Efficient Key-Rotatable and Security-Updatable Homomorphic Encryption

Yoshinori Aono
NICT, Japan
aono@nict.go.jp

Takuya Hayashi
NICT, Japan
takuya.hayashi@nict.go.jp

Le Trieu Phong *
NICT, Japan
phong@nict.go.jp

Lihua Wang
NICT, Japan
wlh@nict.go.jp

ABSTRACT

In this paper we presents the notion of key-rotatable and security-updatable homomorphic encryption (KR-SU-HE) scheme, which is a class of public-key homomorphic encryption in which the keys and the security of any ciphertext can be rotated and updated while still keeping the underlying plaintext intact and unrevealed. We formalise syntax and security notions for KR-SU-HE schemes and then build a concrete scheme based on the Learning With Errors assumption. We then perform testing implementation to show that our proposed scheme is efficiently practical.

Keywords

Homomorphic encryption, Learning With Errors, key rotation, security update.

1. INTRODUCTION

1.1 Background

Key rotation is an important practice for cryptosystems. Succinctly, it is the change of an old key by a new one, while keeping the plaintexts intact.

Key rotation is specified in several security standards. Indeed, on the industrial side, it is required by Payment Card Industry Data Security Standard [22], and is considered as an obligation by Open Web Application Security Project [21], concretely specifying that

"key rotation is a must as all good keys do come to an end either through expiration or revocation. So a developer will have to deal with rotating keys at some point – better to have a system in place now rather than scrambling later."

On the federal side, it is recommended by NIST [19] via the concept of cryptoperiods of keys, namely the time spans during which they are used. As suggested in [19], cryptoperiods are in the order of 1-2 years depending on the considered primitives. Therefore, after a few years, the old key should be changed by a new one.

In this paper, we focus on the problem of key rotation and its close variant called *security update* over public key homomorphic encryption, as it is particularly relevant to secure cloud computing.

1.2 Problem formalisation and naïve solution

We first formalise the problem of key rotation and security update, and then present a naïve solution.

Problem formalisation. Let us consider a public key homomorphic encryption scheme. Suppose encryption under (pk_1, sk_1) has n_1-bit security, while that under (pk_2, sk_2) has n_2-bit security. The question is how to turn an n_1-bit security ciphertext into a ciphertext having n_2-bit security while keeping the underlying plaintext the same. Moreover, homomorphic operations such as additions and multiplications can be still performed after the transformation.

Two possibly interesting cases of the above problem are as follows:

- **Key rotation**: when $n_1 = n_2$, the above is the known problem in practice as the process of re-keying encrypted data from an old key to a new one.
- **Security update**: when $n_1 < n_2$, the problem can be described succinctly as turning security-weakened ciphertexts and the related secret key into ones with higher security assurance.

The naïve solution: decrypt-then-encrypt. It is possible to solve the problem by the decrypt-then-encrypt method. Specifically, use the secret key sk_1 related to pk_1 to decrypt the old ciphertext, and then encrypt the obtained plaintext under pk_2. This approach has potential shortcomings: (1) the plaintext is recovered, and (2) the secret sk_1 must be used for decryption. Concretely, if the task of key rotation is done by an outsourced server in cloud computing, then either by (1) or (2), the server learns the data or the secret key.

Our goal. In this paper, we are interested in developing an efficient solution without the above shortcomings of the naïve solution.

*Corresponding author.

SCC'17, April 2, 2017, Abu Dhabi, United Arab Emirates.

DOI: http://dx.doi.org/10.1145/3055259.3055260

Table 1: Usages of the dimension switching technique.

Dimension switching	Exploited in	Main purpose
(high $\rightarrow$ low) $n_2 < n_1$	[9, 11]	efficiency in FHE
(equal) $n_2 = n_1$	[1, 12, 20]	PRE, obfuscation
(equal, low $\rightarrow$ high) $n_2 \geq n_1$	**this paper**	key rotation, security update

2. OUR CONTRIBUTIONS

We propose a primitive called *key-rotatable and security-updatable homomorphic encryption* (KR-SU-HE) and formalise its security notions. We then construct a concrete scheme based on the Learning With Errors (LWE) assumption. We finally show that our scheme is practical by implementation. Details are given below.

Conceptual proposal. We develop a solution to the above problem where, given public/secret key pair (pk_1, sk_1) of n_1-bit security and (pk_2, sk_2) of n_2-bit security,

- An outsourced server is given an update key $uk_{n_1 \rightarrow n_2}$, which is computed from sk_1 and sk_2, and yet $uk_{n_1 \rightarrow n_2}$ is computationally random from the server's view.
- The server uses $uk_{n_1 \rightarrow n_2}$ to update ciphertexts corresponding to (pk_1, sk_1) into ciphertexts corresponding to (pk_2, sk_2), while underlying plaintexts are kept the same.
- Only sk_2 is needed in decryption of updated ciphertexts.

We call a homomorphic encryption scheme having above transformation *key-rotatable and security-updatable homomorphic encryption*. This type of homomorphic encryption needs new security definitions, because we want to ensure that transformed ciphertexts not only can be decrypted by sk_2, but also have n_2-bit security. We therefore formalise the notion of d-CPA security, where d is an integer specifying the number of updates so far. For example, if one goes from 80- to 128- then to 256-bit security, then $d = 2$ (updates of security). Also, for key rotation purpose, if one periodically switches 10 times among keys of identical bit security, then $d = 10$.

A concrete scheme. We then build a concrete and efficient KR-SU-HE scheme based on the LWE assumption. The tool for our construction is described below.

On a bird's-eye view, increasing security is possible via an enlargement of dimension in the LWE assumption, where higher dimension ensures more security. The switch is exactly the dimension-switching technique original used in fully homomorphic encryption (FHE) [11]. Historically, the paper [11] only considered dimension-*reduction*, namely $n_2 < n_1$ in our notation, for efficiency issues in FHE. Then subsequent paper [9], while notified that the technique works for arbitrary dimensions n_1 and n_2, only made use of the case $n_2 < n_1$ as in [11]. The case $n_1 = n_2$ is considered in [1, 20] for proxy re-encryption and in [12] for obfuscation. To our best knowledge, we are the first to exploit the case $n_1 < n_2$ for security enhancement.

Dimension switching however provides us only with correctness in dimension n_2, not yet security in that dimension. We solve that issue by a re-randomisation in the update process. Details are given in Section 4.

Testing implementation. Taking a concrete parameter set, we implement our proposed scheme to show that it is efficient. Details are given in Section 4.3, including the timings for all algorithms in our proposed scheme.

