Proxy Re-encryption from Lattices

Elena Kirshanova

Horst Görtz Institute for IT-Security
Faculty of Mathematics
Ruhr University Bochum, Germany
elena.kirshanova@rub.de

Abstract. We propose a new unidirectional proxy re-encryption scheme based on the hardness of the LWE problem. Our construction is collusion-safe and does not require any trusted authority for the re-encryption key generation. We extend a recent trapdoor definition for a lattice of Micciancio and Peikert. Our proxy re-encryption scheme is provably CCA-1 secure in the selective model under the LWE assumption.
Keywords. Proxy re-encryption, lattices, learning with errors.

1 Introduction

There are a number of applications (distributed file system of [3], email forwarding) which require that some data encrypted for Alice has to be re-encrypted to Bob. A naive way Alice can accomplish this task is to decrypt the data with her secret key and then encrypt the resulting plaintext under Bob's public key. But this approach requires Alice to actively participate to perform the procedures. Moreover, she needs to repeat the encryption for any further user she wants to resend the message to. In a proxy re-encryption (PRE) scheme, a proxy is given a special information (a re-encryption key) that allows it to translate a ciphertext intended for Alice into a ciphertext of the same message encrypted under Bob's key. In this setting we will call Alice the delegator and Bob the delegatee. The proxy cannot, however, learn either the underlying plaintext or the secret key of either Alice or Bob.

In 1998, Blaze *et al.* ([6]) proposed the first proxy re-encryption scheme. Their construction is based on the ElGamal encryption scheme ([8]): for a group $\mathbb{G}$ of prime order p and g a generator of the group, Alice and Bob choose their key pair (a, g^a) and $(b, g^b), a, b \leftarrow \mathbb{Z}_p^*$. The encryption of a message m intended for Alice then has the form $c = (c_1, c_2) = (mg^r, (g^a)^r)$ for a randomly chosen $r \leftarrow \mathbb{Z}_p^*$. The re-encryption key from Alice to Bob is $\mathsf{rk}_{(Alice \to Bob)} = b/a$, and the proxy translates the ciphertext c to Bob by computing $c' = (c_1, c_2^{b/a}) = (mg^r, (g^b)^r)$.The scheme is CPA secure under the Decisional Diffie-Hellman assumption in $\mathbb{G}$. From the re-encryption key the proxy can easily compute a/b, that allows it to convert the ciphertexts in the inverse direction. Such PRE schemes are called *bidirectional*. More desirable in practice are *unidirectional* schemes, in which a re-encryption key works only in one direction.

In the above PRE scheme if the proxy and one of the parties collude, they can recover the secret key of another party. The second issue is that a proxy knowing $\mathsf{rk}_{A \to B} = b/a$ and $\mathsf{rk}_{B \to C} = c/b$ can compute $\mathsf{rk}_{A \to C} = c/a$. Ateniese *et al.* in [3] listed desired properties for PRE schemes; among them are:

– *Non-interactivity*: $\mathsf{rk}_{(Alice \to Bob)}$ can be generated by Alice alone using Bob's public key; no trusted authority is needed.
– *Proxy transparency*: neither the delegator nor the delegatees are aware of the presence of a proxy, i.e. a recipient of a ciphertext cannot distinguish whether the ciphertext is the original encryption or whether it was re-encrypted. The property is achieved in [6].
– *Key optimality*: the size of Bob's secret key remains constant, regardless of how many delegations he accepts.
– *Collusion resilience* (also called *master key security* in [3] and [4]): it is hard for the coalition of the proxy and Bob to compute Alice's secret key.
– *Non-transitivity*: it should be hard for the proxy to re-delegate the decryption right, namely to compute $\mathsf{rk}_{A \to C}$ from $\mathsf{rk}_{A \to B}$, $\mathsf{rk}_{B \to C}$.

1.1 Related work

Bidirectional proxy re-enryption scheme was proposed by Blaze *et al.* in [6], while a unidirectional construction firstly appeared as a building block of a secure distributed file system in [3], [4]. The formal definition of CCA security for PRE with a bidirectional scheme is present in [7]. CCA security is achieved for the unidirectional setting in [12]. Both schemes use bilinear pairings.

The possibility of using lattice-based assumptions for PRE constructions was shown by Xagawa in [18], but the scheme lacks a complete security analysis. Like Blaze *et al.* the scheme modifies the ElGamal encryption scheme adding the re-encryption key of the form $\mathsf{rk}_{A \to B} = b/a$, where a and b are discrete logarithms of the public keys of Alice and Bob, the scheme of Xagawa and Tanaka is an analogous modification of Regev's encryption scheme ([16]). And, as its ElGamal counterpart, it is bidirectional, it does not provide collusion safeness, neither it is non-interactive: a trusted party is needed to generate the re-encryption keys.

1.2 Our contribution

The main contribution of this paper is a *unidirectional* single-hop proxy re-encryption scheme based on the hardness of lattice-based problems. Prior to [10] there was no known construction that is both unidirectional and multi-hop. But even in [10] this combination comes at the cost of allowing the ciphertext to grow linearly with respect to the number of re-encryptions. Although Gentry in [9] mentions that a fully-homomorphic scheme can achieve multi-use and unidirectionality simultaneously, the constructions of FHE are far from practical. We achieve *CCA-1 security* in the selective model ([4]). Our scheme is the first lattice-based construction that achieves *collusion resilience* and *non-interactivity.* We apply the trapdoor delegation technique proposed in [15]. However, we have to extend the definition of a lattice trapdoor of [15]. The generalization might prove useful for functionalities other than proxy re-encryption as well.

2 Definitions

This section recalls the definition of *unidirectional* proxy re-encryption and the game-based definition of security, where we follow the selective model of Ateniese *et al.* ([3]), but in the chosen-ciphertext security setting, which was formalized in [7]. We are interested in the unidirectional case (i.e. a re-encryption key from pk to pk' should not provide the ability to re-encrypt from pk' to pk).

Definition 1 (Unidirectional PRE) *A unidirectional, proxy re-encryption scheme is a tuple of algorithms* (KeyGen, $\mathsf{ReKeyGen}$, Enc, ReEnc, Dec)*:*

– $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^n)$*. On input the security parameter* 1^n*, the key generation algorithm* KeyGen *outputs a key pair* $(\mathsf{pk}, \mathsf{sk})$*.*
– $\mathsf{rk}_{\mathsf{pk}\to\mathsf{pk}'} \leftarrow \mathsf{ReKeyGen}(\mathsf{pk}, \mathsf{sk}, \mathsf{pk}')$*. On input a private key* sk *of a delegator and a public key of a delegatee* pk'*, algorithm* $\mathsf{ReKeyGen}$ *outputs a unidirectional re-encryption key* $\mathsf{rk}_{\mathsf{pk}\to\mathsf{pk}'}$*.*
– $c \leftarrow \mathsf{Enc}(\mathsf{pk}, \boldsymbol{m})$*. On input a public key* pk *and a message* $\boldsymbol{m}$*, algorithm* Enc *outputs a ciphertext* c*.*
– $c' \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{\mathsf{pk}\to\mathsf{pk}'}, c)$*. On input a re-encryption key* $\mathsf{rk}_{\mathsf{pk}\to\mathsf{pk}'}$ *and a ciphertext* c'*, algorithm* ReEnc *outputs a ciphertext* c' *decryptable under the secret key* sk'*.*
– $\boldsymbol{m} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{pk}, c)$*. On input a secret key* sk*, a public key* pk *and a ciphertext* c'*, algorithm* Dec *outputs a message* $\boldsymbol{m}$ *or the error symbol* $\perp$*.*

Definition 2 (Multi/Single-hop PRE) *A proxy re-encryption scheme is called multi-hop if a proxy can apply further re-encryptions to already re-encrypted ciphertext. In a single-hop setting a ciphertext can be re-encrypted only once.*

The requirements for correctness of decryption depend on whether the scheme is multi-hop or single-hop. Informally, the decryption algorithm should output the correct plaintext, no matter whether the ciphertext is "freshly" encrypted or re-encrypted.

Definition 3 (Single-hop PRE Correctness) *A proxy re-encryption scheme* (KeyGen, KeyGen, $\mathsf{ReKeyGen}$, Enc, ReEnc, Dec) *correctly decrypts for the plaintext space* $\mathcal{M}$ *if:*

– *For all* $(\mathsf{pk}, \mathsf{sk})$ *output by* KeyGen *and for all* $\boldsymbol{m} \in \mathcal{M}$*, it holds that* $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, \boldsymbol{m})) = \boldsymbol{m}$.
– *For any re-encryption key* $\mathsf{rk}_{\mathsf{pk}\to\mathsf{pk}'}$ *output by* $\mathsf{ReKeyGen}(\mathsf{sk}, \mathsf{pk}, \mathsf{pk}')$ *and any* $c = \mathsf{Enc}(\mathsf{pk}, \boldsymbol{m})$ *it holds that* $\mathsf{Dec} = (\mathsf{sk}', \mathsf{ReEnc}(rk_{\mathsf{pk}\to\mathsf{pk}'} c)) = \boldsymbol{m}$.

We give the game-based definition of security for PRE schemes. A discussion follows the definition.

Definition 4 (Unidirectional PRE-CCA1 Game) *Let* 1^n *be the security parameter,* $\mathcal{A}$ *be any ppt adversary. Consider the following experiment for a PRE scheme* $\Pi = (\mathsf{KeyGen}, \mathsf{ReKeyGen}, \mathsf{Enc}, \mathsf{ReEnc}, \mathsf{Dec})$ *with a plaintext space* $\mathcal{M}$*, a key space* $\mathcal{K}$ *and a ciphertext space* $\mathscr{C}$ *(the arrows represent interaction between the adversary and the scheme* Π*):*

$\mathrm{PRE}^{\mathrm{CCA1}}_{\mathcal{A},\Pi}(n)$		$\mathcal{A}$
1.$(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{KeyGen}(1^n)$ Add $(\mathsf{pk}^*, \mathsf{sk}^*)$ to $\mathcal{H}$	$\xrightarrow{1^n, \mathsf{pk}^*}$	
2.$(\mathsf{pk}_H, \mathsf{sk}_H) \leftarrow \mathsf{KeyGen}(1^n)$ $(\mathsf{pk}_H, \mathsf{sk}_H) \in \mathcal{H}$	$\xleftarrow{\text{Add an honest user}}$ $\xrightarrow{\mathsf{pk}_H}$	
3.$(\mathsf{pk}_C, \mathsf{sk}_C) \leftarrow \mathsf{KeyGen}(1^n)$ $(\mathsf{pk}_C, \mathsf{sk}_C) \in \mathcal{C}$	$\xleftarrow{\text{Add a corrupted user}}$ $\xrightarrow{(\mathsf{pk}_C, \mathsf{sk}_C)}$	
4. If $(\mathsf{pk}, \mathsf{sk}) \in \mathcal{H}$ $m = \mathsf{Dec}(c, \mathsf{sk})$	$\xleftarrow{c, \mathsf{pk}}$ $\xrightarrow{m}$	$c \in \mathscr{C}$
5. If $\mathsf{pk}, \mathsf{pk}' \in \mathcal{H}$ or $\mathsf{pk}, \mathsf{pk}' \in \mathcal{C}$ $\mathsf{rk}_{\mathsf{pk} \to \mathsf{pk}'} = \mathsf{ReKeyGen}(\mathsf{pk}, \mathsf{pk}')$	$\xleftarrow{(\mathsf{pk}, \mathsf{pk}')}$ $\xrightarrow{\mathsf{rk}_{\mathsf{pk} \to \mathsf{pk}'}}$	$(\mathsf{pk}, \mathsf{pk}') \in \mathcal{K} \times \mathcal{K}$
	... Repeat steps 2-5 $poly(n)$ times ...	
	$\xleftarrow{m_0, m_1}$	$m_0, m_1 \in \mathcal{M}$
6. $b \to \{0, 1\}$ $c^* = \mathsf{Enc}(\mathsf{pk}^*, m_b)$	$\xrightarrow{c^*}$	
	... Repeat steps 2,3,5 $poly(n)$ times ...	
If $b = b'$ output 1 else output 0	$\xleftarrow{b'}$	$b' \in \{0, 1\}$

An adversary $\mathcal{A}$ wins the game with advantage ϵ if the probability, taken over the random choices of $\mathcal{A}$ and of the oracles, that the experiment $PRE^{CCA1}_{\mathcal{A},\Pi}(n)$ outputs 1, is at least $1/2 + \epsilon$.

