A Generic Attack on Lattice-based Schemes using Decryption Errors

with Application to ss-ntru-pke

Qian Guo[1,2], Thomas Johansson[2], and Alexander Nilsson[2]

[1] Dept. of Informatics, University of Bergen, Box 7803, N-5020 Bergen, Norway
qian.guo@uib.no
[2] Dept. of Electrical and Information Technology, Lund University, P.O. Box 118, 221 00 Lund, Sweden
{thomas.johansson,alexander.nilsson}@eit.lth.se

Abstract. Hard learning problems are central topics in recent cryptographic research. Many cryptographic primitives relate their security to difficult problems in lattices, such as the shortest vector problem. Such schemes include the possibility of decryption errors with some very small probability. In this paper we propose and discuss a generic attack for secret key recovery based on generating decryption errors. In a standard PKC setting, the model first consists of a precomputation phase where special messages and their corresponding error vectors are generated. Secondly, the messages are submitted for decryption and some decryption errors are observed. Finally, a phase with a statistical analysis of the messages/errors causing the decryption errors reveals the secret key. The idea is that conditioned on certain secret keys, the decryption error probability is significantly higher than the average case used in the error probability estimation. The attack is demonstrated in detail on one NIST Post-Quantum Proposal, ss-ntru-pke, that is attacked with complexity below the claimed security level.

Keywords: Lattice-based cryptography, NIST post-quantum standardization, decryption error, LWE, NTRU, Reaction attack.

1 Introduction

Lattice-based cryptography and the learning with errors problem (LWE) [21] has recently developed into one of the main research areas in cryptography. We have considered factoring and the discrete logarithm problem to be fundamental during the last 50 years, but now we may see a shift towards other problems, due to the possibility of quantum computers. Lattice-based cryptography is now an enabler for a rich collection of cryptographic primitives, ranging from simpler ones like key exchange and (public-key) encryption to more advanced constructions like fully homomorphic encryption.

There are several factors in favor of using LWE or related problems as the underlying problem in cryptographic constructions. One is that constructions can

be computationally efficient compared to existing solutions. Another motivating factor is the hope that LWE-based constructions will be resistant towards a quantum computer. It is also potentially the problem that can best provide us with constructions of fully homomorphic encryption [7,5].

A very important problem is to establish the difficulty of solving various LWE-like problems, as it directly gives an upper bound on the security of the cryptographic primitives that base their security on them. There are reductions for LWE to worst-case lattice problems [21,20,6], but it may not always be applicable or it may not give useful help in choosing parameters. Today, the security of a primitive is often estimated from the computational complexity of lattice-basis reduction algorithms like BKZ and its different versions.

Several standardization initiatives for post-quantum primitives are currently running, where most focus is on the NIST post-quantum standardization project [1]. In the analysis of submitted proposals, the most important aspect is their security. Traditionally, the computational complexity for solving problems like LWE through lattice basis reduction is the guide for explicitly suggested parameter choices. Then most proposals are accompanied by some proof of security, relating to some well known and difficult problem in lattice theory, such as the shortest vector problem.

Most lattice-based schemes include the possibility of having decryption errors with some small probability. Making this probability very small has a price, as the parameters should be adjusted accordingly, resulting in a performance loss. A common approach used by some schemes is to use error-correcting codes to correct errors. In essence, a part of the "message" is parity-check bits that enable correction of a fixed number of errors. Such schemes can then have a much larger error probability in each bit position, as it requires that a number of them are in error for a decryption error to occur. Still, the possibility of having decryption errors can be used in cryptanalysis.

In our setting we are considering CCA (chosen-ciphertext attacks) security for PKE (public-key encryption) schemes. It equally well applies to KEMs (key encapsulation mechanisms) and key exchange in the public key setting, assuming that a large number of session keys are exchanged for the same fixed public key. The most basic form of security is the notion of CPA (chosen plaintext attacks) security, which is often not sufficient. A common tool for transforming a CPA-secured PKE into a scheme secured against CCA attacks is the Fujisaki-Okamoto transformation. But a CCA secured cryptosystem using such a transformation requires decryption with a negligible error probability because the attacker can exploit decryption errors in an attack. For schemes trying to provide resistance against attacks exploiting decryption errors, we typically see probabilities smaller than 2^{-128}. Sometimes the calculations of the probability for a decryption error use independence assumptions when there is actually a correlation between the coefficients in the error. The correctness of such approximations on the failure rate is an open and frequently discussed research question.

1.1 Related Works

The idea of exploiting decryption errors has been around for a long time and applies to all areas of cryptography [4]. For lattice-based encryption systems, the old Ajtai-Dwork and NTRU have been a target for attacks based on decryption failures. Hall, Goldberg, and Schneier developed already in 1999 [14] a reaction attack which recovers the Ajtai-Dwork private key by observing decryption failures. Hoffstein and Silverman [15] adapted the attack to NTRU. They also suggested to modify NTRU to use the Fujisaki-Okamoto transform [11] to protect against such attacks, but there were still problems [17]. Further work in this direction on NTRU is given in [12].

More recently, Fluhrer [10] showed how to attack key-exchange protocols in a key reuse setting. In [8] his work was extended to more protocols. In [3] a chosen-ciphertext attack on the proposal HILA5 [22] was given, using decryption failures. These attacks are CCA attacks on proposals with only CPA-security.

In this paper we will only consider CCA attacks on schemes proposed for CCA security. For this setting, Guo, Johansson and Stankovski [13], proposed a key-recovery attack against the CCA-secure version of QC-MDPC, which is a code-based scheme. It uses a distinguishing property that 'colliding pairs' in the noise and the secret can change the decryption failure rate.

1.2 Contributions

In this paper we propose and discuss a generic attack for secret key recovery based on generating decryption errors. It is a chosen-ciphertext attack and it targets schemes proposed for CCA security. The attack is described in a standard PKC setting, assuming that a number of messages are decrypted for a fixed pair of public and secret keys. It can however also be applied to for example KEM schemes if many keys are exchanged for a fixed pair of public and secret keys.

