algorithm, the message space in our underlying RABE will be $\{0, 1\}^\mu$. Therefore, we will consider as a security assumption the version of the decisional learning with errors problem for which the secret key is a

Table 1: Comparison of the proposed scheme with relevant schemes

Scheme	Model	mpk	msk	sk	Sec. Model	PQC
LAWWC23 [40]	KP ABE	$n(m\ell + m + \omega\log(n))\log(q)$	$m^2\log(q)$	$2m\omega\log(n)\log(q)$	Selective	Yes
HGL23 [41]	CP ABE	$(2\ell + 3)mn\log(q)$	$mn\log(q)$	$2mn(\log(N) + \ell + 2)\log(q)$	Selective	Yes
YWDFLW20 [42]	CP ABE	$n(5m + \ell + 1)\log(q)$	$m^2\log(q)$	$2m\log(\ell)\log(q)$	Selective	Yes
KMT [43]	IBE	$6nm\log(q)$	$2nm\log(q)$	$2mn\log(N)\log(q) + 4m^2\log(q)$	Selective	Yes
CLLWN12 [44]	IBE	$5nm\log(q)$	$nm\log(q)$	$2m\log(q)$	Selective	Yes
DZWC20 [39]	KP ABE	$(4\ell + 3)mn\log(q)$	$(4\ell + 2)mn\log(q)$	$2mn\log(q)$	Selective	yes
DHWLG21 [38]	KP ABE	$(\ell + 3)mn\log(q)$	$2mn\log(q)$	$2m^2\log(q)(\log(N) + 2)$	Selective	Yes
Our	CP ABE	$(\lambda + 4)mn\log(q)$	$\lambda + mn\log(q)$	$\lambda + \ell + (\ell + 6)m^2\log(q)$	Adaptive	Yes

matrix of $\mathbb{Z}_q^{m\times\mu}$ instead of a vector of $\mathbb{Z}_q^m$. We denote this version by $\mathsf{DLWE}_{q,n\times\mu,m,\chi}$. Note that when $\mu = 1$, it just becomes the main decisional learning with error problem $\mathsf{DLWE}_{q,n,m,\chi}$. It is also important to recall that Micciancio established in [45, Lemma 8] that there is a polynomial-time reduction from $\mathsf{DLWE}_{q,n,m,\chi}$ to $\mathsf{DLWE}_{q,n\times\mu,m,\chi}$. Moreover, if there is a PPT adversary $\mathcal{A}$ that can distinguish $\mathsf{DLWE}_{q,n\times\mu,m,\chi}$ with an advantage ε, it can then distinguish $\mathsf{DLWE}_{q,n,m,\chi}$ with an advantage ε/μ (see [45, Proof of Lemma 8]). Therefore, it is important to choose parameters to have a security level close to that of $\mathsf{DLWE}_{q,n,m,\chi}$.

In the design of our RPCH, μ is chosen to be equal to the security parameters λ. The key update KUpd and the revocation list Rev algorithms correspond to that of our RABE. However, the secret key generation and the setup algorithms are designed by taking into account our CH scheme.