To describe the security model we first classify all of the users into *honest* ($\mathcal{H}$) and *corrupted* ($\mathcal{C}$). In the honest case an adversary knows only a public key, whereas for a corrupted user the adversary has both secret and public keys.

We start by choosing a target user $(\mathsf{pk}^*, \mathsf{sk}^*)$ and label it as honest. While an adversary queries for the keys, we disallow any adaptive corruption: the adversary cannot be given a decryption key for any user from $\mathcal{H}$ during the game. The adversary can ask for a decryption of a ciphertext c for any user. The adversary is given access to a re-encryption key from pk to pk' forbidding the case when $\mathsf{pk} \in \mathcal{H}$ and $\mathsf{pk}' \in \mathcal{C}$, which is equivalent to an adaptive corruption of pk. Note that the generation of a re-encryption key from a corrupted to a honest party can be accomplished by the adversary himself, since he knows the secret key of a delegator. As long as he can query for the re-encryption key, the adversary can also perform a re-enryption at any time.

After the challenge ciphertext c^* has been produced, we still allow the adversary to query for the re-encryption keys, so he can also re-encrypt c^*.

Definition 5 (PRE-CCA1 Security) *A unidirectional proxy re-encryption scheme is CCA-1 secure, if any ppt adversary wins the Unidirectional PRE-CCA1 Game only with negligible advantage.*

3 Lattices

We denote column-vectors by lower-case bold letters, so row-vectors are represented via transposition (e.g., $\mathbf{b}^t$). Matrices are denoted by upper-case bold letters, an additive subgroup of $m \times n$ matrices over $\mathbb{R}$ is denoted by $M_{m,n}$. For any $\mathbf{B} \in M_{m,n}$ we denote $\sigma_i(\mathbf{B})$ as decreasingly ordered sequence of singular values of $\mathbf{B}$. A symmetric matrix $\boldsymbol{\Sigma} \in M_{n,n}$ is *semidefinite*, if $\mathbf{x}^t\boldsymbol{\Sigma}\mathbf{x} \geq 0$ for all nonzero $\mathbf{x} \in \mathbb{R}^n$. For any $\mathbf{B} \in M_{n,n}$, the unique matrix $\mathbf{B}^+$ is the Moore-Penrose pseudoinverse, if $\mathbf{B}\mathbf{B}^+\mathbf{B} = \mathbf{B}, \mathbf{B}^+\mathbf{B}\mathbf{B}^+ = \mathbf{B}^+$ and $\mathbf{B}\mathbf{B}^+, \mathbf{B}^+\mathbf{B}$ are symmetric. For any matrix $\mathbf{B}$ the symmetric matrix $\boldsymbol{\Sigma} = \mathbf{B}\mathbf{B}^t$ is positive definite. We denote then $\mathbf{B} = \sqrt{\boldsymbol{\Sigma}}$. A function $f : \mathbb{N} \to \mathbb{R}$ is called *negligible*, denoted $f(n) = \mathsf{negl}(n)$, if for every $c \in \mathbb{N}$ there is an integer n_c such that $f(n) \leq n^c$, $\forall n \geq n_c$. Throughout the paper the parameter $r = w(\sqrt{\log n})$ represents a fixed function $r \approx \sqrt{\ln(2/\epsilon)/\pi}$ that arises from the randomized-rounding operation from $\mathbb{R}$ to $\mathbb{Z}$ and corresponds to the so-called *smoothing parameter* for $\mathbb{Z}^n$ (the definition of the smoothing parameter follows).

3.1 Lattice definition

Let $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \subset \mathbb{R}^m$ be a set of n linearly independent vectors. The *lattice* Λ of *rank* n generated by the basis $\mathbf{B}$ is the set of vectors

$$\Lambda = \mathcal{L}(\mathbf{B}) = \{\mathbf{B}\mathbf{c} : \mathbf{c} \in \mathbb{Z}^n\}.$$

We will work with *full-rank* integer lattices, i.e. $\Lambda \subset \mathbb{Z}^m$ with $m = n$. The *dual* lattice Λ^* is the set is the set of all vectors $\mathbf{y} \in \mathbb{R}^m$ satisfying $\langle \mathbf{x},\mathbf{y} \rangle \in \mathbb{Z}$ for all vectors $\mathbf{x} \in \Lambda$. If $\mathbf{B}$ is a basis of an arbitrary lattice Λ, then $\mathbf{B}^* = \mathbf{B}(\mathbf{B}^t\mathbf{B})^{-1}$ is a basis for Λ^*. For a full-rank lattice, $\mathbf{B}^* = \mathbf{B}^{-t}$. We refer to $\widetilde{\mathbf{B}}$ as a Gram-Schmidt orthogonalization of $\mathbf{B}$.

So-called *q-ary integer lattices* are of particular interest in cryptography. These lattices satisfy the relation $q\mathbb{Z}^m \subseteq \Lambda \subseteq \mathbb{Z}^m$ for some integer q. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, integers q, m, n, we define two full-rank m-dimensional q-ary lattices:

$$\Lambda(\mathbf{A}^t) = \{\mathbf{y} \in \mathbb{Z}^m : \exists \mathbf{s} \in \mathbb{Z}_q^n \text{ s.t. } \mathbf{y} \equiv \mathbf{A}^t\mathbf{s} \mod q\}$$
$$\Lambda^\perp(\mathbf{A}) = \{\mathbf{y} \in \mathbb{Z}^m : \mathbf{A}\mathbf{y} \equiv 0 \mod q\}.$$

3.2 Gaussians on Lattices

We define the n-dimensional Gaussian function on $\mathbb{R}^n$ centered at 0:

$$\rho(\mathbf{x}) = \exp(-\pi \cdot \|\mathbf{x}\|^2).$$

For any matrix $\mathbf{B}$ we define a density function of a Gaussian distribution for $\mathbf{x} \in \text{span}(\mathbf{B})$ and for $\boldsymbol{\Sigma} = \mathbf{B}\mathbf{B}^t \geq 0$:

$$\rho_{\sqrt{\boldsymbol{\Sigma}}} = \rho(\mathbf{B}^+\mathbf{x}) = \exp(-\pi \cdot \mathbf{x}^t\boldsymbol{\Sigma}^+\mathbf{c}).$$

Normalizing the above expression by its total measure over $\mathrm{span}(\boldsymbol{\Sigma})$, we obtain a probability density function of the continuous Gaussian distribution $D_{\sqrt{\boldsymbol{\Sigma}}}$. The covariance matrix of this distribution is $\frac{\Sigma}{2\pi}$, we ignore the $\frac{1}{2\pi}$ factor and refer to $\boldsymbol{\Sigma}$ as the covariance matrix of $D_{\sqrt{\boldsymbol{\Sigma}}}$.

The continuous Gaussian distribution $D_{\sqrt{\boldsymbol{\Sigma}}}$ can be discretized to a lattice (or to the "shift" of the lattice) as follows: for $\Lambda \subset \mathbb{R}^n, \mathbf{c} \in \mathbb{R}^n$ and positive semidefinite $\boldsymbol{\Sigma} > \mathbf{0}$ such that $(\Lambda + \mathbf{c}) \cap \mathrm{span}(\boldsymbol{\Sigma})$ is nonempty, the *discrete Gaussian distribution* is

$$D_{\Lambda+\mathbf{c},\sqrt{\boldsymbol{\Sigma}}} = \frac{\rho_{\sqrt{\boldsymbol{\Sigma}}}(\mathbf{x})}{\rho_{\sqrt{\boldsymbol{\Sigma}}}(\Lambda + \mathbf{x})}, \forall \mathbf{x} \in \Lambda + \mathbf{c},$$

where the denominator is merely a normalization factor.

In the definition of the so-called *smoothing parameter* η_ϵ (originally defined in [13]) we follow the notion of [15].

Definition 6 *For a positive semidefinite matrix $\boldsymbol{\Sigma}$ and a lattice $\Lambda \subset \mathrm{span}(\boldsymbol{\Sigma})$, we say that $\sqrt{\boldsymbol{\Sigma}} > \eta_\epsilon(\Lambda)$ if $\rho_{\sqrt{\boldsymbol{\Sigma}^+}}(\Lambda^*) \leq 1 + \epsilon$.*

We will also use the following tail bound on discrete Gaussians.

Lemma 7 ([5], Lemma 1.5) *Let $\Lambda \subset \mathbb{R}^n$ be a lattice and $r \leq \eta_\epsilon(\Lambda)$ for some $\epsilon \in (0,1)$. For any $\boldsymbol{c} \in span(\Lambda)$, we have*

$$\Pr[\|D_{\Lambda+\boldsymbol{c},r}\| \geq r\sqrt{n}] \leq 2^{-n} \cdot \frac{1+\epsilon}{1-\epsilon}.$$

If $\boldsymbol{c} = 0$ then the inequality holds for any $r > 0$, with $\epsilon = 0$.

3.3 Useful Tools

Here we recall some useful facts about subgaussian random variable and the singular value of a matrix. A detailed overview on subgaussian probability distribution is given in [17]. As the name suggests, subgaussian random variable generalizes the notion of Gaussian random variable in the sense that it has the property of a super-exponential tail decay.