3. PRELIMINARIES

Let $\mathbb{Z}_{(0,s)}$ be the discrete Gaussian distribution over the integers $\mathbb{Z}$, with mean 0 and deviation $s > 0$. The mark $\xleftarrow{g}$ is for "sampling at random from a discrete Gaussian" set, so that $x \xleftarrow{g} \mathbb{Z}_{(0,s)}$ means x appears with probability proportional to $\exp(-\pi x^2/s^2)$. Below, $\xleftarrow{\$}$ means "sampling uniformly at random". Also, $\mathbb{Z}_q \subset (-q/2, q/2]$ is the set of integers centered modulus q.

3.1 Learning with Errors (LWE)

Related to the decision LWE assumption LWE(n, s, q), where n, s, q depend on the security parameter, consider matrix $A \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$, vectors $r \xleftarrow{\$} \mathbb{Z}_q^{m \times 1}$, $x \xleftarrow{g} \mathbb{Z}_{(0,s)}^{n \times 1}, e \xleftarrow{g} \mathbb{Z}_{(0,s)}^{m \times 1}$. Then vector $Ax + e$ is computed over $\mathbb{Z}_q$. Define the following advantage of a poly-time probabilistic algorithm $\mathcal{D}$:

$$\mathbf{Adv}_{\mathcal{D}}^{\mathrm{LWE}(n,s,q)}(\lambda) = \Big| \Pr[\mathcal{D}(A, Ax + e) \rightarrow 1] - \Pr[\mathcal{D}(A, r) \rightarrow 1] \Big|.$$

The LWE assumption asserts that $\mathbf{Adv}_{\mathcal{D}}^{\mathrm{LWE}(n,s,q)}(\lambda)$ is negligible as a function of λ. Also note that, originally, x is chosen randomly from $\mathbb{Z}_q^{n \times 1}$ in [23]. However, as showed in [4, 18], one can take $x \xleftarrow{g} \mathbb{Z}_{(0,s)}^{n \times 1}$ without weakening the assumption as we do here.

LWE dimension and security level. The length of secret vector x, namely n, is denoted as the LWE dimension. For the same deviation s and modulus q, the hardness of LWE assumption increases with its dimension n as showed in [10], which is the important fact we will exploit in designing our KR-SU-HE schemes. Therefore, "LWE dimension" and "security level" are sometimes used interchangeably in this manuscript.

3.2 Homomorphic encryption, key rotation, security update

Definition 1 (PHE). *Public key homomorphic encryption (PHE) schemes consist of the following (possibly probabilistic) poly-time algorithms.*

- $\mathsf{ParamGen}(1^\lambda) \rightarrow pp$: *$\lambda$ is the security parameter and the public parameter pp is implicitly fed in following algorithms.*
- $\mathsf{KeyGen}(1^\lambda) \rightarrow (pk, sk)$: *$pk$ is the public key, while sk is the secret key.*
- $\mathsf{Enc}(pk, m) \rightarrow c$: *probabilistic encryption algorithm produces c, the ciphertext of message m.*

- $\mathsf{Dec}(sk, c) \to m$: *decryption algorithm returns message m encrypted in c.*
- $\mathsf{Add}(c, c')$, $\mathsf{AddM}(c, c')$: *In* Add*, for ciphertexts c and c', the output is the encryption of plaintext addition c_{add}. Similarly,* AddM *adds two ciphertexts each was obtained by one multiplication (using* Mul *below).*
- $\mathsf{Mul}(pp, c, c')$: *for ciphertexts c and c', the output is the encryption of plaintext multiplication c_{mul}.*
- $\mathsf{DecA}(sk, c_{\text{add}})$: *decrypting c_{add} to obtain an addition of plaintexts.*
- $\mathsf{DecM}(sk, c_{\text{mul}})$: *decrypting c_{mul} to obtain a multiplication of plaintexts.*

Definition 2 (KR-SU-HE). *A scheme as in Definition 1 is called a* key-rotatable and security-updatable homomorphic encryption *scheme if it has additional algorithms as follows.*

- $\mathsf{UKGen}(pp, pk_1, sk_1, pk_2, sk_2)$: *generating an update key $uk_{1\to 2}$ from the public/secret key pairs (pk_1, sk_1) (old) and (pk_2, sk_2) (new).*
- $\mathsf{Update}(pp, c, uk_{1\to 2}, pk_2)$: *outputting a new ciphertext c_{new} from an old c.*

with the following requirements:

a) Correctness of updated ciphertexts: *succinctly,*

$$\mathsf{Dec}(sk_2, c_{\text{new}}) = \mathsf{Dec}(sk_1, c)$$

with overwhelming probability, namely the correctness is not affected by key rotation and security update.

b) Homomorphisms: *are preserved as long as the ciphertexts in operations are under the same public key (either pk_1 or pk_2), regardless of whether they are (1) directly formed by* Enc*, or (2) indirectly formed by* Update.

Following security notion defines d-CPA security, namely the CPA security of ciphertexts obtained via 1 time of original encryption and subsequent d key rotations or security updates.

Definition 3 (d-CPA security). *With reference to a KR-SU-HE scheme as in Definition 2, consider the following game between an adversary $\mathcal{A}$ and a challenger:*

1. **Setup:** *the challenger creates following keys (pk_0, sk_0), (pk_1, sk_1), ..., (pk_{d-1}, sk_{d-1}) of security levels $n_0, \ldots, n_{d-1}$ respectively, and (pk_d, sk_d) of level n_d. Then (sk_i, pk_i) $(\forall\ 0 \le i \le d-1)$ and pk_d are given to $\mathcal{A}$. The key sk_d is kept secret to $\mathcal{A}$.*
2. **Update key queries:** *$\mathcal{A}$ can ask for many update keys from any (pk_i, sk_i) for $0 \le i \le d-1$ to (pk_d, sk_d)[1]. To each query, the challenger returns a freshly generated update key $uk_{n_i\to n_d}$.*
3. **Challenge:** *$\mathcal{A}$ chooses*
   - *two plaintexts m_0, m_1 of the same length, and*
   - *a series of update keys $uk^*_{n_0\to n_1}, \ldots, uk^*_{n_{d-1}\to n_d}$ of predetermined sizes.*

   $\mathcal{A}$ then submits them to the challenger, who in turn takes $b \in \{0,1\}$ randomly, computes C_0^* as the encryption of m_b under pk_0, namely $C_0^* = \mathsf{Enc}(pk_0, m_b)$. Then applying the update algorithm as follows:

$$\begin{aligned} C_1^* &= \mathsf{Update}(C_0^*, uk^*_{n_0\to n_1}, pk_1), \\ &\vdots \\ C_d^* &= \mathsf{Update}(C_{d-1}^*, uk^*_{n_{d-1}\to n_d}, pk_d). \end{aligned}$$

   The final ciphertext C_d^* is returned to $\mathcal{A}$.
4. **Additional update key queries:** *the same as step 2.*
5. *Finally, $\mathcal{A}$ returns a bit b' as a guess of the hidden bit b.*

A KR-SU-HE scheme is d-CPA-secure if the advantage

$$\mathbf{Adv}^{d-\text{cpa}}_{\mathcal{A}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|$$

is negligible as a function of λ.