$\mathsf{Setup}_{\mathsf{RPCH}}$:
Input: Security parameter λ and number $N = 2^{k_{\mathsf{id}}} = |\mathcal{ID}|$ of users in the system.
Output: $\mathsf{mpk}_{\mathsf{RPCH}}$ and $\mathsf{msk}_{\mathsf{RPCH}}$
1. Compute $(\mathsf{msk}_{\mathsf{RABE}}, \mathsf{mpk}_{\mathsf{RABE}}) \leftarrow \mathsf{Setup}_{\mathsf{RABE}}(1^\lambda, N)$ where $N = 2^{k_{\mathsf{id}}}$.
2. Compute $\mathsf{PP}_{\mathsf{CHET}} \leftarrow \mathsf{Setup}_{\mathsf{CHET}}(1^\lambda)$
3. Generate a pair $(\mathsf{sk}_{\mathsf{CHET}}, \mathsf{pk}_{\mathsf{CHET}}) \leftarrow \mathsf{KeyGen}_{\mathsf{CHET}}(n_1, q_1, \mathcal{D}_{\sigma_1})$ of public and secret keys of our chameleon hash $\mathsf{sk}_{\mathsf{CHET}} = \mathbf{A}_{\sigma_1}^{-1}$ and $\mathsf{pk}_{\mathsf{CHET}} = \mathbf{A}_1 \in \mathbb{Z}_{q_1}^{n_1\times m_1}$.
4. Set $\mathsf{mpk}_{\mathsf{RPCH}} = (\mathsf{mpk}_{\mathsf{RABE}}, \mathsf{pk}_{\mathsf{CHET}})$ and $\mathsf{msk}_{\mathsf{RPCH}} = (\mathsf{msk}_{\mathsf{RABE}}, \mathsf{sk}_{\mathsf{CHET}})$

$\mathsf{SKGen}_{\mathsf{RPCH}}$:
Input: Master secret key $\mathsf{msk}_{\mathsf{RPCH}}$, state st, user identity id with its attribute $\boldsymbol{x}$
Output: Secret key $\mathsf{sk}_{\mathsf{id}}$ for a user with identity id and attribute $\boldsymbol{x}$.
1. Parse $\mathsf{msk}_{\mathsf{RPCH}}$ as $(\mathsf{msk}_{\mathsf{RABE}}, \mathbf{A}_{\sigma_1}^{-1})$.
2. Compute $\mathsf{sk}_{\mathsf{RABE,id}} \leftarrow \mathsf{SKGen}_{\mathsf{RABE}}(\mathsf{msk}_{\mathsf{RABE}}, \mathsf{st}, \mathsf{id}, \boldsymbol{x})$.
3. Set $\mathsf{sk}_{\mathsf{id}} \leftarrow (\mathsf{sk}_{\mathsf{RABE,id}}, \mathbf{A}_{\sigma_1}^{-1})$.
4. Return $\mathsf{sk}_{\mathsf{id}}$.

In the design of our hash algorithm Hash, we use the Fujisaki–Okamoto transformation to our underlying RABE. For that, we use a symmetric encryption scheme with security parameter λ. We choose λ as the length of plaintext in our RABE. Let us denote the symmetric encryption scheme by SymEnc. We also use a hash function $\mathcal{H}_2 : \{0,1\}^\lambda \times \{0,1\}^\lambda \longrightarrow \{0,1\}^\lambda$.

$\mathsf{Hash}_{\mathsf{RPCH}}$:
Input: Master public key $\mathsf{mpk}_{\mathsf{RPCH}}$, message to hash $\boldsymbol{m}$.
Output: A pair $(\boldsymbol{h}, \boldsymbol{r})$ of a hash $\boldsymbol{h}$ and a random value $\boldsymbol{r}$.
1. Parse $\mathsf{mpk}_{\mathsf{RPCH}}$ as $(\mathsf{mpk}_{\mathsf{RABE}}, \mathsf{pk}_{\mathsf{CHET}})$
2. Compute $(\boldsymbol{h}', \boldsymbol{r}, \mathsf{etd}) = \mathsf{Hash}_{\mathsf{CHET}}(\mathsf{pk}_{\mathsf{CHET}}, \boldsymbol{m})$
3. Generate $\mathsf{seed}_{\mathsf{Sym}} \xleftarrow{\$} \{0,1\}^\lambda$
4. Generate the symmetric key $\boldsymbol{k} \leftarrow \mathsf{RO}_{\mathsf{Sym}}(\mathsf{seed}_{\mathsf{Sym}})$.
5. Compute $\mathsf{ct}_s \leftarrow \mathsf{SymEnc}(\boldsymbol{k}, \mathsf{etd})$ and $\mathsf{ct} \leftarrow \mathsf{Enc}_{\mathsf{RABE}}(\mathsf{mpk}_{\mathsf{RABE}}, \mathsf{seed}_{\mathsf{Sym}}, f, \mathsf{T})$
6. Set $\boldsymbol{c} = (\mathsf{ct}_s, \mathsf{ct}, \mathsf{ct}_{hash})$, where $\mathsf{ct}_{hash} = \mathcal{H}_2(\mathsf{seed}_{\mathsf{Sym}}, \mathsf{etd})$
7. Return $\boldsymbol{h} = (\boldsymbol{h}', \boldsymbol{c})$ and $\boldsymbol{r}$, where $\boldsymbol{h}' = (\boldsymbol{h}_1, \boldsymbol{h}_2, \mathsf{pk}'_{\mathsf{CHET}})$ and $\boldsymbol{r} = (\boldsymbol{r}_1, \boldsymbol{r}_2)$.