The model first consists of a precomputation phase where special messages and their corresponding error vectors are generated. Secondly, the messages are submitted for decryption and some decryption errors are observed. Finally, a phase with a statistical analysis of the messages/errors causing the decryption errors reveals the secret key.

The idea is that conditioned on certain secret keys, the decoding error probability is significantly higher than the average case used in typical error probability estimations. The attack is demonstrated in detail on one NIST Post-Quantum Proposal, the ss-ntru-pke version of NTRUEncrypt, that is attacked with complexity below the claimed security level.

A result of the attack is that it is questionable whether it is possible to achieve high security (say 256 bits in the classic sense) for proposals with mid-level decryption error probability (say 2^{-100}).

1.3 Organization

The remaining of the paper is organized as follows. We introduce some background in Section 2 and the general attack model in Section 3. We later describe

the proposed CCA attack on ss-ntru-pke in details in Section 4. This is followed by a section concluding the paper.

2 Background

In this section we want to provide the setting for the attack approach that we propose. It applies to PKC schemes that use ideas from lattice-based cryptography, which at least includes schemes making use of the LWE problem or related lattice problems, like NTRU. It makes use of the fact that decryption is in general not error-free, but can occasionally fail. We illustrate in more detail by giving a generic example of LWE-based schemes and then present some background on employing the Fujisaki-Okamoto transform to achieve CCA security.

Let q be an odd prime and $\mathbb{Z}_q = \{-(q-1)/2, .., (q-1)/2\}$. Let $\mathcal{R}$ denote the ring $\mathbb{Z}_q[x]/(x^N+1)$. In different contexts, we use a polynomial $u(x)$ or a vector $\mathbf{u}$ to denote an element in $\mathcal{R}$. We frequently mix between the notion of $\mathbf{u}$ being a vector and a polynomial, but the meaning should be clear from the context. For two vectors $\mathbf{a}, \mathbf{b}$, we denote $\mathbf{a} \cdot \mathbf{b}$ their inner product. The notion $\mathbf{a} * \mathbf{b}$ means multiplying the polynomials $\mathbf{a}$ and $\mathbf{b}$ in the ring $\mathcal{R}$.

2.1 LWE-Based Scheme

A basic encryption scheme based on the ring-LWE problem has been proposed by Lyubashevsky, Peikert and Regev in [19]. The scheme is described below.

- Key Generation: Generate a polynomial $\mathbf{a}$ with coefficients chosen uniformly in $\mathbb{Z}_q$. Next, randomly generate two polynomials $\mathbf{s}, \mathbf{e} \in \mathcal{R}$ with coefficients chosen from a special distribution $\mathcal{X}$ (often a discrete Gaussian distribution) and compute $\mathbf{b} = \mathbf{a} * \mathbf{s} + \mathbf{e} \in \mathcal{R}$. The public key is $(\mathbf{a}, \mathbf{b})$ and the private key is $(\mathbf{s}, \mathbf{e})$.
- Encryption: Let the message $\mathbf{m}$ have length N and binary coefficients ($m_i \in \{0, 1\}$). Map $\mathbf{m}$ to $\hat{\mathbf{m}}$ by computing $\hat{\mathbf{m}} = (q-1)/2 \cdot \mathbf{m}$. Randomly sample three polynomials $\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3}$ with coefficients in $\mathcal{X}$. The ciphertext is now the pair $(\mathbf{c_1}, \mathbf{c_2})$, where the polynomials are computed as $\mathbf{c_1} = \mathbf{a} * \mathbf{e_1} + \mathbf{e_2}$ and $\mathbf{c_2} = \mathbf{b} * \mathbf{e_1} + \mathbf{e_3} + \hat{\mathbf{m}}$.
- Decryption : Compute $\mathbf{m_0} = -\mathbf{c_1} * \mathbf{s} + \mathbf{c_2}$ and decode the coefficients of $\mathbf{m_0}$ to either 0 or 1, depending on whether the value is closest to 0 or $(q-1)/2$ in $\mathbb{Z}_q$.

Now computing $-\mathbf{c_1} * \mathbf{s} + \mathbf{c_2} = -(\mathbf{a} * \mathbf{e_1} + \mathbf{e_2}) * \mathbf{s} + \mathbf{b} * \mathbf{e_1} + \mathbf{e_3} + \hat{\mathbf{m}} = -\mathbf{e_2} * \mathbf{s} + \mathbf{e} * \mathbf{e_1} + \mathbf{e_3} + \hat{\mathbf{m}}$ Because $-\mathbf{e_2} * \mathbf{s} + \mathbf{e} * \mathbf{e_1} + \mathbf{e_3}$ only contains polynomials with coefficients from $\mathcal{X}$, which means very small coefficients, we can expect that also $-\mathbf{e_2} * \mathbf{s} + \mathbf{e} * \mathbf{e_1} + \mathbf{e_3}$ contains rather small coefficients. This means that the coefficients of $\mathbf{m_0}$ will be rather close to the coefficients of $\hat{\mathbf{m}}$ and this allows us to determine whether the original value was 0 or $(q-1)/2$.

However, with some small probability we can have an erronously decoded coefficient, leading to a decryption error. So if we encrypt messages and send

them for decryption, we can detect whether a decryption error occured, since then the decrypted message is not the same as the original one. To be more precise, we will have a decryption error if

$$||-\mathbf{e_2} * \mathbf{s} + \mathbf{e} * \mathbf{e_1} + \mathbf{e_3}||_\infty \geq q/4. \tag{1}$$

Hence, the probability of having a decryption error can be computed by computing the probability of the above event to happen.

Clearly, for the above scheme (providing CPA security only) an attacker Eve can construct a ciphertext by picking polynomials $\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3}$ of her own choice and in this way affect the decryption error probability to be potentially very large. But such a possibility is not available when schemes are using a transform to provide CCA security.