Definition 8 *A random variable X is subgaussian with parameter s, if $\exists C$ such that $\forall t \geq 0$*

$$\Pr[|X| > t] \leq C\exp(-\pi t^2/s^2).$$

In [17] it is proved that the above definition is equivalent to the inequality for the moment-generating function: $\mathbb{E}[\exp(tX)] \leq \exp(\frac{1}{2}Cs^2t^2)$, $\forall t \in \mathbb{R}$. Since we deal with discrete Gaussians, we will use a more loose definition of the so-called δ-subgaussian variable due to [15]:

Definition 9 *For $\delta > 0$ a random variable X is **δ-subgaussian** with parameter $s > 0$ if for all $t \in \mathbb{R}$, the (scaled) moment-generating function satisfies*

$$\mathbb{E}[2\pi tX] \leq \exp(\delta) \cdot \exp(\pi s^2 t^2).$$

Other than the Gaussian distribution itself, Bernoulli distributed and any bounded random variable are classical examples for subgaussians. Note that if we concatenate independent δ_i-subgaussian random variable with common parameter s into a vector, we obtain a $(\Sigma\delta_i)$ subgaussian vector with parameter s. It is easy to see that for a finite number of independent Gaussian random variables X_i with zero mean, their sum $\Sigma_i X_i$ is a Gaussian random variable with parameter $s = \sqrt{\Sigma_i s_i^2}$. This property is called *rotation invariance* in [17] and also transfers to the subgaussians. In the security proof of our proxy re-encryption scheme we will use the following fact.

Fact 10 *Let $X_1, X_2, \ldots, X_n$ be independent, zero-mean subgaussian random variables with parameter s and $\boldsymbol{a} = (a_1, a_2, \ldots, a_n) \in \mathbb{R}^n$. Then $\Sigma_k(a_k X_k)$ is a subgaussian random variable with parameter $s\|a\|$.*

One can view the addition and subtraction of the subgaussians as the inner product of a subgaussian vector and a $\{0, -1, 1\}$-vector. In our security proof we use this fact to show that the result of a product of a δ-subgaussian matrix (treated as a concatenation of δ-subgaussian columns) by a matrix with $\{0, -1, 1\}$ entries is a δ-subgaussian matrix with a slightly larger parameter s.

Here we recall two facts about the singular values of a random matrix. The first lemma from [17] shows an upper bound on the singular value of the matrix with Gaussian entries adapted to the 0-subgaussian case. The second result ([11]) bounds the singular value of the product of two matrices.

Lemma 11 *Let $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ be a δ-subgaussian random matrix with parameter s. There exist a universal constant $C > 0$ such that for any $t \geq 0$ we have $\sigma_1(\boldsymbol{A}) \leq C \cdot s \cdot (\sqrt{m} + \sqrt{n} + t)$ except with probability at most $2\exp(\delta)\exp(-\pi t^2)$.*

Lemma 12 (Theorem 3.3.16 in [11]) *Let $\boldsymbol{A} \in M_{m,n}, \boldsymbol{B} \in M_{n,m}$ and $\ell = \min\{m, n\}$. The following inequalities hold for the decreasingly ordered singular values of $\boldsymbol{AB}$:*

$$\sigma_i(\boldsymbol{AB}) \leq \sigma_i(\boldsymbol{A})\sigma_1(\boldsymbol{B}) \quad \text{for } i = 1, \ldots, \ell.$$

3.4 Hard problems

There are two lattice-based one-way functions associated with matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ for $m = poly(n)$:

- $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \mod q, \mathbf{x} \in \mathbb{Z}^m$;
- $g_{\mathbf{A}}(\mathbf{e}, \mathbf{s}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \mod q$ for $\mathbf{s} \in \mathbb{Z}_q^n$ and a Gaussian $\mathbf{e} \in \mathbb{Z}^m$.

Given a vector $\mathbf{u}$, finding a short preimage $\mathbf{x}'$ such that $f_{\mathbf{A}}(\mathbf{x}') = \mathbf{u}$ is an instantiation of the SIS problem, which is at least as hard as solving the of Shortest Independent Vector Problem (SIVP) on n-dimensional lattices ([1], [13]). The problem to invert $g_{\mathbf{A}}(\mathbf{e}, \mathbf{s})$, where $\mathbf{e} \leftarrow D_{\alpha q}$, is known as $\mathsf{LWE}_{q,\alpha}$ problem and is as hard as quantumly solving SIVP on n-dimensional lattices ([16]). The decisional-LWE problem asks to distinguish the output of $g_{\mathbf{A}}$ from uniform.

4 G-trapdoor and Algorithms

In this section we briefly describe the main results of [15]: the definition of a so-called **G**-trapdoor and the algorithms $\mathsf{Invert}^{\mathcal{O}}$ and $\mathsf{Sample}^{\mathcal{O}}$ for the LWE and SIS problems.

4.1 Trapdoor generation

In short, a **G**-trapdoor is a transformation (represented by a matrix $\mathbf{R}$) from a public matrix $\mathbf{A}$ to a special matrix $\mathbf{G}$. $\mathbf{G}$ has such a structured form that solving SIS and LWE problems for this matrix (i.e. inverting $g_{\mathbf{G}}$ and $f_{\mathbf{G}}$) can be done efficiently, while for a uniform $\mathbf{A}$ these problems are believed to be hard. As an example of a matrix $\mathbf{G}$ Micciancio and Peikert in [15] consider $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g}^t \in \mathbb{Z}_q^{n \times nk}$, where

$$\mathbf{g}^t = [1 \quad 2 \quad 4 \ldots 2^{k-1}] \in \mathbb{Z}_q^{1 \times k}, \quad q = 2^k.$$

They also give efficient algorithms for inverting $g_{\mathbf{g}}(s, \mathbf{e}) = s \cdot \mathbf{g}^t + \mathbf{e}^t \mod q$ and Gaussian sampling from preimages of $f_{\mathbf{g}}(\mathbf{x}) = \langle \mathbf{g}, \mathbf{x} \rangle \mod q$. By executing these algorithms n times, one solves the same problems for $\mathbf{G}$.

In order to embed this structured matrix into a (uniformly looking) matrix $\mathbf{A}$ together with a transformation $\mathbf{R}$, one should start with a uniform matrix $\mathbf{A}_0$ and a matrix $\mathbf{R}$ and construct $\mathbf{A} = [\mathbf{A}_0 | -\mathbf{A}_0\mathbf{R} + \mathbf{G}]$. For an appropriate choice of dimensions $(\mathbf{A}, \mathbf{AR})$ is $\mathsf{negl}(n)$-far from uniform by the Leftover Hash Lemma. Using $\mathbf{R}$ one can transform:

$$[\mathbf{A}_0 | -\mathbf{A}_0\mathbf{R} + \mathbf{G}] \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{G}$$

and, therefore, invert one-way functions $g_{\mathbf{A}}$, $f_{\mathbf{A}}$.

In [15] an invertible matrix $\mathbf{H}$ is used as: $\mathbf{A} = [\mathbf{A}_0 | -\mathbf{A}_0\mathbf{R} + \mathbf{HG}]$ to construct a CCA-secure encryption scheme. In this case the knowledge of both $\mathbf{R}$ and $\mathbf{H}$ is needed to perform the transformation. Note, that if $\mathbf{H}$ is a zero-matrix, then $[\mathbf{A}_0 | -\mathbf{A}_0\mathbf{R}] \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{0}$, and solving LWE (or SIS) for $\mathbf{A}$ does not longer reduce to solving the same problems for $\mathbf{G}$. This fact is used to construct a challenge ciphertext.

To achieve CCA-security for *re-encryption*, we need to have a pair of transformations ($\mathbf{R}_1$, $\mathbf{R}_2$) for $\mathbf{A}$: $\mathbf{R}_1$ to generate re-encryption keys (i.e. to solve SIS) and $\mathbf{R}_2$ to decrypt (i.e. to solve LWE). Let us define the generalized definition of a **G**-trapdoor:

Definition 13 *Let* $\mathbf{A} = [\mathbf{A}_0 | \mathbf{A}_1 | \ldots | \mathbf{A}_{k-1}] \in \mathbb{Z}_q^{n \times m}$ *for* $k \geq 2$*, and* $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times \bar{m}}$*,* $\mathbf{A}_1, \ldots, \mathbf{A}_{k-1} \in \mathbb{Z}_q^{n \times w}$ *with* $\bar{m} \geq w \geq n$ *and* $m = \bar{m} + (k-1) \cdot w$ *(typically,* $w = n \lceil \log q \rceil$ *). A* **G***-trapdoor for* ***A*** *is a sequence of matrices* $\mathbf{R} = [\mathbf{R}_1 | \mathbf{R}_2 | \ldots | \mathbf{R}_{k-1}] \in \mathbb{Z}^{\bar{m} \times (k-1)w}$ *such that:*

$$[\mathbf{A}_0|\mathbf{A}_1|\dots|\mathbf{A}_{k-1}]\begin{bmatrix}\mathbf{R}_1 & \mathbf{R}_2 & \cdots & \mathbf{R}_{k-1}\\ \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0}\\ \vdots & \vdots & \ddots & \vdots\\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I}\end{bmatrix} = [\mathbf{H}_1\mathbf{G}|\mathbf{H}_2\mathbf{G}|\cdots|\mathbf{H}_{k-1}\mathbf{G}]$$

for invertible matrices $\mathbf{H}_i \in \mathbb{Z}_q^{n\times n}$ *and a fixed* $\mathbf{G} \in \mathbb{Z}_q^{n\times w}$.

To generate a pseudorandom matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ with a $\mathbf{G}$-trapdoor $\mathbf{R} \in \mathbb{Z}_q^{\bar{m}\times(k-1)w}$ one should iteratively execute the algorithm $\mathsf{GenTrap}^{\mathcal{D}}$ of [15] but for $k-1$ invertible $\mathbf{H}_i$'s and for Gaussian $\mathbf{R}_i \leftarrow D_s^{\bar{m}\times w}$ for some $s \geq \eta_\epsilon(\mathbb{Z})$.

4.2 Algorithms

Here we show how to use a generalized trapdoor for the inversion of the function $g_{\mathbf{A}}(\mathbf{s},\mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \mod q$ and preimage sampling for $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \mod q$, where $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ has a trapdoor $\mathbf{R} \in \mathbb{Z}^{(m-w)\times w}$ that satisfies Def. 13. The algorithms below generalize the algorithms $\mathsf{Invert}^{\mathcal{O}}$ and $\mathsf{Sample}^{\mathcal{O}}$ of [15].