$\mathsf{Verif}_{\mathsf{RPCH}}$:
Input: Master public key mpk, message $\boldsymbol{m}$, hash $\boldsymbol{h}$ and random value $\boldsymbol{r}$.
Output: $b \in \{0,1\}$
1. Parse mpk as $(\mathsf{mpk}_{\mathsf{RABE}}, \mathsf{pk}_{\mathsf{CHET}})$ and $\boldsymbol{h}$ as $(\boldsymbol{h}', \boldsymbol{c})$
2. If $\mathsf{Verif}_{\mathsf{CHET}}(\mathsf{pk}_{\mathsf{CHET}}, \boldsymbol{m}, \boldsymbol{h}', \boldsymbol{r}) = 1$
   Return 1.
3. Return 0.

$\underline{\mathsf{DKGen}_{\mathsf{RPCH}}}$:
Input: A secret key $\mathsf{sk}_{\mathsf{id}}$, and the updated key material ku_T.
Output: A decryption key $\mathsf{dk}_{\mathsf{id},T}$.
1. Parse $\mathsf{sk}_{\mathsf{id}}$ as $(\mathsf{sk}_{\mathsf{RABE,id}}, \mathsf{sk}_{\mathsf{CHET}})$
2. Compute $\mathsf{dk}_{\mathsf{RABE,id},T} = \mathsf{DKGen}_{\mathsf{RABE}}(\mathsf{sk}_{\mathsf{RABE,id}}, \mathsf{ku}_T)$
3. Set $\mathsf{dk}_{\mathsf{id},T} = (\mathsf{dk}_{\mathsf{RABE,id},T}, \mathsf{sk}_{\mathsf{CHET}})$