The scheme described in this subsection only provides CPA security and since CCA security is usually a demand, one has to provide additional mechanisms. The most common approach is to use the FO transformation. We do not provide details on how this is done, but we point at the relevant consequences for our attack. Namely, after transformations to provide CCA security, the randomness used in the encryption is no longer controllable in the encryption, e.g. one can no longer choose polynomials $\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3}$. Rather, it is the result of a hash function call, hashing the message and other parameters, that determines the values of these parameters picked from a given distribution. In our case this means that we cannot control the choice of randomness, but rather just pick a message and observe what sample values of $\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3}$ we will obtain.

3 Attack Models

In this section we present and discuss the proposed attack model. The major targets are NIST post-quantum proposals with CCA security using some CCA transformations, e.g., the Fujisaki-Okamoto (FO) one [11] and its variant [16]. We propose a general attack recovering secret keys from the observed decryption errors. The assumptions are the following:

- We are considering schemes of lattice-based types, where in addition there is a small probability of error in the decryption process. This is in accordance with typical lattice-based schemes, which in their encryption add vectors with small noise and and then remove the noise in the decryption.
- We consider chosen-ciphertext attacks, i.e., we may submit a number of ciphertexts and observe the result of decryption. If the ciphertext is an encrypted message, the decryption should return the message. But occasionally there can be a decryption error and then we assume that we observe the output provided by the scheme in case of a decryption error.
- We consider the existence of a number of public keys that can be attacked. For example, this may be a network of users each holding a public key.

We set the maximun number of ciphertexts that can be submitted to a node with a public key to be 2^K and we set the maximum number of public keys in the

system to be 2^L. Refering again to the NIST PQ-project, they have indicated in their call that at least $K = 64$ can be considered. In the discussion forum for the same project, we have also seen researchers mentioning that $L = 64$ can be considered. We will adopt $K = L = 64$ in this paper since it looks like values that are not questioned, although larger values can give more powerful attacks and could definitely be relevant. For example, comparing with attacks on symmetric schemes, such attacks may require receiving a number of plaintext-ciphertext pairs that are close to the number of keys (like 2^{200}), and still they are considered valid attacks.

The proposed attack procedure is split in three steps.

1. Do a precomputation step to establish pairs of messages and corresponding ciphertexts and let informally the set $\mathcal{S}$ denote error vectors corresponding to the different messages. These selected error vectors should be with particular properties, e.g, with large norm and/or with several large entries in certain positions, etc.
2. Send the ciphertexts connected to $\mathcal{S}$ and assume that we learn the decrypted messages. Assume further that a subset have been erroneously decrypted (wrong decoding due to too large error) and let $\mathcal{S}'$ be the error vectors causing decryption failure. This cardinality of this set could be larger than average if certain properties of the secret vector hold. So we submit the set of ciphertexts to each node holding a public key. The node giving the largest decryption failure rate is selected as the target public key for the attack.
3. Do statistical testing on the set $\mathcal{S}'$ (and possibly the set $\mathcal{S}$) to establish relationships between the secret key and given the noise vectors leading to a decryption failure. Analyzing their correlation, we may be able to to recover partial secrets, which can considerably reduce the solving complexity of the underlying hard problem. We are then able to perform a full key-recovery attack via classic approaches like using lattice reduction algorithms.

Generally, this attack model is called reaction attack as we only need to know whether the decryption was successful or not, we do not need to know the message that the decryption process gave as output.

We discuss the three steps briefly. In the precomputation step, we can observe a first difference between different schemes. Most schemes include the public key in the generation of the noisy vectors (as input in the hash function generating the noise). This means that a constructed set $\mathcal{S}$ can only be used for a single public key and then a new such set must be constructed for the next public key. For simplicity, we assume $|\mathcal{S}| = 2^K$. If we set the computational complexity of precomputing a set $\mathcal{S}$ to be 2^λ, the overall complexity of this first step is $2^{\lambda+L}$. On the other hand, there are also schemes where error vectors are generated independent of the public key (e.g. LAC). In such a case the same set $\mathcal{S}$ can be used on all public keys and the complexity is only 2^λ. We could also use Grover-type algorithms to accelerate the pre-computation step. For the sake of simplicity, however, we keep arguments in the classic regime throughout the paper.

For the second step, the idea is that among many public keys, there will be one where the corresponding secret values have a property that causes more decryption errors than on average. So to increase the decryption error probability to a reasonable and detectable level, we consider that a special property in the secret value is held with probability p', where $0 < p' < 1$. We then assume that $p' = 2^{-L}$, so we can expect that this special property in the secret value holds for one public key. As mentioned, with respect to the CCA security, NIST restricts to have at most 2^{64} decryption calls to each user (public key). So in order to distinguish a special property in the secret value corresponding to a public key, one needs to get the failure rate for this case to be larger than 2^{-64}.

Finally, in the statistical testing part, we have a set of error vectors that have caused decryption errors. There seems to be a plethora of methods that can be used to recover secret values. A general approach that we adopt is to consider a smaller part of the secret vector under reconstruction, and select the most probable values for this part, based on the observed error vectors in $\mathcal{S}$. Then one combines such guesses for different part and builds an approximation of a seeked secret vector. A good approximation will mostly be sufficient as it can be used in lattice-basis reduction algorithms.

4 Application to ss-ntru-pke

We have applied the described approach and provide the details of attacking ss-ntru-pke, a version in the NIST PQ submission – NTRUEncrypt [24]. Connected is also the provably secure NTRU [23] whose security is based purely on the hardness of Ring-LWE. NTRUEncrypt with different parameter choices has been around for a long time and is one of the most competitive lattice-based schemes when it comes to performance.

4.1 The Scheme

ss-ntru-pke is the version of NTRUEncrypt targeting the highest security level, being 256 bits. This scheme achieves CCA2 security via the NAEP transform [18], a transform similar to the Fujisaki-Okamoto one with an additional mask. We give a very brief explanation of the scheme. For most of the description and details, we refer to [24]. In the key generation (see Algorithm 1), two secret polynomials $\mathbf{f}, \mathbf{g} \in \mathcal{R}$ are selected, where the coordinates are chosen from a discrete Gaussian $\mathcal{X}_\sigma$ distribution with standard deviation σ. A public key is formed by computing $\mathbf{h} = \mathbf{g}/(p\mathbf{f} + 1)$.