Some comments on Definition 3 are in order. First, 0-CPA is exactly the standard CPA security under pk_0, since there is no key rotation or update. The fact that $\mathcal{A}$ gets secret keys $sk_0, \ldots, sk_{d-1}$ captures the extreme case in which keys in security levels $n_0, \ldots, n_{d-1}$ are insecure. This is generous to $\mathcal{A}$ since in practice we expect new keys are generated and ciphertexts are updated before old keys are considered insecure. Second, the fact $\mathcal{A}$ gets update keys in steps 2, 4 and can challenge with malformed update keys at step 3 envisages that the keys are publicly transmitted to and permanently put in outsourced servers and hence may be modified. This differs from security notions for both public key encryption (PKE) and proxy re-encryption (PRE) in which the adversary only chooses two messages.

Certainly we need also care about the CPA security of ciphertexts directly encrypted under the new public key, which is captured by following notion.

Definition 4 (CPA security). *With respect to a KR-SU-HE scheme as in Definition 2, consider the following game between an adversary $\mathcal{A}$ and a challenger:*

1. **Setup:** *the same as in Definition 3.*
2. **Update key queries:** *the same as in Definition 3.*
3. **Challenge:** *$\mathcal{A}$ chooses two plaintexts m_0, m_1 of the same length, then submits them to the challenger, who in turn takes $b \in \{0,1\}$ randomly and computes $C^* = \mathsf{Enc}(pk_d, m_b)$. The challenge ciphertext C^* is returned to $\mathcal{A}$.*
4. **Additional update key queries:** *the same as in Definition 3.*

A KR-SU-HE scheme is CPA-secure if

$$\mathbf{Adv}^{\text{cpa}}_{\mathcal{A}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|$$

is negligible in λ.

[1] $\mathcal{A}$ itself can generate update keys between lower dimensions $uk_{n_i\to n_j}$ for $0 \le i, j \le d-1$ since it has sk_i and sk_j.

3.3 Secure cloud computing with key rotation and security update

Using KR-SU-HE schemes, a client can store encrypted data on cloud server for computations using homomorphic properties. Moreover, the client can do key rotation for all ciphertexts, namely replacing an old key pair (pk_1, sk_1) by a new one (pk_2, sk_2) by:

1. The client holding (pk_1, sk_1) generates the new key pair (pk_2, sk_2), and then creates $uk_{1\to 2}$ via executing
$$\mathsf{UKGen}(pp, pk_1, sk_1, pk_2, sk_2).$$
The client sends $uk_{1\to 2}$ to the cloud server. Note $uk_{1\to 2}$ is pseudorandom, and hence reveals no information to the cloud server.
2. The cloud server receives $uk_{1\to 2}$ and runs the algorithm $\mathsf{Update}(pp, c, uk_{1\to 2}, pk_2)$ to rotate the key pk_1 encrypting c to pk_2. The output is c_{new}. This step is repeated for all old ciphertexts c under pk_1 in the storage.

Plain data is never recovered in the above steps, which helps preventing any data breach.

4. OUR PROPOSED KR-SU-HE SCHEME

In this section we present and analyse our concrete scheme. Details are below.

Public key homomorphic encryption part. This part is a close variant of the public key encryption scheme in [16].

- $\mathsf{ParamGen}(1^\lambda)$: Fix $q = q(\lambda) \in \mathbb{Z}^+$ and $l \in \mathbb{Z}^+$. Fix $p \in \mathbb{Z}^+$ such that $\gcd(p, q) = 1$. Return $pp = (q, l, p)$.
- $\mathsf{KeyGen}(1^\lambda, pp)$: Take $s = s(\lambda, pp) \in \mathbb{R}^+$ and $n = n(\lambda, pp) \in \mathbb{Z}^+$. Take $R, S \xleftarrow{g} \mathbb{Z}_{(0,s)}^{n\times l}$, and $A \xleftarrow{\$} \mathbb{Z}_q^{n\times n}$. Compute the matrix
$$P = pR - AS \in \mathbb{Z}_q^{n\times l}.$$
Return the public key $pk = (A, P, n, s)$ and the secret key $sk = S$.
- $\mathsf{Enc}(pk, m \in \mathbb{Z}_p^{1\times l})$: Pick vectors of noises $e_1, e_2 \xleftarrow{g} \mathbb{Z}_{(0,s)}^{1\times n}$, and $e_3 \xleftarrow{g} \mathbb{Z}_{(0,s)}^{1\times l}$. Compute
$$\begin{aligned} c_1 &= e_1 A + p e_2 \in \mathbb{Z}_q^{1\times n}, \\ c_2 &= e_1 P + p e_3 + m \in \mathbb{Z}_q^{1\times l}. \end{aligned}$$
Return $c = (c_1, c_2) \in \mathbb{Z}_q^{1\times(n+l)}$.
- $\mathsf{Dec}(S, c = (c_1, c_2))$: Compute $\overline{m} = c_1 S + c_2 \in \mathbb{Z}_q^{1\times l}$ and $m = \overline{m} \bmod p$. Return m.
- $\mathsf{Add}(c, c')$: Return $c + c' \in \mathbb{Z}_q^{1\times(n+l)}$.
- $\mathsf{Mul}(pp, pk, c, c')$: Return
$$c_{\text{mul}} = c^{\mathsf{T}} c' \in \mathbb{Z}_q^{(n+l)\times(n+l)}$$
where $c^{\mathsf{T}} \in \mathbb{Z}_q^{(n+l)\times 1}$ is the transpose of $c \in \mathbb{Z}_q^{1\times(n+l)}$.
- $\mathsf{AddM}(c_{\text{mul}}, c'_{\text{mul}})$: Return $c_{\text{mul}} + c'_{\text{mul}} \in \mathbb{Z}_q^{(n+l)\times(n+l)}$
- $\mathsf{DecA}(sk, c_{\text{add}})$: identical to Dec