LWE Inversion. We start by showing how to use the extended notion of a $\mathbf{G}$-trapdoor to invert an LWE sample $\mathbf{b}^t = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \mod q$. We refer to this procedure as $\mathsf{Invert}^{\mathcal{O}}(\mathbf{R},\mathbf{A},\mathbf{b},\mathbf{H}_i)$ and emphasize in the input that $\mathbf{H}_i$ is an invertible matrix, while the other $\mathbf{H}_j, j \neq i$ can be zero. So for $\mathbf{A} = [\mathbf{A}_0|\mathbf{H}_1\mathbf{G} - \mathbf{R}_1\mathbf{A}_0|...|\mathbf{H}_i\mathbf{G} - \mathbf{R}_i\mathbf{A}_0|...]$:

1. Compute $\widehat{\mathbf{b}}^t = \mathbf{b}\begin{bmatrix}\mathbf{R}_1 & \mathbf{R}_2 & \cdots & \mathbf{R}_{k-1}\\ \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0}\\ \vdots & \vdots & \ddots & \vdots\\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I}\end{bmatrix} = [\mathbf{H}_1\mathbf{G}|...|\mathbf{H}_i\mathbf{G}|...\mathbf{H}_{k-1}\mathbf{G}] \in \mathbb{Z}_q^{(k-1)w}$;
2. Set $\widehat{\mathbf{b}}_1^t = \widehat{\mathbf{b}}^t[w\cdot(i-1)...w\cdot i]$;
3. Obtain $(\hat{\mathbf{s}},\hat{\mathbf{e}})$ by inverting $\widehat{\mathbf{b}}_1^t$ for $\mathbf{G}$. So $(\hat{\mathbf{s}},\hat{\mathbf{e}})$ satisfies $\widehat{\mathbf{b}}_1^t = \hat{\mathbf{s}}\mathbf{G} + \hat{\mathbf{e}} \mod q$.
4. Compute $\mathbf{s} = \mathbf{H}_i^{-t}\hat{\mathbf{s}} \in \mathbb{Z}_q^n$ and $\mathbf{e} = \mathbf{b} - \mathbf{A}^t\mathbf{s} \in \mathbb{Z}_q^m$. Output $(\mathbf{s},\mathbf{e})$.

The algorithm produces a correct output, if the error vector $\mathbf{e}$ is "short enough": $\|\mathbf{e}\| < q/(2\|\mathbf{B}\|s)$ where $\mathbf{B}$ is a basis for $\Lambda^\perp(\mathbf{G})$ and $s = \sqrt{\sigma_1(\mathbf{R}_i)^2 + 1}$. For the detailed proof of correctness see theorem 5.4 in [15].

Gaussian Sampling. We show below how to sample a Gaussian vector $\mathbf{x} \in \mathbb{Z}_q^m$ for a matrix $\mathbf{A} = [\mathbf{A}_0|\dots|\mathbf{A}_{k-1}] \in \mathbb{Z}_q^{n\times m}$ with the generalized trapdoor $\mathbf{R} \in \mathbb{Z}_q^{\bar{m}\times(k-1)w}$ and $k-1$ invertible $\mathbf{H}_i$'s given a coset $\mathbf{u} \in \mathbb{Z}_q^n$.

The intuition behind the algorithm $\mathsf{Sample}^{\mathcal{O}}$ of [15] is the following: for two distributions X and Y with covariance matrices $\boldsymbol{\Sigma}_X$ and $\boldsymbol{\Sigma}_Y$, the covariance of their sum is $\boldsymbol{\Sigma}_X + \boldsymbol{\Sigma}_Y$. A spherical Gaussian distribution with a standard deviation s has covariance matrix $s^2\mathbf{I}$. Therefore, having $\boldsymbol{\Sigma}_X$ and a parameter s as inputs, we can "adjust" $\boldsymbol{\Sigma}_Y$ such that $X + Y$ is a spherical Gaussian with standard deviation s.

So having as input a coset $\mathbf{u}$, a matrix $\mathbf{A}$ with a trapdoor $\mathbf{R}$, an invertible $\mathbf{H}$ and a parameter for the output distribution s, we first sample a vector $\mathbf{z}$ for the matrix $\mathbf{G}$ with fixed parameter $r\sqrt{\boldsymbol{\Sigma}_\mathbf{G}}$ (for the construction from section 4.1 $\sqrt{\boldsymbol{\Sigma}_\mathbf{G}} = 2$). Then we multiply $\mathbf{x}' = \begin{bmatrix}\mathbf{R}\\ \mathbf{I}\end{bmatrix}\mathbf{z}$. The resulting vector $\mathbf{x}'$ satisfies $\mathbf{A}\mathbf{x}' = \mathbf{u}$, but has covariance matrix $\boldsymbol{\Sigma}_{\mathbf{x}'} = [\mathbf{R}|\mathbf{I}]^T(r\boldsymbol{\Sigma}_\mathbf{G})[\mathbf{R}^t|\mathbf{I}]$. In order to output a vector $\mathbf{x}$ as spherical Gaussian with parameter s we add a vector $\mathbf{p}$ with covariance $\boldsymbol{\Sigma}_\mathbf{p} = s^2\mathbf{I} - [\mathbf{R}|\mathbf{I}]^T(r\boldsymbol{\Sigma}_\mathbf{G})[\mathbf{R}^t|\mathbf{I}]$.

1. Choose $\mathbf{p} \leftarrow D_{\mathbb{Z}^m, r\sqrt{\boldsymbol{\Sigma}_\mathbf{p}}}$. View it as $\mathbf{p}^t = [\mathbf{p}_1|\mathbf{p}_2|\dots|\mathbf{p}_k]$, where $\mathbf{p}_1 \in \mathbb{Z}^{\bar{m}}$, $\mathbf{p}_2, \dots, \mathbf{p}_k \in \mathbb{Z}^w$.
2. Compute $\overline{\mathbf{w}}_1 = \mathbf{A}_0(\mathbf{p}_1 - \mathbf{R}_1\mathbf{p}_2)$, $\overline{\mathbf{w}}_i = -\mathbf{A}_0\mathbf{R}_i\mathbf{p}_{i+1}$ for $i = 2, \dots, (k-1)$;
3. Compute $\mathbf{w}_i = \mathbf{G}\mathbf{p}_{i+1}$ for $i = 1, \dots, (k-1)$;
4. Let $\mathbf{v}_1 = \mathbf{H}_1^{-1}(\mathbf{u} - \overline{\mathbf{w}}_1) - \mathbf{w}_1$, $\mathbf{v}_i = -\mathbf{H}_i^{-1}\overline{\mathbf{w}}_i - \mathbf{w}_i$ for $i = 2, \dots, (k-1)$;
5. For each $i = 1, \dots, (k-1)$ choose $\mathbf{z}_i \leftarrow D_{\Lambda^\perp_{\mathbf{v}_i}(\mathbf{G}), r\sqrt{\boldsymbol{\Sigma}_\mathbf{G}}}$. Concatenate the obtained vectors to get $\mathbf{z}^t = [\mathbf{z}_1|\dots|\mathbf{z}_{k-1}] \in \mathbb{Z}^{(k-1)w}$;
6. Output $\mathbf{x} = \mathbf{p} + \begin{bmatrix}\mathbf{R}_1 & \mathbf{R}_2 & \cdots & \mathbf{R}_{k-1}\\ \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0}\\ \vdots & \vdots & \ddots & \vdots\\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I}\end{bmatrix} \cdot \mathbf{z} \in \mathbb{Z}^m$.

5 Chosen Ciphertext Secure Proxy Re-Encryption

In this section we present our main result: the proxy re-encryption scheme that employs a $\mathbf{G}$-trapdoor with the associated algorithms from the previous section. Before giving the formal description of the scheme we provide an intuition behind the generation of the re-encryption keys. The ability to sample short vectors for any coset can be extended to performing the sampling algorithm for any $n \times m$ matrix in a column-wise fashion, that is for each column (coset) we can output a Gaussian column-vector and after m samplings concatenate the result into a matrix. This idea was used in a trapdoor delegation algorithm in [15]. If the input matrix is some public key matrix $\mathbf{A}'$, then the result of sampling (a matrix $\mathbf{X}$) is a transformation $\mathbf{A} \cdot \mathbf{X} = \mathbf{A}'$ between two public keys.

In order to accomplish both tasks: re-encyption key generation and decryption, we propose to use the generalized definition of a $\mathbf{G}$-trapdoor (Def. 13). Thus, we achieve a re-encryption functionality: we sample small matrices with one $\mathbf{G}$-trapdoor ($\mathbf{R}_1$), and at the same time we perform the decryption operation using another $\mathbf{G}$-trapdoor ($\mathbf{R}_2$).

5.1 Construction of the single-hop PRE

Let 1^n be the security parameter and let r refer to a fixed function $w(\sqrt{\log n})$.

– The modulus q is defined as a large enough prime power $q = p^e = poly(n)$ and $k = O(\log q) = O(\log n)$. We define $\bar{m} = O(nk)$ and the total dimension of the public key as $m = \bar{m} + 2nk$.

- $\mathbf{G} \in \mathbb{Z}_q^{n\times nk}$ is a matrix of a special structure (see section 4.2 for an example), so there are efficient algorithms to invert $g_{\mathbf{G}}$ and to sample for $f_{\mathbf{G}}$.
- the trapdoors $\mathbf{R}_i$'s are sampled from the Gaussian $\mathcal{D} = D^{\bar{m}\times nk}_{\mathbb{Z},w(\sqrt{\log n})}$, so that $(\mathbf{A}_0, \mathbf{A}_0\mathbf{R}_1, \mathbf{A}_0\mathbf{R}_2)$ is $\mathsf{negl}(n)$-far from uniformly chosen matrices $(\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3) \in \mathbb{Z}_q^{n\times\bar{m}} \times \mathbb{Z}_q^{n\times nk} \times \mathbb{Z}_q^{n\times nk}$ for $\mathbf{A}_0 \leftarrow \mathbb{Z}_q^{n\times\bar{m}}$ and for any $\mathbf{R}_1, \mathbf{R}_2 \leftarrow \mathcal{D}$.
- All the invertible matrices $\mathbf{H} \in \mathbb{Z}_q^{n\times n}$ that are used in the scheme, are chosen from a set with the "unit differences" property (see [15] for an example): for any two $\mathbf{H}', \mathbf{H}''$ their difference $\mathbf{H}' - \mathbf{H}''$ is also invertible.
- the LWE error rate α for single-hop PRE should satisfy $1/\alpha = O((nk)^3)\cdot r^3$.

We encode the message space $\{0,1\}^{nk}$ to the cosets of $\Lambda/2\Lambda$ for the lattice $\Lambda = \Lambda(\mathbf{G}^t)$ using any basis $\mathbf{B} \in \mathbb{Z}^{nk}$ of Λ, namely for a message $\mathbf{m} \in \{0,1\}^{nk}$ we define the encoding function as $\mathsf{enc}(\mathbf{m}) = \mathbf{Bm} \in \mathbb{Z}^{nk}$. Notice that this mapping can be efficiently inverted.