We show in Algorithm 2 the encryption algorithm of ss-ntru-pke and in Algorithm 3 the decryption algorithm, both from the original proposal [24]. In these descriptions, HASH() represents a hash function, and $\mathcal{B}$ represents a set including all binary polynomials with degree at most $N-1$. The Pad() operation is a function to ensure the message has sufficient entropy, and the Extract() operation is the inverse of Pad().

Algorithm 1 ss-ntru-pke.KeyGen

Input: Parameter sets Param $= \{N, p, q, \sigma\}$ and a *seed*.
Output: Public key $\mathbf{h}$ and secret key $(\mathbf{f}, \mathbf{g})$.
1) Instantiate Sampler with $\mathcal{X}_\sigma^N$ and *seed*;
2) $\mathbf{f} \leftarrow$ Sampler, $\mathbf{g} \leftarrow$ Sampler;
3) $\mathbf{h} = \mathbf{g}/(p\mathbf{f} + 1) \mod q$;

In each encryption of a message $\mathbf{m}$, two polynomials $\mathbf{r}, \mathbf{e} \in \mathcal{R}$ are generated, where the coordinates are again chosen from a discrete Gaussian distribution $\mathcal{X}_\sigma$ with standard deviation σ. This randomness source uses a seed generated as Hash$(\mathbf{m}, \mathbf{h})$. This means that each choice of a message $\mathbf{m}$ will generate also the polynomials $\mathbf{r}, \mathbf{e} \in \mathcal{R}$. Let us denote this by

$$(\mathbf{r}, \mathbf{e}) = \mathbf{G}(\mathbf{m}, \mathbf{h}).$$

Algorithm 2 ss-ntru-pke.Encrypt

Input: Public key $\mathbf{h}$, message *msg* of length *mlen*, Param and a *seed*.
Output: Ciphertext $\mathbf{c}$.
1) $\mathbf{m}$ = Pad(*msg, seed*);
2) *rseed* = Hash$(\mathbf{m}|\mathbf{h})$;
3) Instantiate Sampler with $\mathcal{X}_\sigma^N$ and *rseed*;
4) $\mathbf{r} \leftarrow$ Sampler, $\mathbf{e} \leftarrow$ Sampler;
5) $\mathbf{t} = p \cdot \mathbf{r} * \mathbf{h}$;
6) *tseed* = Hash$(\mathbf{t})$;
7) Instantiate Sampler with $\mathcal{B}$ and *tseed*;
8) $\mathbf{m}_{mask} \leftarrow$ Sampler;
9) $\mathbf{m}' = \mathbf{m}$ - $\mathbf{m}_{mask} \pmod p$;
10) $\mathbf{c} = \mathbf{t} + p \cdot \mathbf{e} + \mathbf{m}'$;

In decryption, with ciphertext $\mathbf{c}$, one computes the message by computing

$$\mathbf{f} * \mathbf{c} = p \cdot \mathbf{r} * \mathbf{g} + p \cdot \mathbf{e} * \mathbf{f} + \mathbf{m}' * \mathbf{f}.$$

A decryption error occurs if $||p \cdot \mathbf{r} * \mathbf{g} + p \cdot \mathbf{e} * \mathbf{f} + \mathbf{m}' * \mathbf{f}||_\infty > q/2$. This basically translates to $||\mathbf{r} * \mathbf{g} + \mathbf{e} * \mathbf{f}||_\infty > q/4$ as $p = 2$ and the last term is much smaller than the first two.

The proposed parameters for ss-ntru-pke for the security level of NIST-V are shown in Table 1.

The decoding error probability is estimated to be less than 2^{-80} in [24].

N	q	$\lvert p\rvert$	$\mathcal{R}$	σ	ϵ	Security
1024	$2^{30}+2^{13}+1$	2	$\frac{\mathbb{Z}_q[x]}{x^N+1}$	724	$< 2^{-80}$	V

Table 1. Proposed ss-ntru-pke parameters.

Algorithm 3 ss-ntru-pke.DECRYPT

Input: Secret key $\mathbf{f}$, public key $\mathbf{h}$, ciphertext $\mathbf{c}$, and PARAM.
Output: *result*.
1) $\mathbf{m}' = \mathbf{f} * \mathbf{c} \pmod p$;
2) $\mathbf{t} = \mathbf{c} - \mathbf{m}'$;
3) $tseed = \text{HASH}(\mathbf{t})$;
4) Instantiate Sampler with $\mathcal{B}$ and *tseed*;
5) $\mathbf{m}_{mask} \leftarrow$ Sampler;
6) $\mathbf{m} = \mathbf{m}' + \mathbf{m}_{mask} \pmod p$;
7) $rseed = \text{HASH}(\mathbf{m}|\mathbf{h})$;
8) Instantiate Sampler with $\mathcal{X}_\sigma^N$ and *rseed*;
9) $\mathbf{r} \leftarrow$ Sampler;
10) $\mathbf{e} = p^{-1}\ (\mathbf{t} - \mathbf{r} * \mathbf{h})$;
11) **if** $||\mathbf{e}||_\infty$ *is big* **then**
  result $= \perp$;
else
  result $=$ Extract($\mathbf{m}$);

4.2 The Attack

We now follow the approach of the previous section and describe an attack. The detailed attack is shown in Algorithm 4, where a more efficient CCA2 version is adopted. We define an equivalence relation for two polynomials $u(x)$, $v(x) \in \mathcal{R}$ if $u(x) = x^i \cdot v(x) \pmod{x^N+1}$, or if $u(x) = -x^i \cdot v(x) \pmod{x^N+1}$, for $i \in \mathbb{Z}$.

Attack step 1 – pre-computation.

We pick random messages $\mathbf{m}$ and generate corresponding $(\mathbf{r}, \mathbf{e}) = \mathbf{G}(\mathbf{m}, \mathbf{h})$ for a given public key $\mathbf{h}$. We keep only vectors $\mathbf{e}$ equivalent to a polynomial that has the first l (e.g., $l = 2$) positions with the same sign and each with size larger than $c \cdot \sigma$, where c is a constant determining the computational effort of finding such error vectors. These vectors form our chosen set $\mathcal{S}$.