$\underline{\mathsf{Adapt}_{\mathsf{RPCH}}}$:
Input: Decoding key $\mathsf{dk}_{\mathsf{id},T}$, master secret key msk, master public key mpk, original message $\boldsymbol{m}$, new message $\boldsymbol{m}'$, hash $\boldsymbol{h}$, and random value $\boldsymbol{r}$.
Output: A new random value $\tilde{\boldsymbol{r}}$
1. Parse mpk as $(\mathsf{mpk}_{\mathsf{RABE}}, \mathsf{pk}_{\mathsf{CHET}})$ and $\boldsymbol{h}$ as $(\boldsymbol{h}', \boldsymbol{c})$.
2. If $\mathsf{Verif}_{\mathsf{CHET}}(\mathsf{pk}_{\mathsf{CHET}}, \boldsymbol{m}, \boldsymbol{h}', \boldsymbol{r}) = 0$:
   Return $\perp$.
3. Parse $\boldsymbol{c}$ as $(\mathsf{ct}_s, \mathsf{ct}, \mathsf{ct}_{hash})$.
4. Parse $\mathsf{dk}_{\mathsf{id},T}$ as $(\mathsf{dk}_{\mathsf{RABE,id},T}, \mathsf{sk}_{\mathsf{CHET}})$.
5. Compute $\mathsf{seed}_{\mathsf{Sym}} \leftarrow \mathsf{Dec}_{\mathsf{RABE}}(\mathsf{dk}_{\mathsf{RABE,id},T}, \mathsf{ct})$.
6. Compute $\boldsymbol{k} \leftarrow \mathsf{RO}_{\mathsf{Sym}}(\mathsf{seed}_{\mathsf{Sym}})$.
7. Compute $\mathsf{etd} \leftarrow \mathsf{Dec}_{\mathsf{Sym}}(\boldsymbol{k}, \mathsf{ct}_s)$.
8. If $\mathcal{H}_2(\mathsf{seed}_{\mathsf{Sym}}, \mathsf{etd}) \neq \mathsf{ct}_{hash}$:
   Return $\perp$.
9. Compute $\tilde{\boldsymbol{r}} \leftarrow \mathsf{Adapt}_{\mathsf{CHET}}(\mathsf{sk}_{\mathsf{CHET}}, \mathsf{etd}, \boldsymbol{m}, \boldsymbol{m}', \boldsymbol{h}', \boldsymbol{r})$.
10. Return $\tilde{\boldsymbol{r}}$.

Note that the RPCH designed in this paper is based on an IND − aCPA secure RABE and a CHET, which is fully indistinguishable, strongly private collision-resistant, unique, and correct. Therefore, according to [10, Theorem 2], we have the following:

Theorem 3. *The proposed* RPCH *is fully indistinguishable, insider collision-resistant, unique, and correct.*

6 Conclusion

To our knowledge, this article presents the first-ever lattice-based RPCH. For its design, we have designed a lattice-based CHET scheme. Due to the fact that our RPCH scheme is based on the generic construction of Derler et al., which requires a RABE scheme, we have presented a new lattice-based RABE scheme. We have proven that the proposed RABE scheme is adaptively secured and compared it to recent and relevant schemes.

Acknowledgements: The majority of the work was done when the corresponding author was affiliated with the University of Waterloo as a Post-doctoral researcher. The authors would like to thank the anonymous reviewers for their comments on an earlier version of this article.

Funding information: This work was supported by Ripple Impact Fund/Silicon Valley Community Foundation (Grant 2018-188473).

Author contributions: All authors have accepted responsibility for the entire content of this manuscript and consented to its submission and publication to the journal, reviewed all the results, and approved the final version of the manuscript. JBK designed, performed the security proof of the proposed schemes and drafted the manuscript. MAH provided technical review and editing of the manuscript.

Conflict of interest: The authors state that there is no conflict of interest.

References

[1] Krawczyk H, Rabin T. Chameleon hashing and signatures; 1998. https://ia.cr/1998/010. Cryptology ePrint Archive, Report 1998/010.

[2] Camenisch J, Derler D, Krenn S, Pöhls HC, Samelin K, Slamanig D. Chameleon-Hashes with ephemeral trapdoors - and applications to invisible sanitizable signatures. In: Fehr S, editor. Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28–31, 2017, Proceedings, Part II. vol. 10175 of Lecture Notes in Computer Science. Springer; 2017. p. 152–82.

[3] Ateniese G, Chou DH, Medeiros Bd, Tsudik G. Sanitizable signatures. In: European Symposium on Research in Computer Security. Springer, Berlin, Heidelberg; 2005. p. 159–77.

[4] Even S, Goldreich O, Micali S. On-line/offline digital signatures. J Cryptol. 1996;9(1):35–67.

[5] Krawczyk H, Rabin T. Chameleon signatures. In: Proceedings of the Network and Distributed System Security Symposium, NDSS 2000, San Diego, California, USA. The Internet Society; 2000. p. 143–54.

[6] Shamir A, Tauman Y. Improved online/offline signature schemes. In: Annual International Cryptology Conference. Springer; 2001. p. 355–67.