- $\mathsf{KeyGen}(1^n)$: choose $\mathbf{A}_0 \leftarrow \mathbb{Z}_q^{n\times\bar{m}}$, $\mathbf{R}_1, \mathbf{R}_2 \leftarrow \mathcal{D}$ and an invertible matrix $\mathbf{H} \leftarrow \mathbb{Z}_q^{nk\times nk}$. Compose the matrix $\mathbf{A} = [\mathbf{A}_0|\mathbf{A}_1|\mathbf{A}_2] = [\mathbf{A}_0| - \mathbf{A}_0\mathbf{R}_1| - \mathbf{A}_0\mathbf{R}_2] \in \mathbb{Z}_q^{n\times m}$ and set the public key as $\mathsf{pk} = (\mathbf{A}, \mathbf{H})$. The secret key is the matrix $sk = [\mathbf{R}_1|\mathbf{R}_2] \in \mathbb{Z}^{\bar{m}\times 2nk}$ with small entries. Notice that
$$\left[\mathbf{A}_0|\mathbf{A}_1|\mathbf{A}_2\right]\begin{bmatrix}\mathbf{R}_1 & \mathbf{R}_2\\ \mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{bmatrix} = \left[\mathbf{0}|\mathbf{0}\right] \in \mathbb{Z}_q^{n\times 2nk}.$$
- $\mathsf{Enc}(\mathsf{pk} = ([\mathbf{A}_0|\mathbf{A}_1|\mathbf{A}_2],\ \mathbf{H}), \mathbf{m} \in \{0,1\}^{nk})$: choose a non-zero invertible matrix $\mathbf{H}_u$, and a vector $\mathbf{s} \leftarrow \mathbb{Z}_q^n$. Set $\mathbf{A}_u = [\mathbf{A}_0|\mathbf{A}_1 + \mathbf{HG}|\mathbf{A}_2 + \mathbf{H}_u\mathbf{G}]$. Sample three error vectors $\mathbf{e}_0 \leftarrow D^{\bar{m}}_{\mathbb{Z},\alpha q}$, $\mathbf{e}_1, \mathbf{e}_2 \leftarrow D^{nk}_{\mathbb{Z},s}$, where $s^2 = (\|\mathbf{e}_0\|^2 + \bar{m}(\alpha q)^2)\cdot r^2$. The composed error vector is a concatenation of the chosen vectors $\mathbf{e} = (\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2) \in \mathbb{Z}^m$. Compute
$$\mathbf{b}^t = 2(\mathbf{s}^t[\mathbf{A}_0|\mathbf{A}_1 + \mathbf{HG}|\mathbf{A}_2 + \mathbf{H}_u\mathbf{G}] \bmod q) + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{enc}(\mathbf{m}))^t \bmod 2q,$$
where the first zero vector has dimension $\bar{m}$, the second has dimension nk. Output the ciphertext $c = (\mathbf{H}_u, \mathbf{b}) \in \mathbb{Z}_q^{n\times n} \times \mathbb{Z}_{2q}^m$.
- $\mathsf{Dec}(\mathsf{pk} = ([\mathbf{A}_0|\mathbf{A}_1|\mathbf{A}_2],\ \mathbf{H}), \mathsf{sk} = [\mathbf{R}_1|\mathbf{R}_2], c = (\mathbf{H}_u, \mathbf{b}))$: Using matrix $\mathbf{H}_u$ compute $\mathbf{A}_u = [\mathbf{A}_0|\mathbf{A}_1 + \mathbf{HG}|\mathbf{A}_2 + \mathbf{H}_u\mathbf{G}]$.
  1. If c has invalid form or $\mathbf{H}_u = \mathbf{0}$, output $\perp$.
  2. With the secret key call an algorithm $\mathsf{Invert}^{\mathcal{O}}([\mathbf{R}_1|\mathbf{R}_2],\ \mathbf{A}_u, \mathbf{b}, \mathbf{H}_u)$. On this input the algorithm (section 4.2) computes the product:
  $$\left[\mathbf{A}_0|\mathbf{A}_1 + \mathbf{HG}| - \mathbf{A}_2 + \mathbf{H}_u\mathbf{G}\right]\begin{bmatrix}\mathbf{R}_1 & \mathbf{R}_2\\ \mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{bmatrix} = \left[\mathbf{HG}|\mathbf{H}_u\mathbf{G}\right] \in \mathbb{Z}_q^{n\times 2nk}.$$
  As output we receive two vectors $\mathbf{z} \in \mathbb{Z}_q^n$ and $\mathbf{e} = (\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2) \in \mathbb{Z}_q^{\bar{m}} \times \mathbb{Z}_q^{nk} \times \mathbb{Z}_q^{nk}$ that satisfy $\mathbf{b}^t = \mathbf{z}^t\mathbf{A}_u + \mathbf{e}^t \mod q$.

3. Check the lengths of the obtained vectors, namely, if $\|\mathbf{e}_0\| \geq \alpha q\sqrt{\bar{m}}$ or $\|\mathbf{e}_j\| \geq \alpha q\sqrt{2\bar{m}nk} \cdot w(\sqrt{\log n})$ for $j = 1, 2$, output $\perp$.
4. Parse $\mathbf{v} = \mathbf{b} - \mathbf{e} \mod 2q$ as $\mathbf{v} = (\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2) \in \mathbb{Z}_{2q}^{\bar{m}} \times \mathbb{Z}_{2q}^{nk} \times \mathbb{Z}_{2q}^{nk}$. If $\mathbf{v}_0 \notin 2\Lambda(\mathbf{A}_0^t)$, output $\perp$. Otherwise, compute

$$\mathbf{v}^t \begin{bmatrix} \mathbf{R}_1 & \mathbf{R}_2 \\ \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \mod 2q \in \mathbb{Z}_{2q}^{nk}$$

and apply enc^{-1} to the last nk coordinates.

– $\mathsf{ReKeyGen}(\mathsf{pk} = ([\mathbf{A}_0|\mathbf{A}_1|\mathbf{A}_2],\ \mathbf{H}), \mathsf{sk} = [\mathbf{R}_1|\mathbf{R}_2], \mathsf{pk}' = ([\mathbf{A}'_0|\mathbf{A}'_1|\mathbf{A}'_2],\ \mathbf{H}'))$:

1. Using the first part of a secret key - the Gaussian matrix $\mathbf{R}_1$ - and the invertible $\mathbf{H} \in \mathbb{Z}_q^{n\times n}$ from the public key, execute $\mathsf{Sample}^{\mathcal{O}}$ (section 4.2) to sample from the cosets of the $\mathbf{A}'_0$. Specifically, we sample column-wise so that for each column of the $\mathbf{A}'_0$ we obtain an $\bar{m} + nk$-dimensional column of the re-encryption key. After sampling $\bar{m}$ times we receive an $(\bar{m} + nk) \times \bar{m}$ matrix and parse it as two matrices $\mathbf{X}_{00} \in \mathbb{Z}^{\bar{m}\times\bar{m}}$ and $\mathbf{X}_{10} \in \mathbb{Z}^{nk\times\bar{m}}$ matrices with Gaussian entries of parameter s.

$$\left[\mathbf{A}_0 | -\mathbf{A}_0\mathbf{R}_1 + \mathbf{H}\mathbf{G}\right] \begin{bmatrix} \mathbf{X}_{00} \\ \mathbf{X}_{10} \end{bmatrix} = \left[\mathbf{A}'_0\right].$$

2. Continue sampling for the cosets obtained from the columns of the matrix $[\mathbf{A}'_1 + \mathbf{H}'\mathbf{G}]$ of pk'. But this time we increase (as explained in the Gaussian sampling algorithms in section 4.2), the Gaussian parameter of the resulting sampled matrix up to $s\sqrt{\bar{m}/2}$

$$\left[\mathbf{A}_0 | -\mathbf{A}_0\mathbf{R}_1 + \mathbf{H}\mathbf{G}\right] \begin{bmatrix} \mathbf{X}_{01} \\ \mathbf{X}_{11} \end{bmatrix} = \left[\mathbf{A}'_1 + \mathbf{H}'\mathbf{G}\right].$$

To achieve a correct re-encryption for the last sampling change the cosets by adding $-\mathbf{A}_2 = \mathbf{A}_0\mathbf{R}_2$:

$$\left[\mathbf{A}_0 | -\mathbf{A}_0\mathbf{R}_1 + \mathbf{H}\mathbf{G}\right] \begin{bmatrix} \mathbf{X}_{02} \\ \mathbf{X}_{12} \end{bmatrix} = \left[\mathbf{A}'_2 + \mathbf{A}_0\mathbf{R}_2\right],$$

where $\mathbf{X}_{01}, \mathbf{X}_{02} \in \mathbb{Z}^{\bar{m}\times nk}$, $\mathbf{X}_{11}, \mathbf{X}_{12} \in \mathbb{Z}^{nk\times nk}$ with entries distributed as Gaussian with parameter $s\sqrt{\bar{m}}$.

3. The re-encryption key is a matrix with Gaussian entries:

$$\mathsf{rk} = \begin{bmatrix} \mathbf{X}_{00} & \mathbf{X}_{01} & \mathbf{X}_{02} \\ \mathbf{X}_{10} & \mathbf{X}_{11} & \mathbf{X}_{12} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \in \mathbb{Z}^{m\times m}.$$

For any matrix $\mathbf{B} \in \mathbb{Z}^{n\times nk}$ the re-encryption key satisfies:

$$[\mathbf{A}_0|\mathbf{A}_1 + \mathbf{H}\mathbf{G}|\mathbf{A}_2 + \mathbf{B}] \cdot \begin{bmatrix} \mathbf{X}_{00} & \mathbf{X}_{01} & \mathbf{X}_{02} \\ \mathbf{X}_{10} & \mathbf{X}_{11} & \mathbf{X}_{12} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} = [\mathbf{A}'_0|\mathbf{A}'_1 + \mathbf{H}'\mathbf{G}|\mathbf{A}'_2 + \mathbf{B}] \quad (1)$$

– $\mathsf{ReEnc}(\mathsf{rk}, c = (\mathbf{H}_u, \mathbf{b}))$: to change the underlying public key in the ciphertext component $\mathbf{b}$ compute $\mathbf{b}'^t =$

$$\mathbf{b}^t \cdot \mathsf{rk} = \mathbf{s}^t[\mathbf{A}'_0|\mathbf{A}'_1 + \mathbf{H}'\mathbf{G}|\mathbf{A}'_2 + \mathbf{H}_u\mathbf{G}] + \widetilde{\mathbf{e}}^t + (\mathbf{0}, \mathbf{0}, \mathsf{enc}(\mathbf{m}))^t \bmod 2q, \quad (2)$$

where $\widetilde{\mathbf{e}} = (\widetilde{\mathbf{e}}_0, \widetilde{\mathbf{e}}_1, \widetilde{\mathbf{e}}_2)$ and $\widetilde{\mathbf{e}}_0 = \mathbf{e}_0\mathbf{X}_{00} + \mathbf{e}_1\mathbf{X}_{10}, \widetilde{\mathbf{e}_1} = \mathbf{e}_0\mathbf{X}_{01} + \mathbf{e}_1\mathbf{X}_{11}, \widetilde{\mathbf{e}_2} = \mathbf{e}_0\mathbf{X}_{02} + \mathbf{e}_1\mathbf{X}_{12} + \mathbf{e}_2$. Finally, output $c' = (\mathbf{H}_u, \mathbf{b}')$.