We set $l = 2$ to illustrate the idea in a concrete attack. For one position, the probability that the entry is larger than $c\sigma$ is $1 - \text{erf}(c/\sqrt{2})$. As we can start from any position, the probability to have two consecutive positions with the same sign and entries larger than $c\sigma$ is $p_\mathbf{e} = N * (1 - \text{erf}(c/\sqrt{2}))^2/2$. If we set $p_\mathbf{e}$ to be 2^{-120}, then c can be as large as 9.193.

Attack step 2 – submit ciphertexts for decryption.

Algorithm 4 The CCA2 attack against ss-ntru-pke

Input: A number (say 2^{64}) of public keys.
Output: The secret polynomials $(\mathbf{f}, \mathbf{g})$ of one public key.
1) Collect messages/ciphertexts with special form for all public keys;
2) Submit them for decryption and determine a weak public key $\mathbf{h}$;
1') Prepare messages/ciphertexts with special form for this weak key $\mathbf{h}$;
2') Submit them for decryption and collect the decryption results;
3) Use statistical analysis to have a guess $(\hat{\mathbf{f}}, \hat{\mathbf{g}})$ close to the corresponding secret key $(\mathbf{f}, \mathbf{g})$;
4) Use lattice reduction algorithms to recover the secret key $(\mathbf{f}, \mathbf{g})$;

We then send the ciphertexts corresponding to the noise vectors in $\mathcal{S}$ to the decryption algorithm. If the targeted secret key $\mathbf{f}$ is also equivalent to a polynomial that has the first l (e.g., $l = 2$) positions with the same sign and each with size larger than $c_s \cdot \sigma$, where c_s is another constant, then the decoding errors can be detectable. We expect to collect several errors and their store their corresponding error vectors $(\mathbf{r}, \mathbf{e})$. The probability to have two consecutive positions with the same sign and entries larger than $c_s\sigma$ is $p_{\mathbf{s}} = N * (1 - \text{erf}(c_{\mathbf{s}}/\sqrt{2}))^2/2$. If we set $p_{\mathbf{s}}$ to be 2^{-64}, then $c_{\mathbf{s}}$ can be as large as 6.802.

If we run 2^{120} precomputation steps for each stored vector with the desired properties, then the overall complexity is 2^{248} since $p_{\mathbf{s}} = 2^{-64}$. Let C_1 denote $2 \cdot c_{\mathbf{s}} c\sigma^2$. We can then have a coefficient in $\mathbf{r} * \mathbf{g} + \mathbf{e} * \mathbf{f}$ whose absolute contribution from these two big entries is at least $C_1 = 2^{25.97}$. We consider the probabilistic behavior of the remaining $(2N - 2)$ positions. As the coefficients of $\mathbf{r}, \mathbf{g}, \mathbf{e}, \mathbf{f}$ are all sampled from a Gaussian distribution with mean 0 and stand deviation $\sigma = 724$, the expected norm of the rest vector in $\mathbf{f}, \mathbf{g}$ with $2N-2$ entries is about $\sqrt{2N-2} \cdot \sigma$. Given a public key, $\mathbf{f}, \mathbf{g}$ is fixed. Thus, this coefficient of $\mathbf{r} * \mathbf{g} + \mathbf{e} * \mathbf{f}$ can be approximated as $C_1 + \Phi_0$, where Φ_0 is Gaussian distribution with mean 0 and standard deviation $\sqrt{2N-2} \cdot \sigma^2$. As the error appears when this coefficient is larger than $q/4$, the error probability[3] can be approximated as

$$P_e = \left(1 - \text{erf}(\frac{q/4 - C_1}{\sqrt{2(2N-2)}\sigma^2})\right) \cdot \frac{1}{2}.$$

We obtain a decoding error probability of $2^{-57.3}$ for this example. Thus we can obtain about $2^{6.7}$ errors from the 2^{64} decryption trails.

An adaptive CCA attack. If we keep the previous setting, i.e., a CCA1 attack, the cost is larger than 2^{248}. However, we can adopt a much more powerful attack model, namely an adaptive CCA (CCA2) attack, consisting of two

[3] The error can occur in both directions. We omit the term $\left(1 - \text{erf}(\frac{q/4+C_1}{\sqrt{2(2N-2)}\sigma^2})\right) \cdot \frac{1}{2}$ as it is negligible compared with $\left(1 - \text{erf}(\frac{q/4-C_1}{\sqrt{2(2N-2)}\sigma^2})\right) \cdot \frac{1}{2}$ for C_1 a very big positive integer.

phases. In the first phase, the attacker spend half of his computational power to determine a weak key; in the later phase, he would put all his remaining resources into attacking this weak key.

To be more specific, we first to prepare 2^{63} messages/ciphertexts for each of the 2^{64} public keys. Then we expect two errors corresponding to one key, which can be claimed as a weak key.

We can also reduce the precomputation work for each key to 2^{89}, if there are 2^{64} public keys. We have $c = 7.956$ and the error probability is $2^{-62.0}$, so we expect to have two errors in the testing stage. We then spend 2^{216} work on another precomputation to have 2^{63} messages with c to be 10.351, done only for this weak key. The error probability in the second phase is estimated as $2^{-53.0}$, so we can have 2^{10} errors. The overall complexity is 2^{217}.

Attack step 3 – statistical analysis.