Key rotation and security update

$\mathsf{UKGen}(pp, pk_1, sk_1, pk_2, sk_2)$:
Let $pk_i = (A_i, P_i, n_i, s_i)$. Let $sk_i = S_i$ $(i = 1, 2)$
Let $\kappa = \lceil \log_2 q \rceil$. Take $X \xleftarrow{\$} \mathbb{Z}_q^{n_1\kappa\times n_2}$, $E \xleftarrow{g} \mathbb{Z}_{(0,s_2)}^{n_1\kappa\times l}$
$Y = -XS_2 + pE + \mathsf{Power2}(S_1) \in \mathbb{Z}_q^{n_1\kappa\times l}$
Return $uk_{n_1\to n_2} = (X, Y)$

$\mathsf{Update}(pp, c, uk_{n_1\to n_2}, pk_2)$:
Let $c = (c_1, c_2) \in \mathbb{Z}_q^{1\times n_1} \times \mathbb{Z}_q^{1\times l}$
Let $pk_2 = (A_2, P_2, n_2, s_2)$, $uk_{n_1\to n_2} = (X, Y)$
Take $f_1, f_2 \xleftarrow{g} \mathbb{Z}_{(0,s_2)}^{1\times n_2}$, $f_3 \xleftarrow{g} \mathbb{Z}_{(0,s_2)}^{1\times l}$
$E_0 = f_1[A_2|P_2] + p[f_2|f_3] \in \mathbb{Z}_q^{1\times(n_2+l)}$
$F = [\mathsf{Bits}(c_1)X | \mathsf{Bits}(c_1)Y + c_2] \in \mathbb{Z}_q^{1\times(n_2+l)}$
Return $c' = E_0 + F \in \mathbb{Z}_q^{1\times(n_2+l)}$

(In E_0 and F, $[\cdots|\cdots]$ is for matrix concatenation.)

Figure 1: Algorithms for key rotation and security update.

- $\mathsf{DecM}(sk, c_{\text{mul}})$: Let $sk = S \in \mathbb{Z}_q^{n\times l}$ and I_l be the identity matrix of size l. Compute
$$\overline{m} = \begin{bmatrix} S \\ I_l \end{bmatrix}^{\mathsf{T}} (c_{\text{mul}}) \begin{bmatrix} S \\ I_l \end{bmatrix} \in \mathbb{Z}_q^{l\times l}.$$
Return $\overline{m} \in \mathbb{Z}_p^{l\times l}$.

The additively homomorphic part of the above scheme has been used in [3] for privacy-preserving logistic regression. For applications in privacy-preserving linear regression and secure biometric-authentication, see [2].

Key rotation and security update part (Figure 1). For these purposes, we add two additional algorithms UKGen (generating the update key) and Update (doing the key rotation, or security update over ciphertexts), according to Definition 2.

The algorithm UKGen takes two key pairs (pk_1, sk_1) and (pk_2, sk_2) and returns a key $uk_{n_1\to n_2}$. The algorithm Update uses that $uk_{n_1\to n_2}$ to turn a ciphertext c under pk_1 to a ciphertext c' decryptable by sk_2. The details are depicted in Figure 1 in which we need the functions $\mathsf{Power2}(\cdot)$ and $\mathsf{Bits}(\cdot)$ explained as follows. Let $v \in \mathbb{Z}_q^n$ and $\kappa = \lceil \log_2 q \rceil$, then there are bit vectors $v_i \in \{0,1\}^n$ such that $v = \sum_{i=0}^{\kappa-1} 2^i v_i$. Define
$$\mathsf{Bits}(v) = [v_0|\cdots|v_{\kappa-1}] \in \{0,1\}^{1\times n\kappa}.$$
Let $W = [W_1|\cdots|W_l] \in \mathbb{Z}_q^{n\times l}$ where W_i are columns. Then
$$\mathsf{Power2}(W) = \begin{bmatrix} W_1 & \cdots & W_l \\ 2W_1 & \cdots & 2W_l \\ \vdots & & \vdots \\ 2^{\kappa-1}W_1 & \cdots & 2^{\kappa-1}W_l \end{bmatrix} \in \mathbb{Z}_q^{n\kappa\times l}.$$
It is easy to check that
$$\mathsf{Bits}(v)\mathsf{Power2}(W) = vW \in \mathbb{Z}_q^{1\times l}.$$
Intuitively, $\mathsf{Bits}(\cdot)$ is used in Update to limit the noise increase, while $\mathsf{Power2}(\cdot)$ is put in UKGen to ensure correctness of updated ciphertexts using above equation. The functions $\mathsf{Bits}(\cdot)$ and $\mathsf{Power2}(\cdot)$ are originated in [9, 11] as part of the dimension switching technique.

4.1 Correctness and homomorphic properties

We informally check the correctness and homomorphisms of our KR-SU-HE scheme, and then formally establish a theorem for choosing parameters.

Correctness of directly-formed ciphertexts. We check that directly-formed ciphertexts can be decrypted correctly. Indeed, in the decryption Dec algorithm,

$$\begin{aligned} c_1S + c_2 &= (e_1A + pe_2)S + e_1P + pe_3 + m \\ &= e_1(AS + P) + pe_2S + pe_3 + m \\ &= p(e_1R + e_2S + e_3) + m \in \mathbb{Z}_q^{1\times l} \end{aligned} \quad (1)$$

will yield correct m in decryption if the noise $p(e_1R+e_2S+e_3)$ is small enough.

Correctness of updated ciphertexts. We have $c' = E_0 + F \in \mathbb{Z}_q^{1\times(n_2+l)}$. As E_0 is an encryption under pk_2 of length l vector $(0,\ldots,0)$, its decryption under sk_2 gives the zero vector. The decryption of F under $sk_2 = S_2$ is

$$\begin{aligned} \overline{m} &= \mathsf{Bits}(c_1)XS_2 + \mathsf{Bits}(c_1)Y + c_2 \\ &= \mathsf{Bits}(c_1)(pE + \mathsf{Power2}(S_1)) + c_2 \\ &= p\mathsf{Bits}(c_1)E + c_1S_1 + c_2 \in \mathbb{Z}_q^{1\times l} \end{aligned} \quad (2)$$

which is equal to the decryption of $c = (c_1, c_2)$ under $sk_1 = S_1$ as long as the added noise $\mathsf{Bits}(c_1)E$ is small, which holds with high probability as matrix E containing small, Gaussian-distributed elements.

Homomorphic property. Directly, our KR-SU-HE scheme can evaluate the following formulas on ciphertexts

$$\sum_{i=1}^{N_{\text{add}}} CT_i \in \mathbb{Z}_q^{1\times(n+l)} \quad (3)$$

$$\sum_{i=1}^{N_{\text{add}}} CT_i^{\mathsf{T}} \cdot CT_i' \in \mathbb{Z}_q^{(n+l)\times(n+l)} \quad (4)$$

in which (3) is a special case of (4) and yet the noise added is smaller. The decryption of (3) and (4) uses the algorithms DecA and DecM respectively, yielding

$$\sum_{i=1}^{N_{\text{add}}} m_i \in \mathbb{Z}_p^{1\times l} \text{ and } \sum_{i=1}^{N_{\text{add}}} m_i^{\mathsf{T}} \cdot m_i' \in \mathbb{Z}_p^{l\times l}$$

where m_i and m_i' are the plaintext vectors in CT_i and CT_i'.

More multiplications. To get several ciphertext multiplications instead of one as above, one can take $l = 1$ in our scheme and uses known techniques of noise reduction as in [8, 9, 14].

Homomorphisms held even after key rotation or security update. Succinctly, formulas (3) and (4) hold even if CT_i, CT_i' $(1 \le i \le N_{\text{add}})$ are altogether encrypted under the same public key, regardless of whether they are directly formed by the Enc algorithm or are indirectly transformed by Update (Figure 1) from old ciphertexts. Intuitively, this is because (i) the zero encryption E_0 part in updated ciphertexts does not interfere with homomorphisms, and (ii) the F part can be correctly decrypted as in (2).

More precisely, consider following cases, where "old" stands for a ciphertext under pk_1 and "new" for a ciphertext under pk_2, and Update(old) for an updated ciphertext from pk_1 to pk_2:

• (old + old) or (new + new): this should be easily seen, as vector addition of two ciphertexts in the same form $e_1[A|P]+p[e_2|e_3]+[0_n|m]$ and $e_1'[A|P]+p[e_2'|e_3']+[0_n|m']$ under identical public key $[A|P]$ (either old or new) gives us a ciphertext whose decryption will yield $(m + m') \mod p$.

• Update(old) + new: let c_{ud} and c_{nw} be the updated-from-old and under-new public-key ciphertexts correspondingly. The decryption under the new secret key S_2 is

$$\mathsf{Dec}(S_2, c_{ud} + c_{nw}) = \mathsf{Dec}(S_2, c_{ud}) + \mathsf{Dec}(S_2, c_{nw}) \pmod p$$

as the noise increases linearly when doing $c_{ud} + c_{nw}$. The decryption $\mathsf{Dec}(S_2, c_{ud})$ works as in (2) and $\mathsf{Dec}(S_2, c_{nw})$ as in (1), yielding corresponding messages m_{ud} and m_{nw} as expected.

Like the above, one can do the multiplication of (old $\times$ old), (new $\times$ new), and (Update(old) $\times$ new) where $\times$ is the outer product of vectors, justifying that formulas (3) and (4) hold even after key rotation or security update.

We end this subsection by the following theorem on how to set parameters for our scheme.

THEOREM 1 (PARAMETERS FOR CORRECTNESS). *Let parameters p, q, s be as in our KR-SU-HE scheme, and $n_1, \ldots, n_h$ $(h \ge 1)$ are the dimensions in key rotation or security updates, and N_{add} the number of additions over multiplied ciphertexts as in (4) where all ciphertexts are with the same n_i, then the correctness of (4) holds with overwhelming probability if*

$$q = N_{\text{add}} \cdot O\Big(p^2 s^4 \sum_{i=1}^{h} n_i + p^2 s^2 (\log_2 q) \sum_{i=2}^{h} n_i\Big)$$

in which the hidden constant in the $O(\cdot)$ is small.

To prove Theorem 1, we will use following lemmas, whose proofs can be derived from [5,6]. Below $\langle\cdot,\cdot\rangle$ stands for inner product. Writing $||\mathbb{Z}_{(0,s)}^n||$ is a short hand for taking a vector from the discrete Gaussian distribution of deviation s and computing its Euclidean norm.

LEMMA 1. *Let $c \ge 1$ and $C = c \cdot \exp(\frac{1-c^2}{2})$. Then for any real $s > 0$ and any integer $n \ge 1$, we have*

$$\Pr\left[||\mathbb{Z}_{(0,s)}^n|| \ge \frac{c \cdot s\sqrt{n}}{\sqrt{2\pi}}\right] \le C^n.$$

LEMMA 2. *For any real $s > 0$ and $T > 0$, and any $x \in \mathbb{R}^n$, we have*

$$\Pr\left[||\langle x, \mathbb{Z}_{(0,s)}^n\rangle|| \ge Ts||x||\right] < 2\exp(-\pi T^2).$$

PROOF (OF THEOREM 1). For now suppose $h = 2$, we check the decryption by secret key S_2 in dimension n_2 of updated ciphertexts. Using the notations as in Figure 1, let $E_0 = [E_1|E_2]$ and $F = [F_1|F_2]$ for $E_1, F_1 \in \mathbb{Z}_q^{1\times n_2}$, and $E_2, F_2 \in \mathbb{Z}_q^{1\times l}$, so $c' = E_0 + F = [E_1 + F_1|E_2 + F_2]$, and the

decryption of c' by S_2 becomes

$$\begin{aligned}
&(E_1+F_1)S_2+E_2+F_2\\
&= \big(f_1A_2+pf_2+\mathsf{Bits}(c_1)X\big)S_2+f_1P_2+pf_3\\
&\quad +\mathsf{Bits}(c_1)Y+c_2\\
&= f_1A_2S_2+pf_2S_2+\mathsf{Bits}(c_1)XS_2+f_1(pR_2-A_2S_2)\\
&\quad +pf_3+\mathsf{Bits}(c_1)Y+c_2\\
&= pf_2S_2+\mathsf{Bits}(c_1)XS_2+pf_1R_2+pf_3\\
&\quad +\mathsf{Bits}(c_1)\big(-XS_2+pE+\mathsf{Power2}(S_1)\big)+c_2\\
&= pf_2S_2+pf_1R_2+pf_3\\
&\quad +p\mathsf{Bits}(c_1)E+\mathsf{Bits}(c_1)\mathsf{Power2}(S_1)+c_2\\
&= \underbrace{p(f_2S_2+f_1R_2+f_3+\mathsf{Bits}(c_1)E)}_{\text{noise incurred after one Update}}+c_1S_1+c_2\in\mathbb{Z}_q^{1\times l}.
\end{aligned}$$

In the worst case, $\mathsf{Bits}(c_1)$ contains all 1's, so that the noise added after one update is

$$p(f_1R_2+f_2S_2+f_3+\mathbf{1}_{1\times n_1\kappa}\cdot E).$$

Generally, the noise added after h updates corresponding to key dimension n_i $(2\le i\le h)$ is a sum of form

$$p\sum_{i=2}^{h}\left(f_1^{(i)}R_2^{(i)}+f_2^{(i)}S_2^{(i)}+f_3^{(i)}+\mathbf{1}_{1\times n_i\kappa}\cdot E^{(i)}\right).$$

Each component in $\mathbb{Z}_q$ of the total noise in $\mathbb{Z}_q^{1\times l}$, namely including the noise in the original ciphertext, can be written as the inner product of two vectors of form

$$\begin{aligned}
e &= (f_1^{(2)},f_2^{(2)},f_3^{(2)},\ \ldots,f_1^{(h)},f_2^{(h)},f_3^{(h)},\\
&\quad e^{(2)},\ldots,e^{(h)},\ \ e_1,e_2,e_3)\\
x &= (r_2^{(2)},s_2^{(2)},\mathbf{0}\mathbf{1}\mathbf{0}_{1\times l},\ldots,r_2^{(h)},s_2^{(h)},\mathbf{0}\mathbf{1}\mathbf{0}_{1\times l},\\
&\quad \underbrace{\mathbf{1}_{1\times n_i\kappa}}_{2\le i\le h},r,r',\mathbf{0}\mathbf{1}\mathbf{0}_{1\times l})
\end{aligned}$$

where, for all $2\le i\le h$,

- Vectors $f_1^{(i)},f_2^{(i)}\xleftarrow{g}\mathbb{Z}_{(0,s)}^{1\times n_i}$, and $f_3^{(i)}\xleftarrow{g}\mathbb{Z}_{(0,s)}^{1\times l}$.
- Vectors $e^{(i)}\xleftarrow{g}\mathbb{Z}_{(0,s)}^{1\times n_i\kappa}$ represents one column in matrix $E^{(i)}$.
- Vectors $e_1\xleftarrow{g}\mathbb{Z}_{(0,s)}^{1\times n_1}, e_2\xleftarrow{g}\mathbb{Z}_{(0,s)}^{1\times n_1}$, and $e_3\xleftarrow{g}\mathbb{Z}_{(0,s)}^{1\times l}$ are the noises in the original ciphertext.
- Vectors $r_2^{(i)},s_2^{(i)}\xleftarrow{g}\mathbb{Z}_{(0,s)}^{1\times n_i}$, and $\mathbf{0}\mathbf{1}\mathbf{0}_{1\times l}$ stands for a vector of length l with all 0's except one 1; $\mathbf{1}_{1\times n_i\kappa}$ for a vector of length $n_i\kappa$ with all 1's. Vectors $r,r'\xleftarrow{g}\mathbb{Z}_{(0,s)}^{1\times n_1}$ represent corresponding columns in matrices R,S.

Here we use the same deviation s for all dimensions to ease the computation. We have

$$\begin{aligned}
e &\in \mathbb{Z}_{(0,s)}^{1\times\left(\sum_{i=1}^{h}(2n_i+l)+\sum_{i=2}^{h}n_i\kappa\right)}\\
||x|| &\le ||(r_2^{(2)},s_2^{(2)},\ldots,r_2^{(h)},s_2^{(h)},r,r')||+\sqrt{\kappa\sum_{i=2}^{h}n_i+h}
\end{aligned}$$

where $(r_2^{(2)},s_2^{(2)},\ldots,r_2^{(h)},s_2^{(h)},r,r')\in\mathbb{Z}_{(0,s)}^{1\times\left(2\sum_{i=1}^{h}n_i\right)}$. Applying Lemma 1 for vector of length $2\sum_{i=1}^{h}n_i$, with high probability of

$$1-C^{2\sum_{i=1}^{h}n_i}(\ge 1-2^{-40}\text{ for all choices of parameters})$$

we have

$$||x||\le\frac{c\cdot s\sqrt{2\sum_{i=1}^{h}n_i}}{\sqrt{2\pi}}+\sqrt{\kappa\sum_{i=2}^{h}n_i+h}.$$

We now use Lemma 2 with vectors x and e. Let ρ be the error per message symbol in decryption, we set $2\exp(-\pi T^2)=\rho$, so $T=\sqrt{\ln(2/\rho)}/\sqrt{\pi}$. The bound on the noise becomes $pTs||x||$, which is not greater than

$$\begin{aligned}
&\frac{ps\sqrt{\ln(2/\rho)}}{\sqrt{\pi}}\left(\frac{c\cdot s\sqrt{2\sum_{i=1}^{h}n_i}}{\sqrt{2\pi}}+\sqrt{\kappa\sum_{i=2}^{h}n_i+h}\right)\\
&\stackrel{\text{def}}{=} B(\rho,h,s,n_1,\ldots,n_h,p,q).
\end{aligned}\tag{5}$$

No update ($h=1$): (5) becomes

$$\frac{ps\sqrt{\ln(2/\rho)}}{\sqrt{\pi}}\left(\frac{c\cdot s\sqrt{2n_1}}{\sqrt{2\pi}}\right)=\frac{pcs^2\sqrt{\ln(2/\rho)\cdot n_1}}{\pi}.$$

which is the upper-bound of noise in each original ciphertext. If original ciphertexts are multiplied and added as in (4), the corresponding noise bound is set below $q/2$ for correctness

$$\frac{N_{\text{add}}n_1p^2c^2s^4\ln(2/\rho)}{\pi^2}\le\frac{q}{2}.\tag{6}$$

With update ($h\ge 2$): Now we consider (5) with $h\ge 2$. As in (6) we set

$$N_{\text{add}}\cdot B(\rho,h,s,n_1,\ldots,n_h,p,q)^2\le\frac{q}{2}\tag{7}$$

becomes the condition for correctness stated in the statement of the theorem, ending the proof. □

4.2 Security analyses

We first provide the intuition and then continue with formal proofs.