Remark 14 *Instead of mapping a message $\boldsymbol{m} \in \{0,1\}^{nk}$ to the lattice cosets, one can use a more common encoding for lattice-based schemes: $\mathsf{enc}(\boldsymbol{m}) = \boldsymbol{m}\lfloor \frac{q}{2} \rfloor$. In this case one should add an extra syndrome $\boldsymbol{A}_3 \in \mathbb{Z}_q^{n\times nk}$ to the public key and sample one more error vector $\boldsymbol{e}_3 \in\leftarrow D_{\mathbb{Z},\alpha q}^{\overline{m}}$ for the encryption, so a ciphertext is of the form $\boldsymbol{b}^t = \boldsymbol{s}^t[\boldsymbol{A}_0|\boldsymbol{A}_1 + \boldsymbol{HG}|\boldsymbol{A}_2 + \boldsymbol{H}_u\boldsymbol{G}|\boldsymbol{A}_3] + (\boldsymbol{e}_0, \boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3)^t + (\boldsymbol{0}, \boldsymbol{0}, \boldsymbol{0}, \mathsf{enc}(\boldsymbol{m}))^t \bmod q$. For the re-encryption key generation one more sampling is needed: $\left[\boldsymbol{A}_0| - \boldsymbol{A}_0\boldsymbol{R}_1 + \boldsymbol{HG}\right]\begin{bmatrix}\boldsymbol{X}_{03}\\ \boldsymbol{X}_{13}\end{bmatrix} = \left[\boldsymbol{A}'_3 - \boldsymbol{A}_3\right]$, which results in an extended (columns $\begin{bmatrix}\boldsymbol{X}_{03}\\ \boldsymbol{X}_{13}\end{bmatrix}$ are added) re-encryption key. All the arguments below on correctness and security can be easily adapted for this case.*

5.2 Correctness

In the re-encrypted ciphertext the error terms are larger than in the original ciphertext (that is the ones that have not been re-encrypted). In the following lemma we show that with the appropriate choice of the LWE parameters α and q the decryption algorithm can tolerate the noise growth.

Lemma 15 *Our PRE scheme with message space $\mathcal{M} = \{0,1\}^{nk}$ decrypts correctly.*

Proof. Recall that by Definition 3 we have to show that the decryption algorithm outputs a correct plaintext both for the original and for the re-encrypted ciphertext. So first of all we describe an original encryption under a public key pk and then proceed with its re-encryption to another public key pk'.

Let $\mathsf{pk} = ([\mathbf{A}_0|\mathbf{A}_1|\mathbf{A}_2], \mathbf{H}), \mathsf{pk}' = ([\mathbf{A}'_0|\mathbf{A}'_1|\mathbf{A}'_2], \mathbf{H}')$ be the public keys output by $\mathsf{KeyGen}(1^n)$ together with two trapdoors $\mathsf{sk} = [\mathbf{R}_1|\mathbf{R}_2]$, $\mathsf{sk}' = [\mathbf{R}'_1|\mathbf{R}'_2]$, so the first pair $(\mathsf{pk}, \mathsf{sk})$ will be the delegator's keys, the second $(\mathsf{pk}', \mathsf{sk}')$ will be for the delegatee. We run the $\mathsf{ReKeyGen}(\mathsf{pk}, \mathsf{sk}, \mathsf{pk}')$ algorithm to obtain

$$\mathsf{rk}_{\mathsf{pk}\to\mathsf{pk}'} = \begin{bmatrix}\mathbf{X}_{00} & \mathbf{X}_{01} & \mathbf{X}_{02}\\ \mathbf{X}_{10} & \mathbf{X}_{11} & \mathbf{X}_{12}\\ \mathbf{0} & \mathbf{0} & \mathbf{I}\end{bmatrix}.$$

As we apply this re-encryption key to a ciphertext $\mathbf{b}$ that encrypts a message $\mathbf{m} \in \{0,1\}^{nk}$ under the delegator's public key $\mathsf{pk} = [\mathbf{A}_0|\mathbf{A}_1|\mathbf{A}_2], \mathbf{H}$ with the invertible matrix $\mathbf{H}_u$ and $\mathbf{e} = (\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2)$, we have $\mathbf{b}^t \cdot \mathsf{rk}_{\mathsf{pk}\to\mathsf{pk}'} =$

$$(2\mathbf{s}^t[\mathbf{A}_0|\mathbf{A}_1 + \mathbf{HG}|\mathbf{A}_2 + \mathbf{H}_u\mathbf{G}] + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{enc}(\mathbf{m})^t) \cdot \mathsf{rk}_{\mathsf{pk}\to\mathsf{pk}'}$$

$$= \mathbf{s}^t[\mathbf{A}_0\mathbf{X}_{00} + (\mathbf{A}_1 + \mathbf{H}\mathbf{G})\mathbf{X}_{10}|\mathbf{A}_0\mathbf{X}_{01} + \mathbf{A}_1\mathbf{X}_{11}|\mathbf{A}_0\mathbf{X}_{02} + \mathbf{A}_1\mathbf{X}_{12} + \mathbf{A}_2 + \mathbf{H}_u\mathbf{G}]+$$

$$(\mathbf{e}_0\mathbf{X}_{00} + \mathbf{e}_1\mathbf{X}_{10}, \mathbf{e}_0\mathbf{X}_{02} + \mathbf{e}_1\mathbf{X}_{11}, \mathbf{e}_0\mathbf{X}_{02} + \mathbf{e}_1\mathbf{X}_{12} + \mathbf{e}_2)^t + (\mathbf{0}, \mathbf{0}, \mathsf{enc}(\mathbf{m}))^t =$$

$$= \mathbf{s}^t[\mathbf{A}'_0|\mathbf{A}'_1\mathbf{H}'\mathbf{G}|\mathbf{A}'_2 + \mathbf{H}_u\mathbf{G}] + (\widetilde{\mathbf{e}}_0, \widetilde{\mathbf{e}}_1, \widetilde{\mathbf{e}}_2)^t + (\mathbf{0}, \mathbf{0}, \mathsf{enc}(\mathbf{m}))^t,$$

where the last equation follows from Eq.(1) of the re-encryption key $\mathsf{rk}_{pk\to\mathsf{pk}'}$.

Now we estimate how the re-encryption algorithm affects the noise and justify the correctness of the decryption for a re-encrypted ciphertext. The arguments for the original ciphertexts are essentially the same as in [Lemma 6.2] of [15].

In the decryption procedure we multiply a (re-encrypted) ciphertext (and thus its error term) by $\mathsf{sk}' = [\mathbf{R}'_1|\mathbf{R}'_2]$ padded with the identity matrix. So in order to obtain a correct output, we require that in the Eq.(2) both terms $\widetilde{\mathbf{e}}_0\mathbf{R}'_1 + \widetilde{\mathbf{e}}_1$ and $\widetilde{\mathbf{e}}_0\mathbf{R}'_2 + \widetilde{\mathbf{e}}_2$ satisfy the length condition of the decryption algorithm: $\widetilde{\mathbf{e}}_0\mathbf{R}'_1 + \widetilde{\mathbf{e}}_1$, $\widetilde{\mathbf{e}}_0\mathbf{R}'_2 + \widetilde{\mathbf{e}}_2 \in \mathcal{P}_{1/2}(q \cdot \mathbf{B}^{-t})$. The terms expand under the multiplication as

$$\widetilde{\mathbf{e}}_0\mathbf{R}'_1 + \widetilde{\mathbf{e}}_1 = \mathbf{e}_0\mathbf{X}_{00}\mathbf{R}'_1 + \mathbf{e}_1\mathbf{X}_{10}\mathbf{R}'_1 + \mathbf{e}_0\mathbf{X}_{01} + \mathbf{e}_1\mathbf{X}_{11}, \tag{3a}$$

$$\widetilde{\mathbf{e}}_0\mathbf{R}'_2 + \widetilde{\mathbf{e}}_2 = \mathbf{e}_0\mathbf{X}_{00}\mathbf{R}'_2 + \mathbf{e}_1\mathbf{X}_{10}\mathbf{R}'_2 + \mathbf{e}_0\mathbf{X}_{02} + \mathbf{e}_1\mathbf{X}_{12} + \mathbf{e}_2, \tag{3b}$$

where $(\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2) \in \mathbb{Z}^m$ is the error vector of the original ciphertext $\mathbf{b}$.

Since we are interested in upper bounds for the length of (3a) and (3b), we should estimate how the length of the Gaussian vectors $\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2$ is affected by the matrix multiplication. We analyze each term of Eq.(3b) separately (the same arguments hold for the terms of (3a)).

According to the sampling algorithm, the parameter s for each column of the $\mathbf{X}_{00}$ (and of $\mathbf{X}_{10}$) is as small as $\sqrt{\sigma_1(\mathbf{R}_1)^2 + 1} \cdot \sqrt{\sigma_1(\Sigma_{\mathbf{G}}) + 2} \cdot r$, where $\mathbf{R}_1$ is the trapdoor that was used in the re-encryption key generation. Combining Lemmas 11 and 12 and the fact that for the $\mathbf{G}$ matrix $\sigma_1(\Sigma_{\mathbf{G}}) = 4$, we obtain

$$\sigma_1(\mathbf{X}_{00}\mathbf{R}'_2) \le \sigma_1(\mathbf{X}_{00}) \cdot \sigma_1(\mathbf{R}'_2) \le C \cdot 4\sqrt{6}\bar{m} \cdot \sqrt{\sigma_1(\mathbf{R}_1)^2 + 1} \cdot r^2,$$

where $C \approx 1/2\pi$. By Lemma 7, we have $\|\mathbf{e}_0\| < \alpha q\sqrt{\bar{m}}$ and therefore

$$\|\mathbf{e}_0\mathbf{X}_{00}\mathbf{R}'_2\| < \alpha q\frac{2\sqrt{6}}{\pi}\bar{m}^{3/2}\sqrt{\sigma_1(\mathbf{R}_1)^2 + 1} \cdot r.$$

Both $\mathbf{e}_1, \mathbf{e}_2$ are sampled from the Gaussian distribution with parameter s, where $s^2 = (\|\mathbf{e}_0\|^2 + \bar{m}(\alpha q)^2) \cdot r^2$, so their lengths are bounded as $\|\mathbf{e}_1\|, \|\mathbf{e}_2\| < \alpha q\sqrt{2\bar{m}nk} \cdot r$. Hence, for the second term of Eq. (3b) it holds that

$$\|\mathbf{e}_1\mathbf{X}_{10}\mathbf{R}'_2\| < \alpha q\frac{3\sqrt{6}}{\pi}\bar{m}\sqrt{2\bar{m}nk} \cdot \sqrt{\sigma_1(\mathbf{R}_1)^2 + 1} \cdot r^3.$$

Now we analyze the singular value for matrix $\mathbf{X}_{02}$ that was sampled with parameter $s\sqrt{\bar{m}/2}$ (the same holds for $\mathbf{X}_{01}, \mathbf{X}_{12}, \mathbf{X}_{11}$):

$$\sigma_1(\mathbf{X}_{02}) \le 2\sqrt{3}\bar{m} \cdot \sqrt{\sigma_1(\mathbf{R}_1)^2 + 1} \cdot r,$$

which implies $\|\mathbf{e}_0\ \mathbf{X}_{02}\| \leq 2\sqrt{3}\alpha q\bar{m} \cdot \sqrt{\sigma_1(\mathbf{R}_1)^2+1} \cdot r$ and $\|\mathbf{e}_1\mathbf{X}_{12}\| \leq \sqrt{2}\alpha q\bar{m} \cdot \sqrt{2\bar{m}nk} \cdot \sqrt{\sigma_1(\mathbf{R}_1)^2+1} \cdot r^2$. By inspecting the remaining term and taking into account the fact that $\bar{m} = O(nk)$ and $\sigma_1(\mathbf{R}_1) \leq O(\sqrt{nk}) \cdot r$, finally we have

$$\|\widetilde{\mathbf{e}}_0\mathbf{R}_2' + \widetilde{\mathbf{e}}_2\| < \alpha q \cdot O(nk)^3 \cdot r^3.$$

By taking $1/\alpha = O(nk)^3 \cdot r^3$ we have the desired property for both error terms, $\widetilde{\mathbf{e}}_0\mathbf{R}_1' + \widetilde{\mathbf{e}}_1$, $\widetilde{\mathbf{e}}_0\mathbf{R}_2' + \widetilde{\mathbf{e}}_2 \in \mathcal{P}_{1/2}(q \cdot \mathbf{B}^{-t})$.

The security proof for our PRE scheme is essentially an adapted version of [15] [Theorem 6.3] to the proxy re-encryption model with a generalized **G**-trapdoor. As discussed in section 4.2, in order to solve LWE for any matrix $\mathbf{A}$ it is necessary to know both a transformation $\mathbf{R}$ and at least one invertible $\mathbf{H}$ embedded into $\mathbf{A}$. So we construct a simulator in a way that as long as there is a nonzero matrix $\mathbf{H}$ in a ciphertext, we are able to transform it to a **G**-matrix and decrypt, but once $\mathbf{H}$ equals to the zero matrix, no **R**-transformation can be applied to a ciphertext to reduce it to a **G**-matrix and recover a message. So when no invertible $\mathbf{H}$ is involved, we embed our LWE samples into a ciphertext and hence, decryption of the challenge helps us in deciding LWE.

Theorem 16 *The above scheme is PRE-CCA1-secure assuming the hardness of decision-*$\mathsf{LWE}_{q,\alpha'}$ *for* $\alpha' = \alpha/3 \geq 2\sqrt{n}/q$.

Proof. First, by [[14], Theorem 3.1] we transform the samples from LWE distribution $A_{\mathbf{s},\alpha'}$ of the form $(\mathbf{a}, b = \langle \mathbf{s}, \mathbf{a}\rangle/q + e \mod 1) \in \mathbb{Z}_q^n \times \mathbb{T}$ to the form $(\mathbf{a}, 2(\langle \mathbf{s}, \mathbf{a}\rangle \mod q) + e' \mod 2q)$ with $e' \to D_{\mathbb{Z},\alpha q}$ via mapping $b \mapsto 2qb + D_{\mathbb{Z}-2qb,s}$, where $s^2 = (\alpha q)^2 - (2\alpha' q)^2 \geq 4n \geq \eta_\epsilon(\mathbb{Z})^2$. The transformation maps the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{T}$ to the discretized uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_{2q}$.

Once the LWE samples are of the desired form, we construct column-wise a matrix $\mathbf{A}_0^*$ out of these samples and a vector $\mathbf{b}^*$ out of the corresponding components b's. A target's user public key is generated as follows: choose two invertible matrices $\mathbf{H}_1^*, \mathbf{H}_2^* \in \mathbb{Z}_q^{n\times n}$, the secret $\mathbf{R}_1^*, \mathbf{R}_2^* \leftarrow \mathcal{D}$ and output $\mathsf{pk}^* = ([\mathbf{A}_0^*| - \mathbf{A}_0^*\mathbf{R}_1^* - \mathbf{H}_1^*\mathbf{G}| - \mathbf{A}_0^*\mathbf{R}_2^* - \mathbf{H}_2^*\mathbf{G}],\ \mathbf{H}_1^*)$. Since the target user belongs to the set of honest users (we do not reveal his secret key), the matrix $\mathbf{H}_2^*$ remains statistically hidden from the adversary.

To generate the public key of an honest user we choose two matrices $\mathbf{X}_{00} \in \mathbb{Z}_q^{\bar{m}\times\bar{m}}$, $\mathbf{X}_{01} \in \mathbb{Z}_q^{nk\times\bar{m}}$ from a Gaussian distribution with parameter s and set

$$\mathbf{A}_0' = \begin{bmatrix}\mathbf{A}_0^*| - \mathbf{A}^*\mathbf{R}_1^*\end{bmatrix}\begin{bmatrix}\mathbf{X}_{00}\\ \mathbf{X}_{10}\end{bmatrix}.$$

Next we choose $\mathbf{R}_1', \mathbf{R}_2' \in \mathbb{Z}_q^{\bar{m}\times nk}$ from a distribution $\mathcal{B}$ defined over $\mathbb{Z}$ that outputs 0 with probability 1/2 and ± 1 with probability 1/4 each. We calculate the rest of the public key as

$$\mathbf{A}_0'\mathbf{R}_1' = \begin{bmatrix}\mathbf{A}_0^*| - \mathbf{A}^*\mathbf{R}_1^*\end{bmatrix}\begin{bmatrix}\mathbf{X}_{00}\\ \mathbf{X}_{10}\end{bmatrix} \cdot \mathbf{R}_1', \quad \mathbf{A}_0'\mathbf{R}_2' = \begin{bmatrix}\mathbf{A}_0^*| - \mathbf{A}^*\mathbf{R}_1^*\end{bmatrix}\begin{bmatrix}\mathbf{X}_{00}\\ \mathbf{X}_{10}\end{bmatrix} \cdot \mathbf{R}_2'.$$

So the whole public key of a honest user is $\mathsf{pk}' = ([\mathbf{A}_0'| - \mathbf{A}_0'\mathbf{R}_1'| - \mathbf{A}_0'\mathbf{R}_2' - \mathbf{H}_2^*\mathbf{G}],\ \mathbf{H}')$ for some randomly chosen invertible $\mathbf{H}' \in \mathbb{Z}_q^{n\times n}$. We add $-\mathbf{H}_2^*\mathbf{G}$ to each honest key. If we choose $\bar{m} \geq n \lg q + 2\frac{nk}{\delta}$ for a small δ, then by ([2]), $\mathbf{A}_0'\mathbf{R}_1'$ is $\mathsf{negl}(n)$-far from uniform, then again $-\mathbf{H}_2^*$ is hidden from the adversary. We denote $\begin{bmatrix}\mathbf{X}_{01}\\ \mathbf{X}_{11}\end{bmatrix} = \begin{bmatrix}\mathbf{X}_{00}\\ \mathbf{X}_{10}\end{bmatrix}\cdot \mathbf{R}_1'$ and $\begin{bmatrix}\mathbf{X}_{02}\\ \mathbf{X}_{12}\end{bmatrix} = \begin{bmatrix}\mathbf{X}_{00}\\ \mathbf{X}_{10}\end{bmatrix}\cdot \mathbf{R}_2'$. Each entry of the resulting matrices $\mathbf{X}_{01}, \mathbf{X}_{11}, \mathbf{X}_{02}, \mathbf{X}_{12}$ is the inner product of a Gaussian $\bar{m}$-dimensional row-vector (of either $\mathbf{X}_{00}$ or $\mathbf{X}_{10}$) and a $\{0,-1,1\}$-vector with half of the coordinates equal zero, which is equivalent to $\bar{m}/2$ additions of Gaussians with parameter s. Since in the scheme we sample $\begin{bmatrix}\mathbf{X}_{01}\\ \mathbf{X}_{11}\end{bmatrix}, \begin{bmatrix}\mathbf{X}_{02}\\ \mathbf{X}_{12}\end{bmatrix}$ with parameter $s\sqrt{\bar{m}/2}$, the simulated re-encryption key

$$\mathsf{rk}_{\mathsf{pk}^*\to\mathsf{pk}'} = \begin{bmatrix}\mathbf{X}_{00} & \mathbf{X}_{01} & \mathbf{X}_{02}\\ \mathbf{X}_{10} & \mathbf{X}_{11} & \mathbf{X}_{12}\\ \mathbf{0} & \mathbf{0} & \mathbf{I}\end{bmatrix} \tag{4}$$

has the same distribution as a re-encryption key in the scheme. We generate the public keys and secret keys for corrupted users in the same way as in the scheme.

To generate a re-encryption key $\mathsf{rk}_{\mathsf{pk}'\to\mathsf{pk}''}$ for any two $\mathsf{pk}' \neq \mathsf{pk}^*, \mathsf{pk}''$ with invertible matrices $\mathbf{H}', \mathbf{H}''$ as corresponding second components, where either both public keys are corrupted or honest, we sample with $\mathbf{H}'$ a matrix $\begin{bmatrix}\mathbf{X}_{00}'\\ \mathbf{X}_{10}'\end{bmatrix}$ for a fixed parameter s and $\begin{bmatrix}\mathbf{X}_{01}'\\ \mathbf{X}_{11}'\end{bmatrix}, \begin{bmatrix}\mathbf{X}_{02}'\\ \mathbf{X}_{12}'\end{bmatrix}$ with fixed $s\sqrt{\bar{m}/2}$ as in the scheme. Note that by fixing the output standard deviation we achieve the same distribution of the re-encryption keys between two honest and two corrupted users, while the trapdoor matrices in these two cases have different parameters: r for a Gaussian $\mathbf{R}$ in the corrupted case, and $\sqrt{2\pi}$ for $\mathcal{B}$-distributed $\mathbf{R}$ of an honest user.

To answer the decryption queries for a ciphertext $c = (\mathbf{b}^t, \mathbf{H}_u)$, where

$$\mathbf{b}^t = \mathbf{s}^t[\mathbf{A}_0'| - \mathbf{A}_0'\mathbf{R}_1' + \mathbf{H}'\mathbf{G}| - \mathbf{A}_0'\mathbf{R}_2' - (\mathbf{H}_2^* - \mathbf{H}_u)\mathbf{G}] + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{enc}(\mathbf{m}))^t$$

under the honest public key $\mathsf{pk}' = [\mathbf{A}_0'| - \mathbf{A}_0'\mathbf{R}_1'| - \mathbf{A}_0'\mathbf{R}_2' - \mathbf{H}_2^*\mathbf{G}],\ \mathbf{H}')$ we first check that $\mathbf{H}_u$ is invertible. Then we use the fact that $\mathbf{H}_2^* - \mathbf{H}_u \in \mathbb{Z}_q^{n\times n}$ is an invertible matrix. So for the second step of our decryption algorithm we call $\mathsf{Invert}^{\mathcal{O}}$ on inputs $([\mathbf{R}_1'|\mathbf{R}_2'], \mathbf{A}_u = [\mathbf{A}_0'| - \mathbf{A}_0'\mathbf{R}_1' + \mathbf{H}'\mathbf{G}| - \mathbf{A}_0'\mathbf{R}_2' - (\mathbf{H}_2^* - \mathbf{H}_u)\mathbf{G}], \mathbf{b}^t, (\mathbf{H}_2^* - \mathbf{H}_u))$ and receive $\mathbf{z} \in \mathbb{Z}_q^n$ and $\mathbf{e} \in \mathbb{Z}_q^m$ such that $\mathbf{b}^t = \mathbf{z}^t\mathbf{A}_u + \mathbf{e}^t \bmod q$. If the length of $\mathbf{e}$ is short enough (step 3) and for $\mathbf{v} = \mathbf{b} - \mathbf{e} = (\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2)$ it holds that $\mathbf{v}_0 \in \mathbb{Z}_q^{\bar{m}}$ and $\mathbf{v}_0 \in 2\Lambda(\mathbf{A}_0^t)$ (step 4), then $\mathbf{v}$ can be expressed as

$$\mathbf{v}^t = 2(\mathbf{s}^t\mathbf{A}_u \mod q) + (\mathbf{0}, \mathbf{0}, \mathsf{enc}(\mathbf{m})^t \mod 2q.$$

To proceed with the decryption we multiply

$$\mathbf{v}^t\begin{bmatrix}\mathbf{R}_1 & \mathbf{R}_2\\ \mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{bmatrix} = 2(\mathbf{s}^t[\mathbf{H}'\mathbf{G}|(\mathbf{H}_2^* - \mathbf{H}_u)\mathbf{G}] \mod q) + (\mathbf{0}, \mathsf{enc}(\mathbf{m})) \mod 2q.$$

Applying $\mathtt{enc}^{-1}$ to the last nk coordinates we are able to decrypt with the message $\mathbf{m}$. To answer the re-encryption query from $\mathsf{pk}' = [\mathbf{A}_0'| - \mathbf{A}_0'\mathbf{R}_1'| - \mathbf{A}_0'\mathbf{R}_2' - \mathbf{H}_2^*\mathbf{G}]$ with $\mathbf{H}' \in \mathbb{Z}_q^{n\times n}$ to $\mathsf{pk}'' = [\mathbf{A}_0''| - \mathbf{A}_0''\mathbf{R}_1''| - \mathbf{A}_0''\mathbf{R}_2'' - \mathbf{H}_2^*\mathbf{G}]$ with $\mathbf{H}'' \in \mathbb{Z}_q^{n\times n}$ we apply $\mathsf{rk}_{\mathsf{pk}'\to\mathsf{pk}''} = \begin{bmatrix} \mathbf{X}_{00}' & \mathbf{X}_{01}' & \mathbf{X}_{02}' \\ \mathbf{X}_{10}' & \mathbf{X}_{11}' & \mathbf{X}_{12}' \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}$ generated as in the original. The re-encryption transforms

$$\mathbf{b}^t = \mathbf{s}^t[\mathbf{A}_0'| - \mathbf{A}_0'\mathbf{R}_1' + \mathbf{H}'\mathbf{G}| - \mathbf{A}_0'\mathbf{R}_2' - (\mathbf{H}_2^* - \mathbf{H}_u)\mathbf{G}] + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{enc}(\mathbf{m}))^t$$

to $\mathbf{b}'^t = \mathbf{s}^t[\mathbf{A}_0''| - \mathbf{A}_0''\mathbf{R}_1'' + \mathbf{H}''\mathbf{G}| - \mathbf{A}_0''\mathbf{R}_2'' - (\mathbf{H}_2^* - \mathbf{H}_u)\mathbf{G}] + \widetilde{\mathbf{e}}^t + (\mathbf{0}, \mathbf{0}, \mathsf{enc}(\mathbf{m}))^t$, decryptable under $\mathsf{sk}'' = [\mathbf{R}_1''|\mathbf{R}_2'']$.

To answer the decryption query we proceed in the same way as for any honest user; note that in this case the ciphertext is of the form $\mathbf{b}^t =$

$$2(\mathbf{s}^t[\mathbf{A}_0^*| - \mathbf{A}_0^*\mathbf{R}_1^*| - \mathbf{A}_0^*\mathbf{R}_2^* - (\mathbf{H}_2^* - \mathbf{H}_u)\mathbf{G}] \bmod q) + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{enc}(\mathbf{m}))^t \bmod 2q.$$

To summarize, we can answer the decryption queries of a ciphertext $c = (\mathbf{b}^t, \mathbf{H}_u)$ for any honest user as long as $\mathbf{H}_u \neq \mathbf{H}_2^*$, which is the case with overwhelming probability. If $\mathbf{H}_u = \mathbf{H}_2^*$ we answer the decryption query with $\perp$.

Finally, for the challenge ciphertext that encrypts the message $\mathbf{m} \in \{0,1\}^{nk}$ under pk^*, we choose $\mathbf{H}_u = \mathbf{H}_2^*$, in which case the encryption is of the form

$$\mathbf{b}^t = 2(\mathbf{s}^t[\mathbf{A}_0^*| - \mathbf{A}_0^*\mathbf{R}_1^*| - \mathbf{A}_0^*\mathbf{R}_2^*] \mod q) + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{enc}(\mathbf{m}))^t \mod 2q$$

for some $\mathbf{s} \in \mathbb{Z}_q^n$ and small $\mathbf{e}$. But instead of calculating this vector $\mathbf{b}$, we take the vector $\mathbf{b}^*$ prepared at the beginning of the game. Notice that if the simulator receives the LWE distribution, then $\mathbf{b}^{*t} = 2(\mathbf{s}^t\mathbf{A}_0^* \mod q) + \widehat{\mathbf{e}}_0^t \mod q$, where $\mathbf{s} \leftarrow \mathbb{Z}_q^n, \widehat{\mathbf{e}_0} \leftarrow D_{\mathbb{Z},\alpha q}$. We set the first nk coordinates of $\mathbf{b}^t$ to $\mathbf{b}^{*t}$. We set the last $2nk$ coordinates of $\mathbf{b}$ to

$$\mathbf{b}_1^t = \mathbf{b}_0^t\mathbf{R}_1^* + \widehat{\mathbf{e}}_1^t \mod 2q \in \mathbb{Z}_{2q}^{nk}, \tag{5}$$

$$\mathbf{b}_2^t = \mathbf{b}_0^t\mathbf{R}_2^* + \widehat{\mathbf{e}}_2^t + \mathsf{enc}(\mathbf{m}) \mod 2q \in \mathbb{Z}_{2q}^{nk}, \tag{6}$$

where $\widehat{\mathbf{e}_1}, \widehat{\mathbf{e}_2} \leftarrow D^{nk}_{\alpha q\sqrt{m}\cdot r}$. Then the challenge ciphertext is $(\mathbf{b} = (\mathbf{b}_0^t, \mathbf{b}_1^t, \mathbf{b}_2^t), \mathbf{H}_2^*)$, which has the same distribution as any ciphertext in the scheme, since $\widehat{\mathbf{e}}_0^t\mathbf{R}_1 + \widehat{\mathbf{e}}_1^t$ – the resulting noise terms in $\mathbf{b}_1^t$ – is according to [[16] Corollary 3.10] within $\mathsf{negl}(n)$-distance from $D_{\mathbb{Z},s}$, where $s^2 = (\|\widehat{\mathbf{e}}_0\|^2 + \bar{m}(\alpha q)^2) \cdot r^2$ – the parameter for vectors $\mathbf{e}_1, \mathbf{e}_2$ in the scheme. The same applies for the noise terms in $\mathbf{b}_2^t$.

Note that $(\mathbf{A}_0^*, \mathbf{b}^*, \mathbf{A}_0^*\mathbf{R}_1^*, \mathbf{A}_0^*\mathbf{R}_2^*, -\mathbf{b}^*\mathbf{R}_1^*, -\mathbf{b}^*\mathbf{R}_2^*)$ is $\mathsf{negl}(n)$-uniform for $\mathbf{R}_1^* \leftarrow \mathcal{D}$, $\mathbf{R}_2^* \leftarrow \mathcal{D}$ by the leftover hash lemma. So the simulated challenge ciphertext has the same distribution as any encrypted message.

6 Conclusions

We presented a unidirectional proxy re-encryption scheme based on hard problems on lattices. It can be seen from the security proof that our generalized $\mathbf{G}$-trapdoor definition leads to a CCA-1 secure construction, but we cannot achieve

CCA-2 security. Another limitation of our construction is that its security is proved in the selective model only. We leave it as an open problem to construct a CCA-2 secure lattice-based construction in the adaptive setting.

Acknowledgements. I thank Alex May for careful reading and valuable comments, Chris Peikert for fruitful discussions and S. Aleshnikov for suggesting the topic.

References

1. M. Ajtai. Generating hard instances of lattice problems. In *Proceedings of STOC*, pages 99–108, 1996.
2. J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. *Theory of Computing Systems*, 48(3):535–553, Apr. 2011.
3. G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *NDSS*, pages 29–43, 2005.
4. G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *ACM TISSEC*, pages 29–43, 2006.
5. W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. In *Mathematische Annalen, Volume 296, Issue 1*, pages 625–635, 1993.
6. M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In *EUROCRYPT*, pages 127–144. Springer-Verlag, 1998.
7. R. Canetti and S. Hohenberger. Chosen-ciphertext secure proxy re-encryption. In *Proc. of ACM-CCS007*, pages 185–194. ACM Press, 2007.
8. T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84*, pages 10–18, 1985.
9. C. Gentry. *A fully homomorphic encryption scheme.* PhD thesis, Stanford University, 2009.
10. M. Green and G. Ateniese. Identity-based proxy re-encryption. In *Proceedings of the 5th international conference on Applied Cryptography and Network Security*, ACNS '07, pages 288–306, 2007.
11. R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis.* Cambridge University Press, 1994.
12. B. Libert and D. Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. In *PKC08, LNCS*, 2008.
13. D. Micciancio and O. Regev. Worst-case to average-case reductions based on gaussian measures. In *SIAM J. on Computing*, pages 372–381, 2004.
14. C. Peikert. An efficient and parallel gaussian sampler for lattices. In *CRYPTO*, pages 145–166, 2006.
15. C. Peikert and D. Micciancio. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718, 2011.
16. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93. ACM Press, 2005.
17. R. Vershynin. Introduction to the non-asymptotic analysis of random matrices, 2011. Available from: `http://www-personal.umich.edu/~romanv/papers/non-asymptotic-rmt-plain.pdf`.
18. K. Xagawa. *Cryptography with Lattices.* PhD thesis, Tokyo Institute of Technology, 2010. Available from: `http://xagawa.net/pdf/2010Thesis.pdf`.