In this step we will try to recover the secret $\mathbf{f}$. Let us first assume that $\mathbf{f}$ has its two big entries in the first two positions of the vector. Then the position in $\mathbf{e} * \mathbf{f}$ where the error occurs, denoted i_0, is the position when the two significant coefficients in $\mathbf{e}$ and those in $\mathbf{f}$ coincide. We now transform each $\mathbf{e}$ in such a way that its two big entries are also to be found in the first two positions. This is done by replacing $\mathbf{e}$ with the corresponding equivalent vector where the two big entries are in the first two positions. Assuming M decryption errors, this now gives us the following knowledge from the received decryption errors:

$$\sum_{i=2}^{N-1} e_i^{(j)} f_i + N_i^{(j)} > q/4 - 2 \cdot c_{\mathbf{s}} c \sigma^2,$$

for $j = 1..M$ and where $N^{(j)}$ denotes the remaining contribution to the noise. Finally we note that assuming that $\mathbf{f}$ has its two big entries in the first two positions is not a restriction, as such an $\mathbf{f}$ vector will just be an equivalent vector of the true $\mathbf{f}$. So we need only to recover $\mathbf{f}$ and then check all equivalent vectors.

We next show how to derive more knowledge of $\mathbf{f}, \mathbf{g}$ with statistical tools.

A heuristic approach. As we have assumed that the two big entries in $(\mathbf{f}, \mathbf{g})$ (or $(\mathbf{e}, \mathbf{r})$) are the first two entries, we use $\mathbf{K}$ (or $\mathbf{V}_i$ for $1 \leq i \leq M$) to denote a vector consisting of the remaining $2N - 2$ entries. Thus, the size of $\mathbf{K}$ (or $\mathbf{V_i}$) can be estimated as $\sqrt{(2N-2)}\sigma$.

We adopt the heuristic assumptions from [12] that all the errors are very close to the folding bound $q/4$, meaning that all the messages leading to an error belong to a hyperplane

$$\mathbf{V}_i \cdot \mathbf{K} = \frac{q}{4} - C_1,$$

where C_1 is the contribution from the two significant entries.

Thus, the mean vector $\hat{\mathbf{V}}$ of $\mathbf{V_i}$ should be close to a scaled vector of $\mathbf{K}$, i.e.,

$$\hat{\mathbf{V}} = \frac{\sum_{i=1}^{M} \mathbf{V}_i}{M} \approx \frac{q/4 - C_1}{\|\mathbf{K}\|^2} \mathbf{K}.$$

We can have an estimation $\hat{\mathbf{K}} = \frac{(2N-2)\sigma^2}{q/4-C_1}\hat{\mathbf{V}}$. If we round the entries of $\mathbf{K}$ to the nearest integer in $\mathbb{Z}_q$, we obtain an estimation $(\hat{\mathbf{f}}, \hat{\mathbf{g}})$ of the secret vector $(\mathbf{f}, \mathbf{g})$.

The remaining question is how good this estimation can be? We heuristically answer this question using the central limit theorem.

Each observation $\mathbf{V_i}$ with approximated norm $\sqrt{2N-2}\sigma$ can be viewed as the summation of the signal point

$$\frac{q/4-C_1}{\|\mathbf{K}\|^2}\mathbf{K},$$

and a noise vector with squared norm

$$(2N-2)\sigma^2 - \frac{(q/4-C_1)^2}{(2N-2)\sigma^2}.$$

By the central limit theorem, if we have M observations, then the squared norm (variance) of the noise can be reduced by a factor of M. Hence, the error norm should be

$$\sqrt{\frac{1}{M}\cdot\left((2N-2)\sigma^2 - \frac{(q/4-C_1)^2}{(2N-2)\sigma^2}\right)}.$$

As we consider $\hat{\mathbf{K}}$ instead of $\hat{\mathbf{V}}$, the true error norm should be resized as

$$\frac{(2N-2)\sigma^2}{q/4-C_1}\cdot\sqrt{\frac{1}{M}\cdot\left((2N-2)\sigma^2 - \frac{(q/4-C_1)^2}{(2N-2)\sigma^2}\right)}. \tag{2}$$

Using this formula, we can have a candidate with error norm $0.169\sqrt{2N-2}\sigma$, assuming that 1024 errors have been collected.

Attack step 4 – lattice reduction.

If $(\Delta\mathbf{f}, \Delta\mathbf{g}) = (\mathbf{f}, \mathbf{g}) - (\hat{\mathbf{f}}, \hat{\mathbf{g}})$ is small, we can recover it using lattice reduction algorithms efficiently. Thus, we obtain the correct value of $(\mathbf{f}, \mathbf{g})$.

If we have the error size to be only $0.169\sqrt{2N-2}\sigma$, as assumed in the previous step, using the LWE estimator from Albrecht et al. [2], it takes about 2^{181} time and 2^{128} memory if one uses sieving to implement the SVP oracle in BKZ. Though the authors of [24] discussed about memory constraint for applying sieving in lattice-based cryptanalysis, we believe it is reasonable to assume for 2^{128} memory when considering a scheme targeting the classic 256-bit security level. Another possibility is to implement the SVP oracle using tuple sieving, further reducing the memory complexity to 2^{117}. The time complexity then increases to 2^{223}, but still far from achieving the claimed 256-security level.

4.3 Experimental Results

We have implemented some of the parts of the attack to check performance against theory. We have chosen exactly the same parameters in ss-ntru-pke as

q	error rate -estimated-	-simulated-
$q = 2^{29}$	$2^{-9.05}$	$2^{-9.19}$
$q = 2^{29} + 2^{26}$	$2^{-12.64}$	$2^{-12.96}$
$q = 2^{29} + 2^{27}$	$2^{-16.91}$	$2^{-17.09}$
$q = 2^{29} + 2^{27} + 2^{26} + 2^{25}$	$2^{-24.62}$	$2^{-24.57}$

Table 2. The simulated error rates v.s. the estimated error rates.

well as in the attack, except for the q value, which in the experiment was set to the values shown in Table 2. The reason being that is we wanted to lower the decryption error rate so that simulation was possible.