[7] Chen X, Zhang F, Susilo W, Mu Y. Efficient generic on-line/offline signatures without key exposure. In: International Conference on Applied Cryptography and Network Security. Springer; 2007. p. 18–30.

[8] Ateniese G, Magri B, Venturi D, Andrade E. Redactable blockchain or rewriting history in Bitcoin and friends. In: 2017 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE; 2017. p. 111–26.

[9] Derler D, Samelin K, Slamanig D, Striecks C. Fine-grained and controlled rewriting in blockchains: Chameleon-Hashing gone attribute-based. In: 26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24–27, 2019. The Internet Society; 2019.

[10] Xu S, Ning J, Ma J, Xu G, Yuan J, Deng RH. Revocable policy-based Chameleon Hash. In: European Symposium on Research in Computer Security. Springer International Publishing; 2021. p. 327–47.

[11] Cash D, Hofheinz D, Kiltz E, Peikert C. Bonsai trees, or how to delegate a lattice basis. J Cryptol. 2012;25(4):601–39.

[12] Ateniese G, de Medeiros B. Identity-based chameleon hash and applications. In: Financial Cryptography: 8th International Conference, FC 2004, Key West, FL, USA, February 9–12, 2004. Revised Papers 8. Springer; 2004.p. 164–80.

[13] Bao F, Deng RH, Ding X, Lai J, Zhao Y. Hierarchical identity-based chameleon hash and its applications. In: Applied Cryptography and Network Security: 9th International Conference, ACNS 2011, Nerja, Spain, June 7–10, 2011. Proceedings 9. Springer; 2011. p. 201–19.

[14] Chen X, Tian H, Zhang F, Ding Y. Comments and improvements on key-exposure free chameleon hashing based on factoring. In: International Conference on Information security and Cryptology. Springer; 2010. p. 415–26.

[15] Zhang F, Safavi-Naini R, Susilo W. ID-based chameleon hashes from bilinear pairings. Cryptology ePrint Archive. 2003.

[16] Brzuska C, Fischlin M, Freudenreich T, Lehmann A, Page M, Schelbert J, et al. Security of sanitizable signatures revisited. In: Public Key Cryptography-PKC 2009: 12th International Conference on Practice and Theory in Public Key Cryptography, Irvine, CA, USA, March 18–20, 2009. Proceedings 12. Springer; 2009. p. 317–36.

[17] Tsabary R. Fully secure attribute-based encryption for t-CNF from LWE. In: Annual International Cryptology Conference. Springer; 2019. p. 62–85.

[18] Xiong H, Huang X, Yang M, Wang L, Yu S. Unbounded and efficient revocable attribute-based encryption with adaptive security for cloud-assisted internet of things. IEEE Internet Things J. 2022;9(4):428–41.

[19] Micciancio D, Peikert C. Trapdoors for lattices: simpler, tighter, faster, smaller. In: Pointcheval D, Johansson T, editors. Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings. Vol. 7237 of Lecture Notes in Computer Science. Springer; 2012. p. 700–18.

[20] Gentry C, Peikert C, Vaikuntanathan V. Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the fortieth annual ACM symposium on Theory of computing; 2008. p. 197–206.

[21] Ajtai M. Generating hard instances of lattice problems. In: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing; 1996. p. 99–108.

[22] Alwen J, Peikert C. Generating shorter bases for hard random lattices. Theory Comput Syst. 2011;48(3):535–53.

[23] Agrawal S, Boneh D, Boyen X. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In: Annual Cryptology Conference. Springer; 2010. p. 98–115.

[24] Laarhoven T, van De Pol J, De Weger B. Solving hard lattice problems and the security of lattice-based cryptosystems. Cryptology ePrint Archive. 2012.

[25] Albrecht MR, Deo A. Large modulus Ring-LWE ≥ module-LWE. In: International Conference on the Theory and Application of Cryptology and Information Security. Springer; 2017. p. 267–96.

[26] Koo Z, Lee Y, Lee JW, No JS, Kim YS. Improved reduction between SIS problems over structured lattices. IEEE Access. 2021;9:157083–92.
[27] Micciancio D, Peikert C. Hardness of SIS and LWE with small parameters. In: Annual Cryptology Conference. Springer; 2013. p. 21–39.
[28] Regev O. On lattices, learning with errors, random linear codes, and cryptography. J ACM (JACM). 2009;56(6):1–40.
[29] Lyubashevsky V. Lattice signatures without trapdoors. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer; 2012. p. 738–55.
[30] Goldreich O, Goldwasser S, Micali S. How to construct random functions. J ACM. 1986;33(4):792–807.
[31] Boneh D, Waters B. Constrained pseudorandom functions and their applications. In: International Conference on the Theory and Application of Cryptology and Information Security. Springer; 2013. p. 280–300.
[32] Peter N, Tsabary R, Wee H. One-one constrained pseudorandom functions. In: 1st Conference on Information-theoretic Cryptography (ITC 2020). Schloss Dagstuhl-Leibniz-Zentrum für Informatik; 2020.
[33] Sahai A, Waters B. Fuzzy identity-based encryption. In: Cramer R, editor. Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22–26, 2005, Proceedings. vol. 3494 of Lecture Notes in Computer Science. Springer; 2005. p. 457–73.
[34] Goyal V, Pandey O, Sahai A, Waters B. Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security; 2006. p. 89–98.
[35] Derler D, Samelin K, Slamanig D. Bringing order to Chaos: the case of collision-resistant Chameleon-Hashes. In: Kiayias A, Kohlweiss M, Wallden P, Zikas V, editors. Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4–7, 2020, Proceedings, Part I. vol. 12110 of Lecture Notes in Computer Science. Springer; 2020. p. 462–92.
[36] Derler D, Krenn S, Samelin K, Slamanig D. Fully collision-resistant Chameleon-Hashes from simpler and post-quantum assumptions. In: International Conference on Security and Cryptography for Networks. Springer; 2020. p. 427–47.
[37] Naor D, Naor M, Lotspiech J. Revocation and tracing schemes for stateless receivers. In: Annual International Cryptology Conference. Springer; 2001. p. 41–62.
[38] Dong X, Hu Y, Wang B, Liu M, Gao W. Lattice-based revocable attribute-based encryption with decryption key exposure resistance. IET inform security. 2021;15(6):428–41.
[39] Dong X, Zhang Y, Wang B, Chen J. Server-aided revocable attribute-based encryption from lattices. Security Commun Netw. 2020;2020:1460531.
[40] Luo F, Al-Kuwari S, Wang H, Wang F, Chen K. Revocable attribute-based encryption from standard lattices. Comput Standards Interfaces. 2023;84:103698.
[41] Huang B, Gao J, Li X. Efficient lattice-based revocable attribute-based encryption against decryption key exposure for cloud file sharing. J Cloud Comput. 2023;12(1):1–15.
[42] Yang K, Wu G, Dong C, Fu X, Li F, Wu T. Attribute based encryption with efficient revocation from lattices. Int J Netw Secur. 2020;22(1):161–70.
[43] Katsumata S, Matsuda T, Takayasu A. Lattice-based revocable (hierarchical) IBE with decryption key exposure resistance. Theoretic Comput Sci. 2020;809:103–36.
[44] Chen J, Lim HW, Ling S, Wang H, Nguyen K. Revocable identity-based encryption from lattices. In: Information Security and Privacy: 17th Australasian Conference, ACISP 2012, Wollongong, NSW, Australia, July 9–11, 2012. Proceedings 17. Springer; 2012. p. 390–403.
[45] Micciancio D. On the hardness of learning with errors with binary secrets. Theory Comput. 2018;14(1):1–17.