Security intuition on updated ciphertexts. Suppose keys and ciphertexts in dimension n_1 is considered insecure, so that the ciphertexts must be transformed into secure dimension $n_2>n_1$. In other words, consider the worst case in which an adversary $\mathcal{A}$ owns the secret sk_1 in dimension n_1, referring to Figure 1, we need to show that the transformed ciphertext,

$$c'=E_0+F\in\mathbb{Z}_q^{1\times(n_2+l)}$$

remains secure. That holds true since the encryption using pk_2 of zero

$$E_0=f_1[A_2|P_2]+p[f_2|f_3]\in\mathbb{Z}_q^{1\times(n_2+l)}$$

is pseudo-random under the LWE assumption in dimension n_2 regardless of F. Certainly, we need to mention that $\mathcal{A}$ additionally has access to public matrices A_2 and $P_2=pR_2-A_2S_2$ and even the update key $uk_{n_1\to n_2}$ containing matrices $(X,-XS_2+pE)$. These pieces of information can be arranged in matrix form as

$$\begin{bmatrix}A_2\\X\end{bmatrix},p\begin{bmatrix}R_2\\E\end{bmatrix}-\begin{bmatrix}A_2\\X\end{bmatrix}S_2$$

which is also pseudo-random under LWE assumption in dimension n_2.

Formally, the security of our proposed scheme is ensured by the following theorems.

THEOREM 2 (d-CPA SECURITY). *Our scheme is d-CPA secure under the decision LWE assumption in dimension n_d for all $d \geq 1$. In particular, for any poly-time adversary $\mathcal{A}$ against the proposed scheme, there is a poly-time $\mathcal{D}$ against LWE satisfying*

$$\mathbf{Adv}^{d-\mathrm{cpa}}_{\mathcal{A}}(\lambda) \leq (l+1) \cdot \mathbf{Adv}^{\mathrm{LWE}(n_d,s_d,q)}_{\mathcal{D}}(\lambda).$$

PROOF. We proceed in games as follows. Let $\mathbf{Game}_0$ be the d-CPA game in Definition 3. The public key at dimension n_d is $(A_d, P_d = pR_d - A_dS_d)$ and the update key from dimension n_i ($i < d$) to dimension n_d is $uk_{n_i \to n_d} = \big(X_i, -X_iS_d + pE_i + \mathsf{Power2}(S_i)\big)$, where $X_i \xleftarrow{\$} \mathbb{Z}_q^{n_i\kappa \times n_d}$ and Gaussian noises $E_i \xleftarrow{\mathrm{g}} \mathbb{Z}_q^{n_i\kappa \times l}$ are freshly chosen by the challenger for each update key.

$\mathbf{Game}_1$: this game is identical to $\mathbf{Game}_0$ except that public P_d is taken randomly, and above update keys are computed and returned by the challenger as $uk_{n_i \to n_d} = \big(X_i, Y_i\big)$, for random matrices $X_i \xleftarrow{\$} \mathbb{Z}_q^{n_i\kappa \times n_d}$ and $Y_i \xleftarrow{\$} \mathbb{Z}_q^{n_i\kappa \times l}$.

The difference between $\mathbf{Game}_0$ and $\mathbf{Game}_1$ are in turning the matrices $(A_d, P_d = pR_d - A_dS_d)$ into random, and tuples of form $(X_i, -X_iS_d + pE_i + \mathsf{Power2}(S_i))$ into random tuples (X_i, Y_i) where indexes i correspond to update key queries. In other words, the second term $\mathbf{ST}$ in the following is turned into a random matrix

$$\mathbf{FT} = -\begin{bmatrix} A_d \\ \begin{bmatrix} \vdots \\ X_i \\ \vdots \end{bmatrix}_i \end{bmatrix}, \mathbf{ST} = \mathbf{FT} \cdot S_d + p \begin{bmatrix} R_d \\ \begin{bmatrix} \vdots \\ E_i \\ \vdots \end{bmatrix}_i \end{bmatrix},$$

with secret S_d of dimension n_d. To be exact with the LWE assumption, we need to get rid of the p multiplication in $\mathbf{ST}$. Here we use the condition $\gcd(p, q) = 1$ so that p^{-1} exists in $\mathbb{Z}_q$. Consider the pair $(p^{-1}\mathbf{FT}, p^{-1}\mathbf{ST})$ where operations are over $\mathbb{Z}_q$. Since $\mathbf{FT}$ is random , so is $p^{-1}\mathbf{FT}$. Furthermore, $p^{-1}\mathbf{ST}$ is of form $[p^{-1}\mathbf{FT} \times \mathrm{secret} + \mathrm{noise}]$ of the LWE assumption. Therefore the pair $(p^{-1}\mathbf{FT}, p^{-1}\mathbf{ST})$ is random under the LWE assumption, so is the pair $(\mathbf{FT}, \mathbf{ST})$ again due to $\gcd(p, q) = 1$.

Thus both games are indistinguishable to $\mathcal{A}$ under the LWE assumption.

$\mathbf{Game}_2$: this game is identical to $\mathbf{Game}_1$ except that the challenge ciphertext is changed to random. Specifically,

$$\begin{aligned} C^*_d &= \mathsf{Update}(uk^*_{n_{d-1} \to n_d}, C^*_{d-1}, \underbrace{A_d, P_d}_{pk_d}) \\ &= f_1[A_d|P_d] + p[f_2|f_3] + F \in \mathbb{Z}_q^{1 \times (n_d + l)}, \end{aligned}$$

where f_1, f_2, f_3 are Gaussian noises chosen by the challenger, is now computed as

$$C^*_d = R_d + F$$

where $R_d \xleftarrow{\$} \mathbb{Z}_q^{1 \times (n_d + l)}$. The change is still indistinguishable to $\mathcal{A}$ thanks to the LWE assumption with secret $f_1 \in \mathbb{Z}_s^{1 \times n_d}$.

Since R_d is random, so is C^*_d. Therefore, in this game, all information $\mathcal{A}$ gets from its queries is random and hence useless in guessing the hidden bit b. Thus in this game $\Pr[b' = b] = \frac{1}{2}$.

The term $l + 1$ in the reduction comes from applying $\mathrm{LWE}(n_d, s_d, q)$ over the l columns of S_d in $\mathbf{Game}_1$, and another one time in $\mathbf{Game}_2$, ending the proof. □

THEOREM 3 (CPA SECURITY). *Our scheme is CPA secure under the decision LWE assumption. In particular, for any poly-time adversary $\mathcal{A}$ against the proposed scheme, there is a poly-time $\mathcal{D}$ against LWE satisfying*

$$\mathbf{Adv}^{\mathrm{cpa}}_{\mathcal{A}}(\lambda) \leq (l+1) \cdot \mathbf{Adv}^{\mathrm{LWE}(n_d,s_d,q)}_{\mathcal{D}}(\lambda).$$

PROOF. The proof is almost identical to that of Theorem 2. The difference is in $\mathbf{Game}_2$ where the challenge ciphertext

$$C^* = \mathsf{Enc}(pk_d, m_b) = e^*_1[A_d|P_d] + p[e^*_2|e^*_3] + [0_{1 \times n_d}|m_b]$$

is replaced by a random tuple, which is indistinguishable to the adversary thanks to the $\mathrm{LWE}(n_d, s_d, q)$ assumption with secret e^*_1. □

4.3 Testing implementation

For parameter selection related to the LWE assumption, we rely on attacks on LWE in [1,16,17] and [13]. In particular, our choices of parameters are consistent with the newest one in [13].

For generating discrete Gaussian noises, we employ the Knuth-Yao algorithm [15]. The noise generation only occupies a negligible time in encryption. For example, when $s = 10$, our implementation can generate more than $4 \cdot 10^4$ Gaussian samples in one millisecond using only 1.65 megabytes to store a binary tree.

Table 2: Our timings when $q = 2^{114}, p = 2^{30} + 1$.

bit-sec	KeyGen	Enc	Dec	Add	Mul	AddM	DecM
80	1428	63.2	0.92	0.003	35.1	29.3	1313
128	2513	94.7	1.22	0.004	60.8	50.1	2296
256	7249	313	2.05	0.010	164	136	6643

(All times are in milliseconds, averaged over 1000 executions.)

	bit-sec → bit-sec	UKGen	Update
Key rotation	80 → 80	165.6 (s)	1.1 (s)
	128 → 128	291.4 (s)	1.9 (s)
	256 → 256	846.6 (s)	5.3 (s)
Security update	80 → 128	238.2 (s)	1.5 (s)
	128 → 256	519.8 (s)	3.3 (s)

(In both tables, times are averaged over 1 thread of a Xeon E5-2660 v3, 2.60GHz machine.)

Taking $q = 2^{114}, p = 2^{30} + 1, s = 8.0, l = 64$, we need $n = 2661, 3530, 5847$ respectively for estimated 80-, 128-, 256-bit securities. The message length $l = 64$ is sufficient to handle real numbers of 64-bit precision. The running times are reported in Table 2. As seen in the table, generating an update key requires a few minutes, while updating a ciphertext needs a few seconds. The update task can be fully parallelized on the cloud server.

5. CONCLUSION

We conceptually propose the notion of key-rotatable and security-updatable homomorphic encryption, and build a concrete scheme. Our scheme is proved secure under the LWE assumption, and is showed efficient via implementation. It can be used in secure cloud computing to accomplish the vital task of key rotation and security update when necessary.

6. REFERENCES

[1] Y. Aono, X. Boyen, L. T. Phong, and L. Wang. Key-private proxy re-encryption under LWE. In G. Paul and S. Vaudenay, editors, *INDOCRYPT*, volume 8250 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2013.

[2] Y. Aono, T. Hayashi, L. T. Phong, and L. Wang. Fast and secure linear regression and biometric authentication with security update. *IACR Cryptology ePrint Archive*, 2015:692, 2015.

[3] Y. Aono, T. Hayashi, L. T. Phong, and L. Wang. Privacy-preserving logistic regression with distributed data sources via homomorphic encryption. *IEICE Transactions*, 99-D(8):2079–2089, 2016.

[4] B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In S. Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 595–618. Springer, 2009.

[5] W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(1):625–635, 1993.

[6] W. Banaszczyk. Inequalities for convex bodies and polar reciprocal lattices in $\mathbb{R}^n$. *Discrete & Computational Geometry*, 13(1):217–231, 1995.

[7] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In K. Nyberg, editor, *EUROCRYPT*, volume 1403 of *Lecture Notes in Computer Science*, pages 127–144. Springer, 1998.

[8] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 868–886. Springer, 2012.

[9] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In S. Goldwasser, editor, *ITCS*, pages 309–325. ACM, 2012. Available at https://eprint.iacr.org/2011/277.pdf.

[10] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé. Classical hardness of learning with errors. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *STOC*, pages 575–584. ACM, 2013.

[11] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In R. Ostrovsky, editor, *FOCS*, pages 97–106. IEEE, 2011.

[12] N. Chandran, M. Chase, F. Liu, R. Nishimaki, and K. Xagawa. Re-encryption, functional re-encryption, and multi-hop re-encryption: A framework for achieving obfuscation-based security and instantiations from lattices. In H. Krawczyk, editor, *Public-Key Cryptography - PKC 2014*, volume 8383 of *Lecture Notes in Computer Science*, pages 95–112. Springer, 2014.

[13] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Proceedings, Part I*, pages 3–33, 2016.

[14] C. Gentry. *A fully homomorphic encryption scheme.* PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig.

[15] D. E. Knuth and A. C. Yao. The complexity of non-uniform random number generation. *Algorithms and Complexity, Academic Press, New York*, pages 357–428, 1976.

[16] R. Lindner and C. Peikert. Better key sizes (and attacks) for LWE-based encryption. In A. Kiayias, editor, *CT-RSA*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339. Springer, 2011.

[17] M. Liu and P. Q. Nguyen. Solving BDD by enumeration: An update. In E. Dawson, editor, *CT-RSA*, volume 7779 of *Lecture Notes in Computer Science*, pages 293–309. Springer, 2013.

[18] D. Micciancio and O. Regev. Lattice-based cryptography. In *Post-Quantum Cryptography*, pages 147–191. Springer, 2009.

[19] National Institute of Standards and Technology (NIST). Recommendation for Key Management: Part 1: General (Revision 3). http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf.

[20] R. Nishimaki and K. Xagawa. Key-private proxy re-encryption from lattices, revisited. *IEICE Transactions*, 98-A(1):100–116, 2015.

[21] Open Web Application Security Project. https://www.owasp.org/index.php/Cryptographic_Storage_Cheat_Sheet.

[22] Payment Card Industry Data Security Standard. https://www.pcisecuritystandards.org/documents/Prioritized_Approach_V2.0.pdf.

[23] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In H. N. Gabow and R. Fagin, editors, *STOC*, pages 84–93. ACM, 2005.