We put two consecutive entries in $\mathbf{f}$ each of size $6.2 \cdot \sigma$ and we generated error vectors with two large positive entries each of size $9.2 \cdot \sigma$. For such choice, we first verified the decryption error probabilities, as seen in Table 2. These match the theoretical results well.

q	error norm $/(\sqrt{2N-2}\sigma)$ -estimated-	-simulated-
$q = 2^{29}$	0.487	0.472
$q = 2^{29} + 2^{26}$	0.391	0.360
$q = 2^{29} + 2^{27}$	0.326	0.302
$q = 2^{29} + 2^{27} + 2^{26} + 2^{25}$	0.261	0.250

Table 3. The simulated error norm v.s. the estimated error norm. ($M = 1024$)

M	error norm $/(\sqrt{2N-2}\sigma)$ -estimated-	-simulated-
$M = 256$	0.522	0.490
$M = 512$	0.369	0.348
$M = 1024$	0.261	0.250
$M = 1536$	0.213	0.212

Table 4. The simulated error norm v.s. the estimated error norm. ($q = 2^{29} + 2^{27} + 2^{26} + 2^{25}$)

For each choice of q we then collected up to $M = 2^{10}+2^9 = 1536$ error vectors and processsed them in a statistical analysis step, to get a good approximation of $(\mathbf{f}, \mathbf{g})$. As the heuristic approach described, we first created an approximation of $(\mathbf{f}, \mathbf{g})$, say denoted by $(\hat{\mathbf{f}}, \hat{\mathbf{g}})$, by simply computing $\hat{f}_i = E \cdot \frac{\sum_{j=0}^{M-1} e_i^{(j)}}{M}$ as the value in the ith position. Here E is a constant that makes the norm of the vector to be as the expected norm of $\mathbf{f}$. Clearly, this is a very simple way of exploring the dependence between f_i and e_i, but still it seems to be sufficient.

We have plotted the simulated error norms for various q and M in Figure 1. Furthermore, we show the comparison between the simulated error norms and the estimated error norms according to the central limit theorem, i.e., Equation (2), in Table 3 and Table 4.

In the prior table, M is fixed to 1024 and q varies, while in the latter table, q is fixed to $2^{29} + 2^{27} + 2^{26} + 2^{25}$ and M varies. We see that in all the cases, the simulated data match the estimated data well, though the simulation seems always slghtly better than the estimation, i.e., with smaller error norms. Another observation from Table 4 is that the estimation using the central limit theorem becomes more accurate when M becomes larger, which is also very reasonable.

4.4 Summarizing the Attack

The best attack is a CCA2 type attack where we in precomputation work use $2^{89+63} = 2^{152}$ operations to derive 2^{63} special ciphertexts that are submitted for decryption. With probability 2^{-64} the secret $\mathbf{f}$ has the desired property of two consecutive big entries. If so, we will most likely see several decoding errors and such a weak key has been detected. When the weak key has been detected, we perform yet another precomputation that uses 2^{216} operations to derive 2^{63} additional special ciphertexts again submitted for decryption. We receive in expectation 1024 decryption errors and the knowledge from the error vectors will allow us to reconstruct $\mathbf{f}$ without too much trouble using lattice reduction algorithms, as experimental results strongly indicated. The overall complexity is thus approximately 2^{217} if the SVP oracle in BKZ is implemented via lattice sieving. Actually, the cost of the lattice reduction algorithms in the final stage is not the bottleneck, as we have many other powerful statistical tools in Step 3, e.g., the Maximum Likelihood Test approach, to make this cost negligible.

5 Conclusions and Future Works

We have proposed a generic reaction attack model against lattice-based schemes and applied this model to attacking ss-ntru-pke, a version in the NTRUEncrypt submission to the NIST post-quantum project. Specifically, we have presented an adaptive CCA attack on the claimed 256-bit (classic) security level of ss-ntru-pke. This attacking idea can be treated as extension of reaction attacks [13,9] that already jeopardize the CCA security of MDPC and LDPC based crypto-systems.

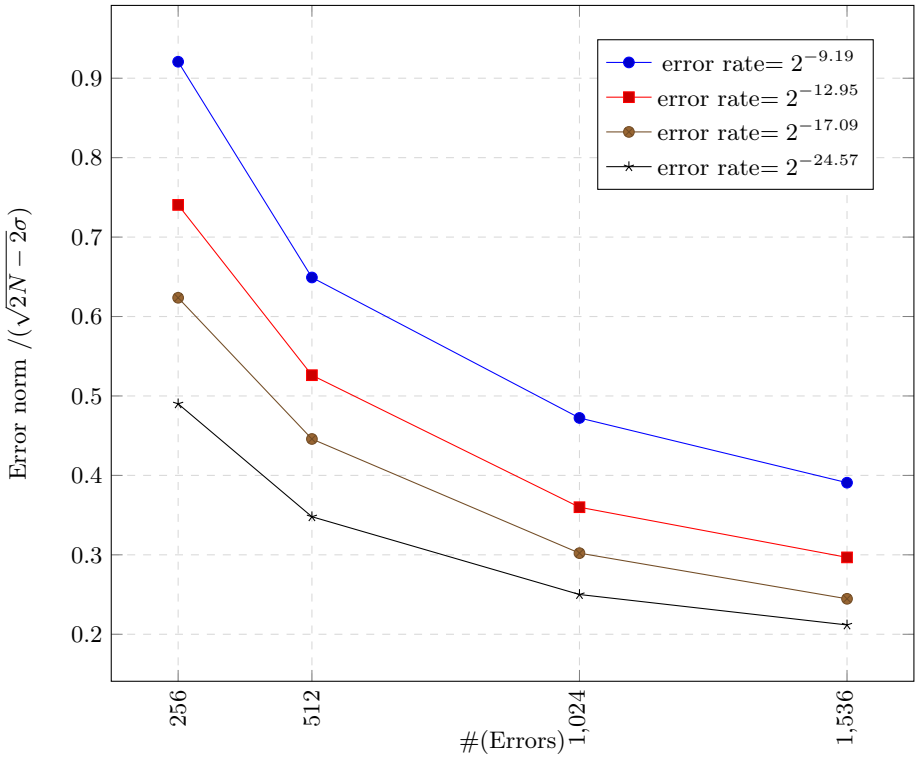

Fig. 1. Error norm as a function of the number of collected error vectors.

6 Acknowledgements

This work was supported in part by the Norwegian Research Council (Grant No. 247742/070), by the Swedish Research Council (Grant No. 2015-04528), by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, and by the Swedish Foundation for Strategic Research (SSF) project RIT17-0005.

References

1. NIST Post-Quantum Cryptography Standardization. `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization`, accessed: 2018-09-24
2. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Cryptology ePrint Archive, Report 2015/046 (2015), `http://eprint.iacr.org/2015/046`
3. Bernstein, D.J., Bruinderink, L.G., Lange, T., Panny, L.: HILA5 Pindakaas: On the CCA security of lattice-based encryption with error correction. In: Joux, A.,

Nitaj, A., Rachidi, T. (eds.) AFRICACRYPT 18: 10th International Conference on Cryptology in Africa. Lecture Notes in Computer Science, vol. 10831, pp. 203–216. Springer, Heidelberg, Germany, Marrakesh, Morocco (May 7–9, 2018)
4. Boldyreva, A., Degabriele, J.P., Paterson, K.G., Stam, M.: On symmetric encryption with distinguishable decryption failures. In: Moriai, S. (ed.) Fast Software Encryption – FSE 2013. Lecture Notes in Computer Science, vol. 8424, pp. 367–390. Springer, Heidelberg, Germany, Singapore (Mar 11–13, 2014)
5. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) ITCS 2012: 3rd Innovations in Theoretical Computer Science. pp. 309–325. Association for Computing Machinery, Cambridge, MA, USA (Jan 8–10, 2012)
6. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th Annual ACM Symposium on Theory of Computing. pp. 575–584. ACM Press, Palo Alto, CA, USA (Jun 1–4, 2013)
7. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: Ostrovsky, R. (ed.) 52nd Annual Symposium on Foundations of Computer Science. pp. 97–106. IEEE Computer Society Press, Palm Springs, CA, USA (Oct 22–25, 2011)
8. Ding, J., Alsayigh, S., RV, S., Fluhrer, S., Lin, X.: Leakage of signal function with reused keys in RLWE key exchange. Cryptology ePrint Archive, Report 2016/1176 (2016), http://eprint.iacr.org/2016/1176
9. Fabsic, T., Hromada, V., Stankovski, P., Zajac, P., Guo, Q., Johansson, T.: A reaction attack on the QC-LDPC McEliece cryptosystem. Cryptology ePrint Archive, Report 2017/494 (2017), http://eprint.iacr.org/2017/494
10. Fluhrer, S.: Cryptanalysis of ring-LWE based key exchange with key share reuse. Cryptology ePrint Archive, Report 2016/085 (2016), http://eprint.iacr.org/2016/085
11. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) Advances in Cryptology – CRYPTO'99. Lecture Notes in Computer Science, vol. 1666, pp. 537–554. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–19, 1999)
12. Gama, N., Nguyen, P.Q.: New chosen-ciphertext attacks on NTRU. In: Okamoto, T., Wang, X. (eds.) PKC 2007: 10th International Conference on Theory and Practice of Public Key Cryptography. Lecture Notes in Computer Science, vol. 4450, pp. 89–106. Springer, Heidelberg, Germany, Beijing, China (Apr 16–20, 2007)
13. Guo, Q., Johansson, T., Stankovski, P.: A key recovery attack on MDPC with CCA security using decoding errors. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology – ASIACRYPT 2016, Part I. Lecture Notes in Computer Science, vol. 10031, pp. 789–815. Springer, Heidelberg, Germany, Hanoi, Vietnam (Dec 4–8, 2016)
14. Hall, C., Goldberg, I., Schneier, B.: Reaction attacks against several public-key cryptosystems. In: Varadharajan, V., Mu, Y. (eds.) ICICS 99: 2nd International Conference on Information and Communication Security. Lecture Notes in Computer Science, vol. 1726, pp. 2–12. Springer, Heidelberg, Germany, Sydney, Australia (Nov 9–11, 1999)
15. Hoffstein, J., Silverman, J.H.: NTRU Cryptosystems Technical Report Report# 016, Version 1 Title: Protecting NTRU Against Chosen Ciphertext and Reaction Attacks

16. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017: 15th Theory of Cryptography Conference, Part I. Lecture Notes in Computer Science, vol. 10677, pp. 341–371. Springer, Heidelberg, Germany, Baltimore, MD, USA (Nov 12–15, 2017)
17. Howgrave-Graham, N., Nguyen, P.Q., Pointcheval, D., Proos, J., Silverman, J.H., Singer, A., Whyte, W.: The impact of decryption failures on the security of NTRU encryption. In: Boneh, D. (ed.) Advances in Cryptology – CRYPTO 2003. Lecture Notes in Computer Science, vol. 2729, pp. 226–246. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2003)
18. Howgrave-Graham, N., Silverman, J.H., Singer, A., Whyte, W.: NAEP: Provable security in the presence of decryption failures. Cryptology ePrint Archive, Report 2003/172 (2003), http://eprint.iacr.org/2003/172
19. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) Advances in Cryptology – EUROCRYPT 2010. Lecture Notes in Computer Science, vol. 6110, pp. 1–23. Springer, Heidelberg, Germany, French Riviera (May 30 – Jun 3, 2010)
20. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In: Mitzenmacher, M. (ed.) 41st Annual ACM Symposium on Theory of Computing. pp. 333–342. ACM Press, Bethesda, MD, USA (May 31 – Jun 2, 2009)
21. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) 37th Annual ACM Symposium on Theory of Computing. pp. 84–93. ACM Press, Baltimore, MA, USA (May 22–24, 2005)
22. Saarinen, M.J.O.: Hila5. Tech. rep., National Institute of Standards and Technology (2017), available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions
23. Stehlé, D., Steinfeld, R.: Making NTRU as secure as worst-case problems over ideal lattices. In: Paterson, K.G. (ed.) Advances in Cryptology – EUROCRYPT 2011. Lecture Notes in Computer Science, vol. 6632, pp. 27–47. Springer, Heidelberg, Germany, Tallinn, Estonia (May 15–19, 2011)
24. Zhang, Z., Chen, C., Hoffstein, J., Whyte, W.: Ntruencrypt. Tech. rep., National Institute of Standards and Technology (2017), available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions