Assessment of the Key-Reuse Resilience of NewHope

Aurélie Bauer[1], Henri Gilbert[1,2], Guénaël Renault[1,3], and Mélissa Rossi[4,5]

[1] ANSSI, Paris, France
[2] UVSQ, Versailles, France
[3] Sorbonne Université, CNRS, INRIA, Laboratoire d'Informatique de Paris 6, LIP6, Équipe PolSys, Paris, France
[4] Thales, Gennevilliers, France
[5] Département d'informatique de l'Ecole normale supérieure, CNRS, PSL Research University, INRIA, Paris, France
✉ melissa.rossi@ens.fr

Abstract. NewHope is a suite of two efficient Ring-Learning-With-Error based key encapsulation mechanisms (KEMs) that has been proposed to the NIST call for proposals for post-quantum standardization. In this paper, we study the security of NewHope when an active adversary accesses a key establishment and is given access to an oracle, called key mismatch oracle, which indicates whether her guess of the shared key value derived by the party targeted by the attack is correct or not. This attack model turns out to be relevant in key reuse situations since an attacker may then be able to access such an oracle repeatedly with the same key – either directly or using faults or side channels, depending on the considered instance of NewHope. Following this model we show that, by using NewHope recommended parameters, several thousands of queries are sufficient to recover the full private key with high probability. This result has been experimentally confirmed using Magma CAS implementation. While the presented key mismatch oracle attacks do not break any of the designers' security claims for the NewHope KEMs, they provide better insight into the resilience of these KEMs against key reuse. In the case of the CPA-KEM instance of NewHope, they confirm that key reuse (e.g. key caching at server side) should be strictly avoided, even for an extremely short duration. In the case of the CCA-KEM instance of NewHope, they allow to point out critical steps inside the CCA transform that should be carefully protected against faults or side channels in case of potential key reuse.

Keywords: PQ-crypto, lattice based cryptography, active attack, side channels

This research has been partially funded by ANRT under the program CIFRE 2016/1583. We acknowledge the support of the French Programme d'Investissement d'Avenir under national project RISQ P141580. This work is also partially supported by

1 Introduction

The insecurity of the main asymmetric cryptosystems (RSA, (EC)DLP) in front of a potential quantum computer has led the crytographic community to investigate new quantum resistant primitives. In 2016, NIST has initiated a process to develop and standardize one or more public-key cryptographic algorithms which are supposed to be quantum safe. Cryptosystems based on lattices represent one of the most promising directions for such systems.

Key Encapsulation Mechanisms (or KEMs) are one of the most important asymmetric cryptographic primitives. The NIST call specifically asks for quantum resistant KEM proposals in order to replace number theory based Diffie-Hellman key establishment protocols, which can be broken in the quantum computation model. Potential candidates for post quantum key establishment include the ones based on the lattice based Ring Learning With Errors Problem (Ring-LWE) introduced in [21,6]. Recently, Google conducted real life TLS experiments [5] with a Ring-LWE based key exhange scheme: the NewHope-Usenix system [5]. While these experiments show the efficiency of NewHope-Usenix, the specification of the reconciliation step of the system is rather complex. The technicality of this step requires a large fraction of the algorithm description in the original paper [1]. This and perhaps also intellectual property right considerations led the designers to introduce a simplified new variant initially named NewHope-Simple [24] where the *reconciliation-based* approach of NewHope-Usenix is replaced by an *encryption-based* approach. Thanks to the combined use of *encoding* and *compression* techniques, the performance price to pay for this new version in terms of bandwith overhead is quite marginal. Now NewHope-Simple has been transformed into NewHope, a suite of two candidate KEM mechanisms of the NIST call for proposals [22] named NewHope-CPA-KEM and NewHope-CCA-KEM, in short CPA-KEM and CCA-KEM. Both mechanisms are encryption-based: they rely upon an auxiliary probabilistic public key encryption allowing to encrypt a 256-bit data named CPA-PKE, that is not submitted to the NIST call for proposals as a standalone mechanism.

CPA-KEM, that is nearly identical to NewHope-Simple, is only claimed to be a passively secure KEM. It can be viewed as the CPA-PKE encryption of a hashed secret random value ν followed by hashing ν on both sides. Unlike CPA-KEM, CCA-KEM is claimed to be secure with respect to adaptively chosen ciphertext attacks. It is derived from CPA-PKE in a less straightforward manner, by applying a variant of the Fujisaki-Okamoto transform [10]. An essential feature of this transform is that the encryption of ν is derandomized: this allows the decrypting party Alice to re-encrypt the decryption result, check that the result matches the received ciphertext, and use this test to prevent information leakages on the private key in active attacks where "dishonestly" derived

the European Union PROMETHEUS project (Horizon 2020 Research and Innovation Program, grant 780701).

ciphertext values are sent by an adversary.

While the specification of CPA-KEM and CCA-KEM does not formally prevent re-using the same CPA-PKE (public key, private key) pair in multiple key establishments, the design rationale section of the NewHope specification requires that such key pairs be never cached and that a fresh key pair be generated at each key establishment[6]. In the case of CPA-KEM, one of the main reasons for this requirement is that, unlike the classical Diffie-Hellman key establishment, the original Ring-LWE based KEM with reconciliation is known to be vulnerable to a practical active attack in a key reuse situation as shown in [9]. Despite not being based on the reconciliation paradigm, CPA-KEM shares sufficiently many features with its predecessor for being conjectured also vulnerable to similar attacks. In the case of CCA-KEM, this requirement could be justified by the fact that no real perfect forward privacy can be offered if key caching is applied, i.e. private keys are not ephemeral[7].

Motivation

With its strong performance and its Ring-LWE based security, NewHope is a high profile candidate of the NIST competition. There is a good chance for it to be implemented in the future for Internet protocols. So, studying its security under several attacker models is important.

In this paper, we investigate the resilience of the CPA-KEM and CCA-KEM versions of NewHope in a misuse situation where the same key pair is reused for multiple key establishment by the private key owner – who will be referred to as Alice in the sequel. Note that Alice is also the party who initiates the two-round key establishment in both schemes. We use the generic name of key mismatch oracle to refer to the private key recovery attack models we are considering, that are closely inspired from the adversary model considered in [9]. While slightly less powerful than a CCA attack against an encryption based KEM where a decryption oracle is available, attacks using a key mismatch oracle still belong to the active attack category. Their common feature is that the adversary is assumed

[6]The single potential exception to this requirement is the *publicseed* part of the public key, whose caching "for say a few hours" seems to be considered by the designers as a viable alternative in situations where the preferred solution of a systematic renewal would turn out to be prohibitively expensive.

[7]On the other hand this requirement is not fully in line with the former observation, in the NewHope-Usenix paper, that "One could enable key caching with a transformation from the CPA-secure key establishment to a CCA-secure key establishment [...]". Given the performance advantage that may be provided by key caching at server side in certain applications, one can wonder whether it will be strictly followed in practice in all deployments of CCA-KEM if strong cryptanalytic arguments in favour of this conservative choice are not developed during the evaluation of the candidates to the NIST call.

to be able: (1) to actively interact with Alice by performing multiple KEM establishment where Alice uses the same key pair, (2) to produce each time a guess on the resulting secret key derived by Alice and (3) to access a binary oracle that indicates whether this guess is valid or not.

Our study is motivated by the belief that an in-depth understanding of the security offered by candidate KEM mechanisms submitted to the NIST call for proposals in key reuse situations is a useful part of their cryptanalytic evaluation, even for those candidates for which key reuse is considered as a misuse of the mechanism. Having an accurate estimate of the number of queries to the key mismatch oracle and of the complexity of the private key recovery really helps to assess the possible danger[8]. We focus here on a case study of the NewHope candidate KEMs. An advantage of this choice is that previous work on reconciliation-based Ring-LWE schemes such as [9] can be partly leveraged. However, as will be seen in the sequel, the fact that the NewHope suite is encryption-based and is using encoding techniques induces substantial differences and non-trivially complicates the cryptanalysis. To the best of our knowledge, no investigation of attacks against a scheme without reconciliation in a key mismatch oracle model was published so far.

Previous work

The danger of accessing a key mismatch oracle within some key agreement protocols in a key share reuse context has been already exposed several times. Early examples showing the vulnerability of some standardized Diffie-Hellman key agreement protocols in such a context were introduced in [19]. The potential danger of a somewhat related type of attack, namely so-called reaction attacks against PKE schemes [12], where an adversary can submit a chosen ciphertext to the legitimate private key owner and access a binary information about her reaction (whether the decryption succeeds or fails for instance), is probably even better known. Bleichenbacher's attack against RSA PKCS#1 of Crypto'98 [3] can been viewed as an early reaction attack. In 1999, Hall, Goldberg and Schneier presented reactions attacks against several PKE schemes [12]. In the particular case of lattice based cryptography, several notes on the vulnerability of NTRU to reaction attacks and its protection against such attacks were published [13,14]. In 2003, Howgrave-Graham *et al* proposed a reaction attack on NTRUEncrypt that leverages decryption failures [16]. A recent example of reaction attack is Guo *et al*'s key recovery attack on the code-based PKE QC-MDPC [11]. It is thus

[8]A similar need to investigate the resilience of candidate algorithms in misuse situations was encountered in the framework of the CAESAR competition aimed at selecting authenticated encryption primitives. In that competition, much analysis was conducted on the resistance of candidates to key recovery attacks in misuse cases such as nonce or decryption-misuse and this provided quite useful information for the algorithms selection process.

natural that NSA, in 2015, warns NIST Post-Quantum candidates against active attacks [18]. Few times later, the first concrete attacks on a Ring-LWE based key establishment leveraging a key mismatch oracle was proposed by Fluhrer [9] (see also [7,8]).

These attacks rely on the fact that the reconciliation step can be exploited by an active adversary to retrieve some information on the secret static key. Despite the warnings issued by the American agency, certain NIST candidates are vulnerable to active attacks. Indeed, it is shown in [2] that the secret key of the NIST candidate HILA5 can be recovered in the key mismatch oracle setting following Fluhrer's approach. In summary, despite the raising awareness of the cryptography research community that key mismatch oracle attacks threaten many lattice based KEMs in case of key reuse, relatively few examples of such attacks have been published so far.

About the side channel protection of NewHope, no dedicated countermeasure has been proposed for NewHope so far, but in [20] a side channel protection for a similar scheme has been proposed. This paper describes a provably first-order secure masking scheme and its integration into a CCA conversion.

Our contribution

In the following, we evaluate the security of NewHope when the attacker gets access to a key mismatch oracle. We concretely explain how the attacker can have access to such an oracle in different scenarios with the CPA-KEM and the CCA-KEM. We first introduce a straightforward way to recover such an oracle in the CPA-KEM. The adversary enters a key establishment with Alice, derives from her guess on the shared key produced by Alice a guess on the resulting session key she produces, and attempts to initiate a session with Alice under this guessed session key. The success or mismatch of this protected communication attempt provides the desired key mismatch oracle[9].

Then, at the end of the paper, we elaborate other scenarios on the CCA version which require side channels. Indeed, the CCA-KEM version induces major extra differences with formerly analyzed reconciliation-based schemes, that also deserve being analyzed. Because of the CCA transform, a key mismatch oracle cannot be accessed directly. But we show that for unsufficiently protected implementations, simple faults or side channels could bypass this transform and provide the desired key mismatch oracle. While unprotected versions of CCA-KEM are extremely efficient, its implementations must be very carefully protected against any key mismatch oracle leakage if key pairs are potentially reused. This might eventually come with a cost in terms of performance. This study may help the implementors to protect the algorithms against a possible key mismatch oracle leakage.

[9]It is worth noticing that the same direct access to a key mismatch oracle remains feasible if the KEM exchange is embedded in an authenticated key establishment protocol, under the sole condition that the adversary is the owner of a valid authentication (or signature) key.

The core of this work is the description of a new attack on NewHope using the key mismatch oracle. Even if the existence of previous work attacks (see [9] and [7]) casts suspicion on the resistance of NewHope CPA-KEM against active attacks in the same key-reuse setting, one has to take into account substantial differences between the reconciliation-based paradigm of the original NewHope and the encryption-based paradigm of CPA-KEM. Because of these differences, the detail of Fluhrer's attack [9] is not really inspiring for mounting an attack and any direct transposition attempt would be hopeless. Finding an efficient way of deriving information on the secret key from the key mismatch oracle with a low number of queries induces several issues. The main difficulty is to retrieve enough leakages after the application of the encoding and compression functions. had to investigate how to leverage these functions in order to find a simple way to instantiate the oracle. We then identified precise elements in the polynomial ring that can be used by the adversary to recover the secret. Finally, we had to take into account the fact that NewHope coefficients are in $[-8, 8]$.

We experimented our attack with a Magma CAS proof of concept. Under NewHope parameters, we were able to recover exactly the secret $\mathbf{S}$ with on average $16,700$ queries for NewHope1024 which corroborates the expected performances of the model.

Paper outline

In Section 2 we introduce some notation and describe the NewHope CPA scheme. In Section 3, we describe the notion of key mismatch oracle and how practical it can be for the CPA-KEM. In Section 4, we detail our attack using the key mismatch oracle. In Section 4.4, we present our experiments. In Section 5, we show how the key mismatch oracle can be retrieved with side channels with the CCA-KEM. Finally, in Section 6, we summarize our results and discuss future research.

2 Preliminaries

2.1 Notations

Let q be a prime number in $\mathbb{N}$ and let $\mathbb{Z}_q$ denote the ring elements $\mathbb{Z}/q\mathbb{Z}$. Depending on the context, the elements in $\mathbb{Z}_q$ can be equivalently represented as integers in $\{0, \dots, q-1\}$ or in $\{-(\frac{q-1}{2}), \dots, (\frac{q-1}{2})\}$. In the following, the notation $\mathcal{R}$ refers to the polynomial ring $\mathbb{Z}_q[x]/(x^N + 1)$ with N a power of 2. If $\mathbf{P}$ belongs to $\mathcal{R}$, it is a polynomial of degree $(N-1)$ with coefficients $\mathbf{P}[i]$ belonging to the set $\mathbb{Z}_q$. Such elements can also be represented as vectors whose i-th coordinate is the coefficient related to x^i. In the sequel we use either the polynomial notation or the vectorial one. For readability, bold capital letters are used to refer to elements in $\mathcal{R}$ and bold lowercase letters will refer to compressed elements, i.e. elements in $\mathcal{R}$ with small coefficients.

Let us define $\mathcal{G}_a$ as the centered Gaussian distribution of standard deviation a and ψ_k the centered binomial distribution of parameter k. Its standard deviation is $\sqrt{k/2}$. One may sample from ψ_k for integer $k > 0$ by computing $\sum_{i=1}^{k} b_i - b'_i$, where the b_i,$b'_i \in \{0, 1\}$ are uniform independent bits.

Property 1. The elements generated according to a centered binomial distribution ψ_k of parameter k are in the interval $[-k, k]$. Thus, the coefficients of the small elements drawn from $\mathcal{R}$ in NewHope are in $[-8, 8]$.

In the figures and algorithms, the notation $\overset{\$}{\leftarrow} \mathcal{D}$ means picking an element in $\mathcal{R}$ having all its coefficients generated at random according to distribution $\mathcal{D}$. The notation $\xleftarrow{coin} \mathcal{D}$ means using a $coin \in \{0, ..., 255\}^{32}$ as a seed to pick a pseudorandom element in $\mathcal{R}$ having all its coefficients according to distribution $\mathcal{D}$. This is generally done using a hash function like SHAKE-128. In the paper we refer several times to $Sign(a)$ with $a \in \mathbb{Z}$ by using the convention that it is defined as *positive* when $a \geq 0$ and as *negative* when $a < 0$. If $x \in \mathbb{R}$, the integer $\lfloor x \rceil$ is defined as $\lfloor x + \frac{1}{2} \rfloor \in \mathbb{Z}$.

2.2 NewHope

NewHope [24] [22] is a RING-LWE based key establishment scheme derived from NewHope-Usenix[1], that is simplified because it does not use the reconciliation anymore. In this section, we describe NewHope, where we omit some details (e.g. the so-called NTT transform or the encoding of the messages) to simplify the presentation. This does not imply any loss of generality for our attack. To ease the understanding, we will describe the CPA-KEM version of NewHope in this section as the key mismatch oracle can be easily derived. We will present the CCA-KEM later in Section 5 when we present some ways to access a key mismatch oracle.

The polynomial ring $\mathcal{R}$ used in NewHope has the following parameters: $(N, q) = (1024, 12289)$ or $(N, q) = (512, 12289)$. The coefficients of the small elements drawn from $\mathcal{R}$ follow a centered binomial distribution ψ_k^N with $k = 8$. The standard deviation is $a = \sqrt{\frac{8}{2}} = 2$. We decided to focus on explaining the attack for $N = 1024$. Indeed, for $N = 512$ there is twice less redundancy and the attack is easier. Thus, we fix $N = 1024$. These elements will be seen as vectors of size N with integer components. We denote $s = 1536$ which is such that $q = 8s + 1$. The aim of the system is to share a key of size 256 bits following the exchange mechanism outlined below and represented in Figure 1.

A public value $\mathbf{A} \in \mathcal{R}$ is derived from a published *seed*. Four specific functions are introduced: Encode, Decode, Compress and Decompress. They are described in Algorithms 1,2,3 and 4. Note that we partly deviate from the notation of the original specification of these algorithms, since we use the parameter s (the original description is in [24]). The following paragraphs describe these functions.

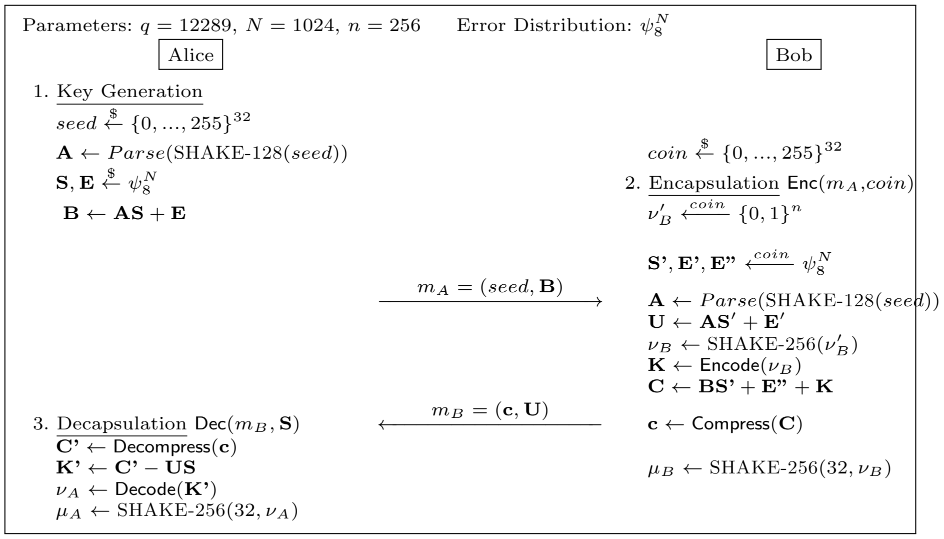

Parameters: $q = 12289$, $N = 1024$, $n = 256$ Error Distribution: ψ_8^N
Alice
Bob
1. Key Generation
$seed \xleftarrow{\$} \{0, ..., 255\}^{32}$
$\mathbf{A} \leftarrow Parse(\text{SHAKE-128}(seed))$
$\mathbf{S}, \mathbf{E} \xleftarrow{\$} \psi_8^N$
$\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$
$coin \xleftarrow{\$} \{0, ..., 255\}^{32}$
2. Encapsulation $\mathsf{Enc}(m_A, coin)$
$\nu'_B \xleftarrow{coin} \{0,1\}^n$
$\mathbf{S'}, \mathbf{E'}, \mathbf{E''} \xleftarrow{coin} \psi_8^N$
$m_A = (seed, \mathbf{B})$
$\mathbf{A} \leftarrow Parse(\text{SHAKE-128}(seed))$
$\mathbf{U} \leftarrow \mathbf{AS}' + \mathbf{E}'$
$\nu_B \leftarrow \text{SHAKE-256}(\nu'_B)$
$\mathbf{K} \leftarrow \mathsf{Encode}(\nu_B)$
$\mathbf{C} \leftarrow \mathbf{BS'} + \mathbf{E''} + \mathbf{K}$
$m_B = (\mathbf{c}, \mathbf{U})$
3. Decapsulation $\mathsf{Dec}(m_B, \mathbf{S})$
$\mathbf{c} \leftarrow \mathsf{Compress}(\mathbf{C})$
$\mathbf{C'} \leftarrow \mathsf{Decompress}(\mathbf{c})$
$\mathbf{K'} \leftarrow \mathbf{C'} - \mathbf{US}$
$\nu_A \leftarrow \mathsf{Decode}(\mathbf{K'})$
$\mu_B \leftarrow \text{SHAKE-256}(32, \nu_B)$
$\mu_A \leftarrow \text{SHAKE-256}(32, \nu_A)$

Fig. 1. Simplified NewHope

Compress and Decompress. The function Compress (Algorithm 3) takes as input a vector $\mathbf{C}$ in $\mathcal{R}$ and applies on each of its component a *modulus switching* to obtain an element $\mathbf{c}$ in $\mathbb{Z}_8[x]/(x^N + 1)$. Compressing a vector $\mathbf{C}$ essentially means keeping the 3 most significant bits of each coefficient. The function Decompress (Algorithm 4) shifts the bits of the input $\mathbf{c} \in [0, 8[^N$ to place them among the most significant bits. These functions are not the inverse of each other.

Encode and Decode. The Encode function takes a n-bit input ν where $n = N/4$ and creates an element $\mathbf{K} \in \mathcal{R}$ which stores 4 times the element ν. The redundancy is used by the function Decode to recover ν with a noisy $\mathbf{K}$.

NewHope Key Encapsulation Mechanism. Let us now describe this scheme.

1. *Setup*: Alice generates 2 small secrets $\mathbf{S}$ and $\mathbf{E}$ in $\mathcal{R}$. She sends $\mathbf{B} = \mathbf{AS} + \mathbf{E}$ to Bob.
2. *Key Encapsulation*: From a random *coin* acting as a seed, Bob derives 3 small secrets $\mathbf{S'}$, $\mathbf{E'}$ and $\mathbf{E''}$ in $\mathcal{R}$ and a random element ν_B of size n which will be

Algorithm 1: Key Encoding

```
function Encode($\nu \in \{0,1\}^n$)
k ← 0
for $i := 0$ to $n - 1$ do
    $\mathbf{K}_i \leftarrow \nu_i.4s$
    $\mathbf{K}_{i+n} \leftarrow \nu_i.4s$
    $\mathbf{K}_{i+2n} \leftarrow \nu_i.4s$
    $\mathbf{K}_{i+3n} \leftarrow \nu_i.4s$
end
Return k
```

Algorithm 2: Key Decoding

```
function Decode($\mathbf{K} \in \mathcal{R}$)
$\nu \leftarrow 0$
for $i := 0$ to $n - 1$ do
    $t \leftarrow \sum_{j=0}^{3} \left| \mathbf{K}_{i+jn} - 4s \right|$
    if $t < q$ then $\nu_i \leftarrow 1$ else $\nu_i \leftarrow 0$
end
Return $\nu$
```

Algorithm 3: Compression

```
1 function Compress($\mathbf{C} \in \mathcal{R}$)
2 for $i := 0$ to $N-1$ do
3     $\mathbf{c}[i] \leftarrow \left\lceil \frac{8.\mathbf{C}[i]}{q} \right\rfloor \mod 8$
4 end
5 Return $\mathbf{c}$
```

Algorithm 4: Decompression

```
1 function Decompress($\mathbf{c} \in [0, 8[^N$)
2 for $i := 0$ to $N-1$ do
3     $\mathbf{C'}[i] \leftarrow \left\lceil \frac{q.\mathbf{c}[i]}{8} \right\rfloor$
4 end
5 Return $\mathbf{C'}$
```

the encapsulated key. He computes $\mathbf{U} = \mathbf{AS'} + \mathbf{E'}$. He encodes ν_B into a redundant element $\mathbf{K}$ of $\mathcal{R}$ using the algorithm Encode (Algorithm 1). Bob uses Compress (Algorithm 3) to compress $\mathbf{C} = \mathbf{BS'} + \mathbf{E''} + \mathbf{K}$ into an element with very small coefficients as described above. He sends $(\mathbf{c} = \text{Compress}(\mathbf{C}), \mathbf{U})$ to Alice. He deduces the shared secret as $\mu_B = \text{SHAKE-256}(32, \nu_B)$.

3. *Key Decapsulation*: Alice decompresses $\mathbf{c}$ with Decompress into $\mathbf{C'}$(Algorithm 4). She computes $\mathbf{C'} - \mathbf{US}$ which is close to

$$\mathbf{C} - \mathbf{US} = \mathbf{ES'} + \mathbf{E''} + \mathbf{K} - \mathbf{E'S}. \tag{1}$$

Since $\mathbf{ES'} + \mathbf{E''} - \mathbf{E'S}$ is small, she recovers an estimated value ν_A of ν_B with a decoding algorithm called Decode(Algorithm 2). From ν_A, she can deduce $\mu_A = \text{SHAKE-256}(32, \nu_A)$.

Since $\mathbf{S}, \mathbf{S'}, \mathbf{E}, \mathbf{E'}, \mathbf{E''}$ are small, Alice and Bob get the same key $\mu = \mu_B = \mu_A$ with high probability.

Remark 1. This Section presented NewHope-CPA-KEM which is the target Section 4's analysis. However a PKE called NewHope-CPA-PKE has been introduced in [22]. The slim difference lies on the fact that ν_B becomes the encrypted message. The CCA security of the CCA version, called NewHope-CCA-KEM relies on the CPA security of NewHope-CPA-PKE (see Section 5).

3 The key mismatch oracle

This section introduces the notion of key mismatch oracle and a way to access it in the CPA version. We will always consider a malicious active adversary, Eve, who acts as Bob. Her messages, key and intermediate values will be denoted as m_E, μ_E and ν_E instead of m_B, μ_B and ν_B.

Remark 2. One might wonder how a malicious Alice can recover Bob's secret in a case of key reuse by Bob. In NewHope, this can be done with 2 queries, see Appendix A.

The goal of the adversary is to recover Alice's static private keys $\mathbf{S}$ and $\mathbf{E}$ by using the following oracle several times. We will focus on recovering the secret $\mathbf{S}$. $\mathbf{E}$ can be derived from $\mathbf{S}$ with $\mathbf{E} = \mathbf{B} - \mathbf{AS}$.

Definition 1. *(key mismatch oracle) A key mismatch oracle is an oracle that outputs a bit of information on the possible mismatch at the end of the key encapsulation mechanism.*
In the NewHope context, the key mismatch oracle is the oracle that takes any message m_E and any key hypothesis μ_E as input and outputs the following

$$\mathcal{O}_1(m_E, \mu_E) = \begin{cases} 1 & \text{if } (\mu_A =) \ \mathsf{Decapsulation}(m_E, \mathbf{S}) = \mu_E \\ -1 & \text{otherwise} \end{cases} \tag{2}$$

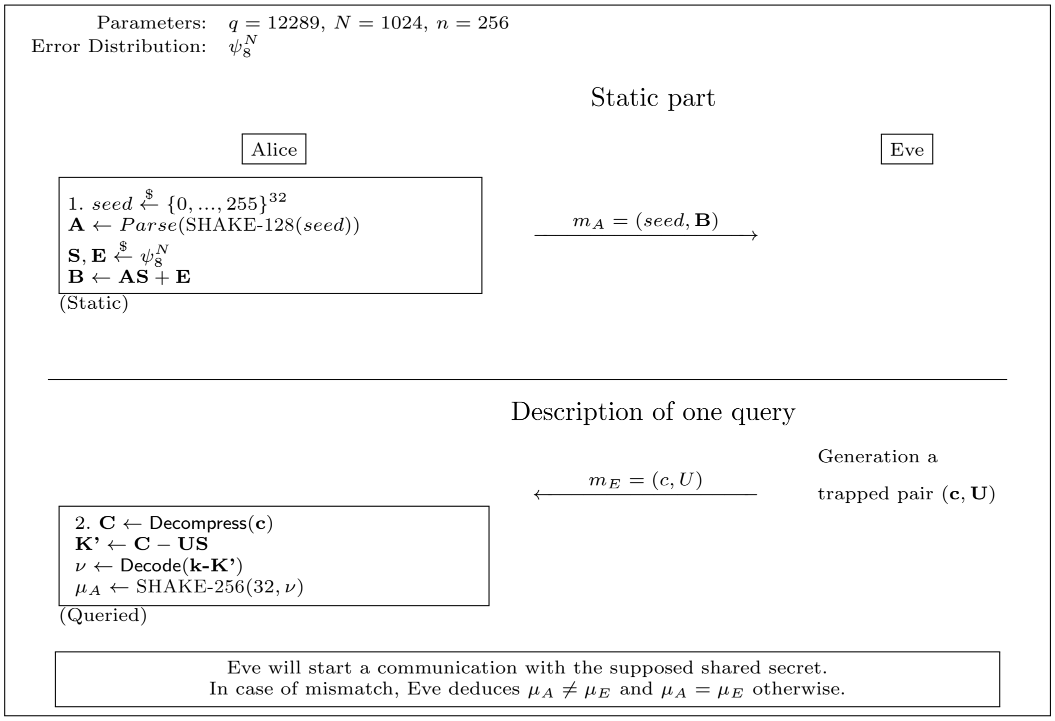
Such an oracle should leak information on secret **S** because its output is clearly correlated to the value of **S**. However, this oracle is less powerful than a CCA decryption oracle against CPA-PKE. Indeed, the only information given is a bit representing the possible key mismatch. The difficulty is to choose appropriate (m_E, μ_E) to retreive information of a small part of **S**. In Section 4, we present how to recover the secret **S** from such an oracle.

The simplest way to access such an oracle is when the CPA-KEM is implemented with static secrets. In other words, Alice will keep her secrets **S** and **E** for several key establishment requests. We consider that Eve does not necessarily follow the scheme specification. She can "cheat" and generate a message m_E that is not derived from a coin or from random small secrets **S'**, **E'** and **E"**. By definition, the CPA version of NewHope is passively secure, an attacker using a key mismatch oracle is outside of the security assumptions[10]. This has been well highlighted in paragraph 2.3, Section **No key caching** of the original paper of NewHope-Usenix. However an implementation of NewHope which allows misuse cases (see [23]) cannot be completely excluded. Thus it is important to precisely evaluate such a threat and consider the following attack model.

Attack Model 1 *Alice will accept any syntactically correct message m_E and always try to use the corresponding shared key for communicating. When she derives the shared key, either she is able to decrypt messages exchanged after that with Eve (and thus Eve deduces that the shared key is the same) or she will notify Eve that something went wrong with the key agreement. Eve will then deduce that the key is different. In both cases, Eve gets the desired key mismatch oracle.*

This model is summed up in Figure 2. In Section 5, we show how to get access to such an oracle with side channels in the CCA framework.

[10]While key reuse is against the designers' requirements of the NIST submission NewHope, as expressed in the footnote in the design rationale on p. 16, this requirement does not seem to be formally reflected in the algorithm description of section 1.2. This section indeed defines separate algorithms for key pairs generation, (en/de)capsulation, but does not state that a pair shall be used only once. Thus, though running NewHope with key reuse represents a misuse situation, analyzing the security of this scheme in this situation is definitely much more relevant question than considering variations in the formal specification of NewHope and investigating resulting weaknesses.



Parameters: $q = 12289$, $N = 1024$, $n = 256$
Error Distribution: ψ_8^N
Static part
Alice
Eve
1. $seed \xleftarrow{\$} \{0, ..., 255\}^{32}$
$\mathbf{A} \leftarrow Parse(\text{SHAKE-128}(seed))$
$\mathbf{S}, \mathbf{E} \xleftarrow{\$} \psi_8^N$
$\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$
(Static)
$m_A = (seed, \mathbf{B})$
Description of one query
Generation a
trapped pair $(\mathbf{c}, \mathbf{U})$
$m_E = (c, U)$
2. $\mathbf{C} \leftarrow \text{Decompress}(\mathbf{c})$
$\mathbf{K'} \leftarrow \mathbf{C} - \mathbf{US}$
$\nu \leftarrow \text{Decode}(\mathbf{k}\text{-}\mathbf{K'})$
$\mu_A \leftarrow \text{SHAKE-256}(32, \nu)$
(Queried)
Eve will start a communication with the supposed shared secret.
In case of mismatch, Eve deduces $\mu_A \neq \mu_E$ and $\mu_A = \mu_E$ otherwise.

Fig. 2. The Attack Model

4 Attack on NewHope with Key Mismatch Oracle

We assume here that Eve, the attacker, has access to $\mathcal{O}_1$, a key mismatch oracle as defined in Section 3. Let us now explain how she proceeds to recover Alice's static secret key $\mathbf{S}$ following Attack Model 1.

4.1 Rewriting the Key Mismatch Oracle

The use of the key mismatch oracle obviously leaks information on Alice's secret key $\mathbf{S}$. But the task of recovering $\mathbf{S}$ entirely seems much more complicated. Indeed as defined in Section 3, the only information provided by the key mismatch oracle is a bit representing the success or mismatch of the key agreement. The difficulty for Eve is to choose appropriate (m_E, μ_E) pairs to get useful information on small parts of $\mathbf{S}$.

In a first step, Eve simplifies her part of the protocol in such a way that the knowledge of the key mismatch oracle output bit $\mathcal{O}_1(m_E, \mu_E)$ can be easily exploited. To do so, she can fix for instance μ_E such that:

$$\nu_E = (1, 0, \ldots, 0) \text{ and thus } \mu_E = \text{SHAKE-256}(32, \nu_E). \quad (3)$$

The value of $\nu_E = (1, 0, \ldots, 0)$ has not been arbitrarily chosen ; as we will see later, the 0 in positions 1 to $n-1$ will help the success rate of the attack (see Proposition 3). From now on, the value of μ_E is *fixed* according to Equation (3). Moreover, when replacing m_E by its definition: $m_E = (\mathbf{c}, \mathbf{U})$, the oracle $\mathcal{O}_1$ can be reformulated using the oracle $\mathcal{O}_2$ defined below.

Definition 2. *Let us introduce oracle $\mathcal{O}_2$ such that $\mathcal{O}_2(\boldsymbol{c}, \boldsymbol{U}) = \mathcal{O}_1\big((\boldsymbol{c}, \boldsymbol{U}), \mu_E\big)$*

With this new definition, Eve can adapt the values of $\mathbf{c}$ and $\mathbf{U}$ to leverage Oracle $\mathcal{O}_2$ and retrieve information on $\mathbf{S}$. In other words, since μ_E is fixed, the inputs $(\mathbf{c}, \mathbf{U})$ are the degrees of liberty for finding $\mathbf{S}$.

From Alice's side, the link between ν_A and $\mathbf{S}$ passes through the functions Decode, Decompress (see Figure 2) and the element $\mathbf{K'}$: $\nu_A = \mathsf{Decode}(\mathbf{k'}) = \mathsf{Decode}(\mathbf{C} - \mathbf{US}) = \mathsf{Decode}(\mathsf{Decompress}(\mathbf{c}) - \mathbf{US})$. Thus, from the definition of the Decode algorithm, the value of $\nu_A[i]$, the i-th component of ν_A, is deduced from the following sign computation:

$$Sign\left(\sum_{j=0}^{3}\left|(\mathsf{Decompress}(\mathbf{c}) - \mathbf{U}\cdot\mathbf{S})[i+nj] - 4s\right| - q\right) \tag{4}$$

We recall here that 0 is positive by convention.

The problem for Eve is that she is unable to know the number of errors that will occur at the end of the decryption computations and the positions in which they appear. Indeed, the key mismatch oracle only gives one bit of information corresponding either to *mismatch* or *success*. If there is a mismatch, Eve knows that *at least* one bit of ν_A is different from ν_E but she can not determine which one (or which ones). Therefore, in order to mount an effective attack, Eve needs to restrict all these different possibilities by making the following hypothesis:

Hypothesis 1 *For i from 1 to $n-1$, the component $\nu_A[i]$ is equal to 0.*

If Hypothesis 1 is verified, any failure in the communication comes from a single error in ν_A located in the very first component $\nu_A[0]$. Indeed, in that case, the success of the exchange only depends on the first computed value $\nu_A[0]$. In particular, if we assume this hypothesis, the oracle $\mathcal{O}_2$ depends only on the $\nu_A[0]$ and we obtain the following result.

Lemma 1. *Under Hypothesis 1, the initial oracle $\mathcal{O}_2$ can be rewritten as*

$$\mathcal{O}_2(\boldsymbol{c}, \boldsymbol{U}) = Sign\left(\sum_{j=0}^{j=3}\left|(\mathsf{Decompress}(\boldsymbol{c}) - \boldsymbol{US})[0+nj] - 4s\right| - q\right)$$

For mounting her attack, Eve has to find pairs $(\mathbf{c}, \mathbf{U})$ that

1. target the smallest number of bits of $\mathbf{S}$
2. verify Hypothesis 1

For item 1, since the Decode algorithm takes coefficients of $\mathbf{S}$ four by four, the size of the smallest target is a quadruplet of coefficients of $\mathbf{S}$. Actually, for a given quadruplet of integers $\underline{\ell} = (\ell_0, \ell_1, \ell_2, \ell_3)$ and a target index k (i.e. an index corresponding to the components of $\mathbf{S}$ that Eve wants to retrieve), by taking

$$\mathbf{U} = sx^{-k} \quad \text{and } \mathbf{c} = \sum_{j=0}^{3} \Big((\ell_j + 4) \mod 8 \Big) \cdot x^{nj} \tag{5}$$

one can prove (see in Proposition 1) that Eve targets the quadruplet $\big(\mathbf{S}[k + nj]\big)_{j=0\cdots 3}$. Indeed, the element x^{-k} will "rotate" $\mathbf{S}$ in order to target $\big(\mathbf{S}[k + nj]\big)_{j=0\cdots 3}$ and $\mathbf{c}$ is induced by the quadruplet $\underline{\ell} = (\ell_0, \ell_1, \ell_2, \ell_3)$ that can vary.

About item 2, with this choice of $(\mathbf{c}, \mathbf{U})$, the Hypothesis 1 has good chances to be verified because the coefficients of $\mathbf{c}$ outside from the set $\{k + nj|\ j = 0 \cdots 3\}$ are 0. So, the same coefficients of $\mathbf{C} - \mathbf{US}$ have good chances to be small. Then, Alice is likely to derive 0 for these coefficients of ν_A. However, it is not always verified and this will impact the attack's success rate. We will discuss and compute this probability later in Proposition 3.

We can now introduce $\mathcal{O}_3$, a reformulation of $\mathcal{O}_2$ depending on target index k and the quadruplet $\underline{\ell}$ (see equation 5):

$$\mathcal{O}_3(k, \underline{\ell}) = \mathcal{O}_2 \left(sx^{-k}, \sum_{j=0}^{3} \Big((\ell_j + 4) \mod 8 \Big) \cdot x^{nj} \right)$$

This formulation of the key mismatch oracle is more convenient in order to explain how Eve will gather information on $\mathbf{S}$ from instantiations of $\underline{\ell}$. The following proposition shows a first result in this direction.

Proposition 1. ***Final oracle.*** *Let us assume that Hypothesis 1 is verified. Let k be a target index ($k \in [0, n-1]$). For a given integer quadruplet $\underline{\ell}$ in $[-4, 3]^4$, the $(\boldsymbol{c}, \boldsymbol{U})$ explicited in Equation 5 is such that*

$$\mathcal{O}_3(k, \underline{\ell}) = Sign \left(\sum_{j=0}^{j=3} \Big| \ell_j - \boldsymbol{S}[k + nj] \Big| - 8 \right)$$

Proof. The proof is given in Appendix B.

In the next section, we explain how to effectively use the form $\mathcal{O}_3$ of the key mismatch oracle to extract the secret $\mathbf{S}$.

4.2 Recovering very small coefficients of $\mathbf{S}$

Let us recall that the secret $\mathbf{S}$ is a polynomial in $\mathbb{Z}_q[x]/(x^N+1)$ with coefficients in $[-8,8]$, it can be seen as a vector of N components $\mathbf{S}[i]$. Eve will recover the coefficients of the secret $\mathbf{S}$ four by four. Let k be the index of the targeted quadruplet $[\mathbf{S}[k],\mathbf{S}[k+n],\mathbf{S}[k+2n],\mathbf{S}[k+3n]]$. The index k goes from 0 to $n-1$ and for each fixed k, Eve will call the oracle $\mathcal{O}_3(k,\underline{\ell})$ with several appropriate value of $\underline{\ell}$ until she gets the secret values.

For simplicity, let us now **fix the index** k and denote $S_j=\mathbf{S}[k+nj]$.

The following proposition and corollary describe an algorithm that, when iterated (see Corollary 1), allows to recover S_j for j from 0 to 3.

Proposition 2. *Let us fix j in $[0,3]$. Under Hypothesis 1, if S_j is in $[-3,2]$ and $(S_i)_{i\neq j}\in[-4,4]$, there exists a probabilistic algorithm $\mathcal{A}$ which recovers the value S_j in 8 queries to oracle $\mathcal{O}_3$ with a success probability depending on the distribution of $(S_i)_{0\leq i\leq 3}$.*

Corollary 1 *Under Hypothesis 1, if S_j is in $[-3,2]$ and $(S_i)_{i\neq j}\in[-4,4]$, there exists a probabilistic algorithm $\mathcal{A}'$ which recovers the value S_j with an average number of queries to oracle $\mathcal{O}_3$ depending on the distribution of $(S_i)_{0\leq i\leq 3}$.*

In the sequel of this section, we give the proof of Proposition 2 by first presenting the construction of the algorithm and then by introducing a method to assess the success rate. We refer the reader to Appendix D for the proof of Corollary 1.

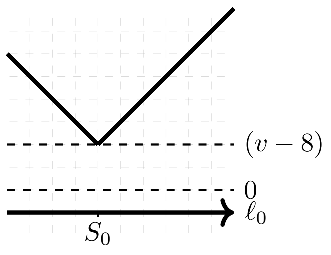
Proof of Proposition 2

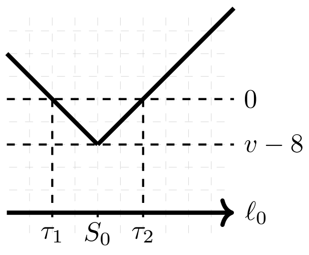
Description of $\mathcal{A}$. Let us prove the proposition by focusing on the secret S_0 and by explaining how it can be recovered in 8 queries to oracle $\mathcal{O}_3$. The process will then be exactly the same for the three other values S_1,S_2 and S_3.

The first step consists in taking the 3 values ℓ_1,ℓ_2,ℓ_3 at random inside the interval $[-4,3]$. Knowing that all S_j are fixed, the quantity $\sum_{j=0}^{3}|\ell_j-S_j|-8$ can thus be expressed by $f_v(\ell_0)=\big|\ell_0-S_0\big|+v-8$ with $v=\sum_{i=1}^{3}|\ell_j-S_j|$ a fixed unknown constant (since all S_j are unknown). Let us now see how $f_v(\ell_0)$ behaves when ℓ_0 varies, see Figure 3 for an illustration.

We now assume that one makes 8 queries to the oracle $\mathcal{O}_3$: one for each value of ℓ_0 varying inside $[-4,3]$. Such queries imply having access to $Sign\Big(f_v(\ell_0)\Big)$ $\forall\ell_0\in[-4,3]$. The analysis can thus be split in 2 cases:
1. If $(v-8)\geq 0$ then all queries to oracle 1 obviously lead to "positive signs". It is quite clear when one looks at Figure 3.

2. If $(v-8)<0$, two subcases occur





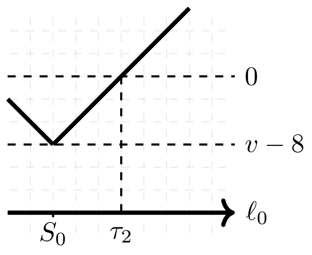


Fig. 3. If $v - 8 \geq 0$

Fig. 4. If $v - 8 > 0$

Fig. 5. If $v - 8 \gg 0$

– In some cases, there exists two possible values $\tau_1 < \tau_2$ such that the function $|\ell_0 - S_0| + (v - 8)$ goes from a positive value to a negative one at point τ_1 and then from a negative value to a positive one at point τ_2. We call this case the *favorable case*. Figure 4 provides a good illustration.

ℓ_0	-4	$\cdots$	$\tau_1 - 1$	τ_1	$\tau_1 + 1$	$\cdots$	$\tau_2 - 1$	τ_2	$\tau_2 + 1$	$\cdots$	3
$\mathcal{O}$	+	$\cdots$	+	+	-	$\cdots$	-	+	+	$\cdots$	+

– If $(v - 8) < 0$ and $v \ll 8$, only one change of sign will occur in the interval $[-4, 3]$. Figure 5 provides a good illustration.

Figure 4 illustrated what happens in the *favorable case*. Around S_0, the trace has a slope equal to + or - 1. Because of the symmetry, the value S_0 can simply be recovered by:

$$S_0 = \frac{\tau_2 + \tau_1}{2} \,. \tag{6}$$

If we are not in the *favorable case*, two such values τ_1 and τ_2 do not exist. This means that the constant v is not appropriate.

Termination of $\mathcal{A}$. For any $S_0 \in [-3, 2]$, $\mathcal{A}$ has a non zero success probability. Indeed, no matter the values of (S_1, S_2, S_3) in $[-4, 4]^3$, the 3-uple $(\ell_1, \ell_2, \ell_3) \in [-4, 3]^3$ defined by

$$\Big(\ell_1 = S_1 - 2 \cdot Sign(S_1), \quad \ell_2 = S_2 - 2 \cdot Sign(S_2), \quad \ell_3 = S_3 - 3 \cdot Sign(S_3) \Big)$$

is at least one of the choices inducing a favorable case. Actually, one can check that this choice implies that $v = 7$. Thus $v - 8 = -1$ which always gives a favorable case for finding $S_0 \in [-3, 2]$.

Success probability. A precise probabilistic study on the $(S_j)_{1 \leq j \leq 3}$ to assess the success rate of algorithm $\mathcal{A}$ is detailed in Appendix C. In Table 1, one can find the probability of success assuming that S_1, S_2, S_3 follow a binomial distribution ψ_4. The expected number of iterations is the average amount of tries before recovering the secret. $\square$

Table 1. Success probability of $\mathcal{A}$ for $(S_j)_{1\leq j\leq 3}$ following ψ_4 distribution

S_0	-3	-2	-1	0	1	2
Probability (%)	14	27	39	39	27	14
Expected number of iterations ($1/probability$)	7.1	3.7	2.6	2.6	3.7	7.1

Example 1. Let us suppose that $S_i = [0, -2, 1, -1]$. For $(\ell_1, \ell_2, \ell_3) = (2, -2, -1)$, $\sum_{j=0}^{3} |\ell_j - S_j| - 8 = |\ell_0 - S_0| - 2$. If we query the sign of the latter for $\ell_0 = -4, -3, -2, -1, 0, 1, 2, 3$, we get : $+$, $+$, $+$, **-**, **-**, **-**, $+$, $+$. We can conclude that $S_0 = \frac{1-1}{2} = 0$. Whereas, for $(\ell_1, \ell_2, \ell_3) = (-2, 0, 1)$, $\sum_{j=0}^{3} |\ell_j - S_j| - 8 = |\ell_0 - S_0| - 5$. The sign for $\ell_0 = -4, -3, -2, -1, 0, 1, 2, 3$ becomes : **-**, **-**, **-**,**-**, **-**, **-**,**-**, **-**. We cannot conclude anything on S_0.

At the end of this section, with Corollary 1, we know that if $\mathbf{S}$ is generated with coefficients following the ψ_4 distribution and if Hypothesis 1 is verified, there exist an algorithm that recovers each coefficient of $\mathbf{S}$ that is in $[-3, 2]$ (i.e. almost 96% of the coefficients). If a coefficient of $\mathbf{S}$ is not in $[-3, 2]$, no favorable case will appear and the coefficient will not be found. In the next section, we adapt this method for NewHope.

4.3 Recovering S for NewHope parameters

In this section, we describe a way to recover $\mathbf{S}$ for NewHope parameters, *i.e.* when the binomial parameter is 8. According to Property 1, the coefficients of $\mathbf{S}$[k] are in $[-8, 8]$. This is outside from the hypothesis made in Proposition 2. Indeed, the coefficients $\mathbf{S}$[k] should lie in $[-3, 2]$. One can make the following change in order to fit with Proposition 2 hypothesis : $\mathbf{S}^1 = \frac{\mathbf{S}}{2}$. In order to target $\mathbf{S}^1$ instead of $\mathbf{S}$, one can change $\mathbf{U}$ from Equation 5 to be the following $\mathbf{U} = \frac{s}{2}x^{-k}$.

Let us wrap up the attack into the following Proposition.

Proposition 3. *There exists a probabilistic algorithm $\mathcal{B}$ which recovers NewHope secret $\boldsymbol{S}$ with high probability using an average of* 18,500 *queries for* $N = 1024$.

Proof. Let $k \in [0, n-1]$. The distribution of probabilities for $\mathbf{S}[k]$ is in Table 2

Table 2. Distribution ψ_8 (note that the probability is the same for negative values)

$\mathbf{S}[k]$	0	1	2	3	4	5	6	7	8
Probability ($\times 2^{16}$)	12870	11440	8008	4368	1820	560	120	16	1

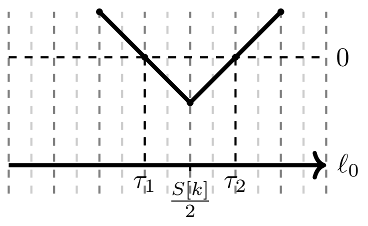


Fig. 6. When $\mathbf{S}[k] \bmod 2 = 0$

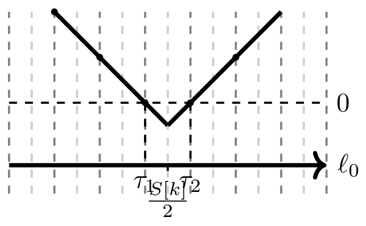


Fig. 7. When $\mathbf{S}[k] \bmod 3 = 1$

Case 1: $\mathbf{S}[k]$ belongs to $\{-8, -7, 5, 6, 7, 8\}$. The probability of this case is around 1%. In that case, at most one change of sign will always happen. Then, only the sign of $\mathbf{S}[k]$ can be recovered and a brute force should be done at the end of the attack to distinguish among the possible values. For $N = 1024$, on average 10 coefficients out of 1024 will not be found. When a positive value is not found, it has 8/10 chances to be a 5. At the end of the attack, a bruteforce step evaluating $\mathbf{B} - \mathbf{AS}$ and taking account of the probabilities can be done.

Case 2: $\mathbf{S}[k]$ belongs to $\{-6, ..., 4\}$. In that case, $\mathbf{S}^1[k]$ belongs in the interval $[-3, 2]$. The attack is the one from Proposition 2 with a different secret $\mathbf{S}^1[k]=\frac{\mathbf{S}[k]}{2}$. However, the results will not be as accurate as before. We will show that there is a subtelty that allows Eve to recover the exact value of $\mathbf{S}[k]$.

There are 2 possible results depending on $\mathbf{S}[k] \bmod 2$:

– If $\mathbf{S}[k] \bmod 2 = 0$, then $\mathbf{S}^1[k] \in \{-3, -2, -1, 0, 1, 2\}$. Proposition 2 allows Eve to recover $\mathbf{S}^1$. In other words, Eve will recover a succession of signs where an odd number of $(-)$ occurs (see figure 6). She will then be able to recover $\frac{\mathbf{S}[k]}{2} = \frac{\tau_1+\tau_2}{2}$ and then $\mathbf{S}[k]$.

– If $\mathbf{S}[k] \bmod 2 = 1$ then $\mathbf{S}^1[k] \in \{-2.5, -1.5, -0.5, 0.5, 1.5\}$. In a favorable case of proposition 2, the situation will be different. As in figure 7, the number of $(-)$ is then even.

Wrap up. Here is a procedure to recover $\mathbf{S}[k]$.

Case 1 If the number of $(-)$ is odd, then $\mathbf{S}[k]$ is even and $\mathbf{S}[k] = 2\frac{\tau_1+\tau_2}{2} = \tau_1 + \tau_2$

Case 2 If the number of $(-)$ is even, then $\mathbf{S}[k] = 2\left\lfloor \frac{\tau_1+\tau_2}{2} \right\rfloor + 1$

Case 3 If at most one change of sign occur, the procedure is restarted.

If the number of restarts is too large (say, $\geq M$), the procedure is stopped and the coefficient, placed in a bruteforce set, is found at the end of the attack.

Number of queries The amount of queries is derived with the same technique as in Appendix D. See Table 3 for the average number of queries. Let us set the threshold M to 50, to get the total average number of queries, we compute the

Table 3. Average number of queries

Value	-6	-5	-4	-3	-2	-1	0	1	2	3	4	-8,-7,5, 6, 7 or 8
Average queries	33	33	19	20	16	17	17	15	22	20	38	M

Algorithm 5: Key recovery algorithm

```
Output: S
for $k := 0$ to $n-1$ do
    $\mathbf{U} \leftarrow \frac{s}{2} x^{-k}$
    for $j := 0$ to $3$ do
        secret-coeff $\leftarrow \perp$
        while $secret\text{-}coeff = \perp$ and $nbQueries \leq Maxqueries$ do
            $(\ell_0, \ell_1, \ell_2, \ell_3) \xleftarrow{\$} [-4, 3]^4$
            $b \leftarrow ZeroMatrix(8)$
            for $i := -4$ to $3$ do
                $\ell_j \leftarrow i$
                $\mathbf{c} \leftarrow \sum_{j=0}^{3} (\ell_j + 4) \mod 8 x^{nj}$
                $b[i] \leftarrow \mathcal{O}_1(Encode(\mathbf{c}, \mathbf{U}), \text{SHAKE-256}(32, (1, 0, ..., 0)))$
            end
            secret-coeff $\leftarrow FindS(b)$
        end
        $\mathbf{S}[k + nj] \leftarrow$ secret-coeff
    end
end
Return S
```

expected number of queries for $\mathbf{S}[k]$ (≈ 18) and multiply it by $N = 1024$.

Success probability. The success probability depends only on Hypothesis 1 with $\mathbf{S}^1$, which becomes the following for $\mathbf{S}$.

Hypothesis 2. $\forall i, k \in [1, n-1]$ $\sum_{j=0}^{j=3} \left| \frac{\mathbf{S}[k+i+nj \mod N]}{2} + 4 \right| \geq 8$

Hypothesis 2. is true with a probability 94.6% for $N = 1024$. Indeed, to compute this probability, one can check whether each quadruplet verifies it. Only a few unlikely quadruplet (e.g. $[8, 8, 8, 8]$) do not verify the hypothesis.

$\square$

The pseudo code of the attack is provided in Algorithms 5 and 6.

4.4 Experimental Results

We implemented a proof of concept with Magma CAS [4][11]. We coded NewHope according to its parameters and used the key mismatch oracle for the attack. We worked on a basic optimization of the number of queries. We ran 1000 experiments and recovered more than 95% of the secret keys in an average time of 30 minutes per key and 16,700 queries. We still think that the number of queries and the time can be better optimized.

[11]The Magma code can be found at https://www.di.ens.fr/~mrossi/

Algorithm 6: function *FindS*

```
Data: $b$
Output: secret-coeff
$\tau_1 \leftarrow \perp$                    //index of the change of sign $(+) \rightarrow (-)$
$\tau_2 \leftarrow \perp$                    //index of the change of sign $(-) \rightarrow (+)$
for $i := -3$ to $2$ do
    if $b[i-1] = (+)$ and $b[i] = (-)$ then
        $\tau_1 = i$                         // First sign change
    if $b[i] = (-)$ and $b[i+1] = (+)$ then
        $\tau_2 = i$                         //Second sign change
    end
end
$\tau \leftarrow \tau_1 + \tau_2$
if $\tau \mod 2 = 0$ then
    secret-coeff $\leftarrow \tau$
if $\tau \mod 2 = 1$ then
    secret-coeff $\leftarrow 2\lfloor \frac{\tau}{2} \rfloor + 1$
else
    secret-coeff $\leftarrow \perp$          //In that case, $\tau = \perp$
Return secret-coeff
```

Algorithm 7: NewHope CCA-KEM Key Generation

```
function NewHope CCA-KEM.Gen()
$(pk, sk) \leftarrow$ NewHope-CPA-PKE.Gen()
$s \leftarrow \{0, \ldots, 255\}^{32}$
return $(pk, \bar{sk} = (sk||pk||\text{SHAKE-256}(32, pk)||s)$
```

Algorithm 8: NewHope-CCA-KEM Encapsulation

```
function NewHope-CCA-KEM-Encaps($pk$)
$coin \leftarrow \{0, \ldots 255\}^{32}$
$\mu \leftarrow \text{SHAKE-256}(32, coin) \in \{0, \ldots, 255\}^{32}$
$K||coin'||d \leftarrow$
  $\text{SHAKE-256}(96, \mu||\text{SHAKE-256}(32, pk)) \in$
  $\{0, \ldots, 255\}^{32+32+32}$
$c \leftarrow$NewHope-CPA-PKE.Encrypt$(pk, \mu; coin')$
$ss \leftarrow \text{SHAKE-256}(32, K||\text{SHAKE-256}(32, c||d))$
return $(\bar{c} = c||d, ss)$
```

5 Accessing the **key mismatch oracle** with the CCA version of **NewHope**

In order to be protected against active attacks, the CPA-KEM of NewHope has been transformed according to the Hofheinz, Hövelmanns and Kiltz CCA transformation [15] which is a variant of the Fujisaki-Okamoto transformation [10]. The CCA security is then based on the CPA security of the PKE. The CCA transformation of the algorithms defining this version of NewHope is detailed in Algorithms 7, 8 and 9. These algorithms use the underlying CPA-PKE of NewHope as defined in Section 1.2.1 of [22].

One can note the main security measure in Algorithm 9 where the instruction in red corresponds to a double encryption to check if the message m_B has been honestly generated.

More precisely, in the key mismatch oracle, the message m_B can be adjusted by the attacker but with this CCA version of NewHope, Eve must follow the protocol and generate m_E according to a seed called *coin*$'$ that is derived from μ_E and another seed called *coin* (a 32-byte random integer). Then, Alice will

Algorithm 9: NewHope-CCA-KEM Decapsulation

```
1  function NewHope-CCA-KEM.Decaps ($\bar{c}$, $\bar{sk}$)
2  $c||d \leftarrow \bar{c} \in \{0, \ldots, 255\}^{\text{NEWHOPE\_CPAPKE\_CIPHERTEXTBYTES}+32}$
3  $sk||pk||h||s \leftarrow \bar{sk} \in \{0, \ldots, 255\}^{32+32+32+32}$
4  $\mu' \leftarrow$ NewHope-CPA-PKE.Decrypt($c$, $sk$)
5  $K'||coin''|d' \leftarrow$ SHAKE-256($96, \mu'||h$) $\in \{0, \ldots, 255\}^{32+32+32}$
6  if $c =$NewHope-CPA-PKE.Encrypt($pk, \mu'; coin''$) and $d = d'$
7  then $fail \leftarrow 0$ else $fail \leftarrow 1$ end if
8  $K_0 \leftarrow K'$
9  $K_1 \leftarrow s$
10 return $ss =$ (SHAKE-256(32, $K_{fail}$||SHAKE-256(32, $c||d$))
```

derive $coin'$ to check if μ_E was computed following the protocol. Then, a key mismatch will come from the following oracle

$$\mathcal{O}_4(coin, \mu_E) = \begin{cases} 1 & \text{if } \mathsf{Dec}(\mathsf{Enc}(m_A, coin), \mathbf{S}) = \mu_E \\ -1 & otherwise \end{cases} \tag{7}$$

This oracle is less convenient than $\mathcal{O}_1$ because with an honest behaviour, the error probability is claimed to be lower than 2^{-213} in the NIST specification (paragraph 4.2.7 of [22]). In the sequel, we point at critical steps inside the CCA transform that let Eve access oracle $\mathcal{O}_1$ using side channel or fault attacks.

On using side channel or fault attack

When the attacker has access to a device implementing Alice's side of the exchange, the attack model should take into account situations where some algorithmic security measures may be bypassed by using hardware attacks.

Power analysis. Here we consider that Eve is able to make a power analysis during the verification step of Alice decapsulation algorithm.

```
* Name:          verify
* Description:  Compare two arrays for equality in constant time.
* Arguments:
* const unsigned char *a:  pointer to first byte array
* const unsigned char *b:  pointer to second byte array
* size_t len:              length of the byte arrays
* Returns 0 if the byte arrays are equal, 1 otherwise
**************************************************/

int verify(const unsigned char *a, const unsigned char *b, size_t len)
{
  uint64_t r;
  size_t i;
  r = 0;

  for(i=0;i<len;i++)
    r |= a[i] ^ b[i];

  r = (-r) >> 63;
  return r;
}
```

Fig. 8. Code for NewHope-CCA-KEM verification step

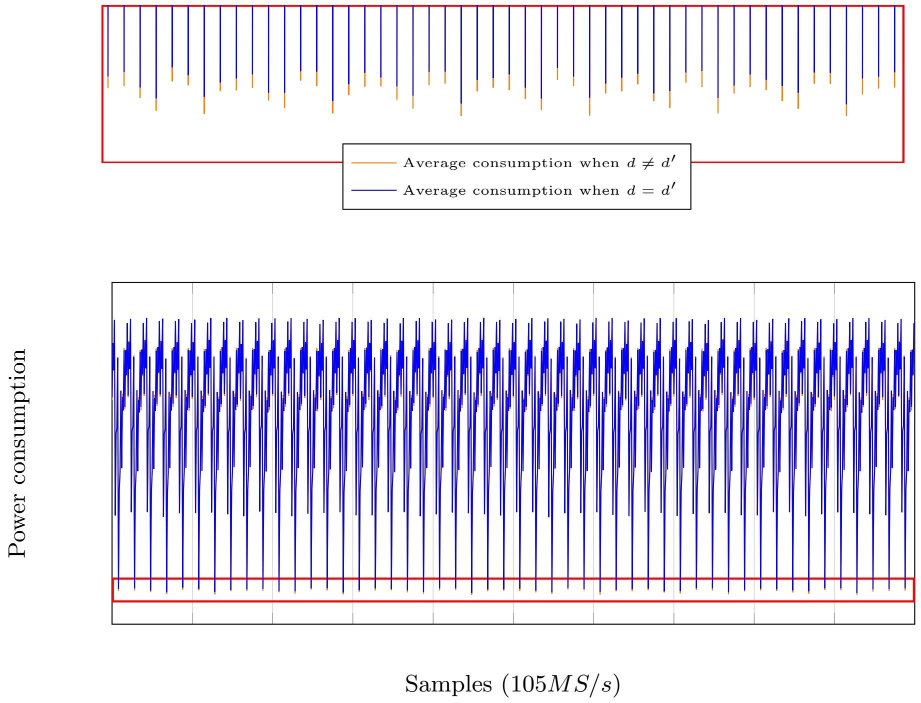


Fig. 9. Single Trace leakage : this trace represents the power consumption while computing the loop of the verify function, the red rectangle above represents a zoom on the red rectangle in the lower part.

Attack Model 2 *We assume that Alice has done the CCA key generation. Eve sends messages m_E with a wrong coin. Alice will then reject any messages m_e because the verification is never passed. Eve, the attacker, is able to make a power analysis during the verification step of Alice's* decapsulation *algorithm.*

A first idea would be to target the testing of the equality $d = d'$ with single power analysis. The code in figure 10 corresponds to NewHope-CCA-KEM verification step where $a = (c, d)$, $b = (\text{NewHope-CPA-PKE.Encrypt}(pk, \mu'; coin''), d')$ and $len = 17/8 \cdot N + 32$.

This naive method actually works well in practice for an unprotected scheme because when $d = d'$, r is ored with 0 during $17/8 \cdot N$ iterations and when $d \neq d'$, r is ored with arbitrary values during $17/8 \cdot N$ iterations. With a single trace analysis, the equality $d = d'$ can then leak. To check it, we set up an experiment with a Chipwhisperer Lite (XMEGA 128D4). In Figure 9, we represent a power consumption trace measured for the loop of the verify function. The blue trace corresponds to the average consumption when $d = d'$ and the orange trace represents the average consumption when $d \neq d'$. If we zoom in (see the red rectangle), we can remark that, on our plateform, the trace for $d \neq d'$ always consumes more than the one for $d = d'$.

One would reasonably argue that unprotected schemes are always vulnerable. The main aim of [20] is to propose a countermeasure to such an attack for a similar scheme which uses the CCA transform. It is an open problem, in this protected context, to extend this approach to a second order power analysis attack. A more realistic model relies on an invasive attack, this is what we present in the sequel.

Single Fault Attack. We consider inserting a fault during the computation of the verification step which cancels the CCA transform. The attack model becomes:

Attack Model 3 *We assume that Alice has done the* ***CCA*** *key generation. Eve, the attacker, is able to set the value r to 0 in the verification step of Alice's* decapsulation *algorithm.*

If Eve is able to set the value r to 0 anytime during the check $d = d'$, she can bypass the reencryption and the mismatch will appear only if $d \neq d'$. Then oracle $\mathcal{O}_1$ becomes accessible. Indeed, Eve can thus send any message m_E. Alice derives a wrong $coin'$ but the verification is skipped with the fault. If $d = d'$, Alice derives the shared key for initiating a communication. If $d \neq d'$, Alice will notice Eve that the key agreement failed. Eve will then deduce that the key is different. This vulnerability has been underlined in Section 3.6 of [20] for a similar scheme. But no countermeasure has been added to protect against this single fault attack, which can practically be induced by a laser. Countermeasures should have thus to be considered (see [17]), what may impact the efficiency of the verification.

6 Conclusion

The resilience of NIST post-quantum candidate algorithms in misuse situations is worth being investigated. It will help developers to propose implementations with countermeasures tightly designed to ensure the security in extreme contexts (e.g. smart card, IoT) without decreasing too much the efficiency. In this paper, we describe an active attack against NewHope-CPA-KEM with (public, private) key pair reuse. This clearly confirms that if the designers' caveat against any private key reuse (e.g. temporary caching) is not strictly followed, this results in a practical, low complexity, key recovery attack. Our study indeed indicates that setting an upper limit of a few hundreds on the number of authorized key reuses would not be conservative enough, and already expose private keys to significant information leakages. While unprotected versions of CCA-KEM are extremely efficient, their implementation must be very carefully protected against any key mismatch oracle leakage if key pairs are potentially reused. As explained in this paper, this is particularly true for countermeasures against fault attacks. This might eventually come with a cost in terms of performance. This consideration may become even more important if one considers second order side channel or combined attacks, which could be a sequel of this work.

References

1. E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe. Post-quantum key exchange - A new hope. In T. Holz and S. Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 327–343. USENIX Association, 2016.
2. D. J. Bernstein, L. Groot Bruinderink, T. Lange, and L. Panny. HILA5 Pindakaas: On the CCA Security of Lattice-Based Encryption with Error Correction. In *AFRICACRYPT 2018*, pages 203–216, Cham, 2018. Springer.
3. D. Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, pages 1–12, 1998.
4. W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).
5. M. Braithwaite. Experimenting with Post-quantum Cryptography. Posting on the Google Security Blog, 2016.
6. J. Ding. A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem. *IACR Cryptology ePrint Archive*, 2012:688, 2012.
7. J. Ding, S. Alsayigh, R. V. Saraswathy, S. R. Fluhrer, and X. Lin. Leakage of Signal Function with reused Keys in RLWE Key Exchange. In *IEEE International Conference on Communications, ICC 2017*, pages 1–6. IEEE, 2017.
8. J. Ding, S. Fluhrer, and R. V. Saraswathy. Complete Attack on RLWE Key Exchange with Reused Keys, without Signal Leakage. In *Information Security and Privacy*, pages 467–486, Cham, 2018.
9. S. Fluhrer. Cryptanalysis of Ring-LWE based Key Exchange with Key Share Reuse. Cryptology ePrint Archive, Report 2016/085, 2016. https://eprint.iacr.org/2016/085.
10. E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In *CRYPTO LNCS*, pages 537–554, 1999.
11. Q. Guo, T. Johansson, and P. Stankovski. A key recovery attack on MDPC with CCA security using decoding errors. In J. H. Cheon and T. Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 789–815, 2016.
12. C. Hall, I. Goldberg, and B. Schneier. Reaction Attacks against Several Public-Key Cryptosystems. In V. Varadharajan and Y. Mu, editors, *Information and Communication Security*, pages 2–12, Berlin, Heidelberg, 1999. Springer.
13. J. Hoffstein and J. H. Silverman. Protecting NTRU Against Chosen Ciphertext and Reaction Attacks. Technical Report 16, NTRU Cryptosystems Technical Report.
14. J. Hoffstein and J. H. Silverman. Reaction Attacks Against the NTRU Public Key Cryptosystem. Technical Report 15, NTRU Cryptosystems Technical Report.

15. D. Hofheinz, K. Hövelmanns, and E. Kiltz. A Modular Analysis of the Fujisaki-Okamoto Transformation. In Y. Kalai and L. Reyzin, editors, *Theory of Cryptography*, pages 341–371, Cham, 2017. Springer.
16. N. Howgrave-Graham, P. Q. Nguyen, D. Pointcheval, J. Proos, J. H. Silverman, A. Singer, and W. Whyte. The Impact of Decryption Failures on the Security of NTRU Encryption. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, pages 226–246, 2003.
17. M. Joye and M. Tunstall, editors. *Fault Analysis in Cryptography*. Information Security and Cryptography. Springer, 2012.
18. D. Kirkwood, B. Lackey, J. McVey, M. Motley, J. Solinas, and D. Tuller. Failure is not an Option : Standardization Issues for Post-quantum Key Agreement, 2015. NIST Workshop On Cybersecurity in a Post Quantum World.
19. A. Menezes and B. Ustaoglu. On Reusing Ephemeral Keys in Diffie-Hellman Key Agreement Protocols. *IJACT*, 2(2):154–158, 2010.
20. T. Oder, T. Schneider, T. Pöppelmann, and T. Güneysu. Practical CCA2-Secure and Masked Ring-LWE Implementation. IACR Transactions on CHES, 2016. https://eprint.iacr.org/2016/1109.
21. C. Peikert. Lattice Cryptography for the Internet. In *PQCrypto 2014, Canada. Proceedings*, pages 197–219, 2014.
22. T. Pöppelmann, E. Alkim, R. Avanzi, J. Bos, L. Ducas, A. de la Piedra, P. Schwabe, and D. Stebila. NewHope, 2017. Submission to Round 1 of NIST Post Quantum Cryptography Competition.
23. P. Rogaway and T. Shrimpton. A provable-security treatment of the key-wrap problem. In *Advances in Cryptology - EUROCRYPT 2006*, pages 373–390, 2006.
24. P. Schwabe, E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe. NewHope without Reconciliation. Cryptology ePrint Archive, Report 2016/1157, 2016. https://eprint.iacr.org/2016/1157.

Appendices

A Variant when Bob is attacked

Another point of view that was investigated in [7] is when Bob is static.

Attack Model 4 *The client Bob generates static private secrets **S'**, **E'**, and **E"** and keeps them for multiple communications with honest servers (Alice). The attacker, (Eve) is only able to observe the different public messages sent by Bob and the servers.*

In NewHope, this part is a less likely scenario because the server is able to modify $\mathbf{A}$. Then, if 2 servers communicate with different $\mathbf{A}$ with a static Bob. Then anyone curious can recover Bob's secret. Here is a simple explanation.
In that case, Bob generates static secrets **S'**, **E'**, and **E"**.

1. There is no need to recover **E"**. Indeed, it is generated to be small and Compress removes all the small parts. Thus, its value does not influence the protocol.
2. One possible method to recover **S'** and **E'** by observing 2 queries with different $\mathbf{A}$ can be the following:
   - We observe a first key exchange $m_{E,1} = \mathsf{Encode}(seed_1, \mathbf{B}_1)$ where $\mathbf{A}_1 = Parse(\text{SHAKE-128}(seed_1))$ and $m_{B,1} = \mathsf{Encode}(\mathbf{C}_1, \mathbf{U}_1)$ sent by Bob.
   - We observe a second key exchange $m_{E,2} = \mathsf{Encode}(seed_2, \mathbf{B}_2)$ where $\mathbf{A}_2 = Parse(\text{SHAKE-128}(seed_2)) \neq \mathbf{A}_1$ and $m_{B,2} = \mathsf{Encode}(\mathbf{C}_2, \mathbf{U}_2)$ sent by Bob.
   At that point, $\mathbf{A}_1, \mathbf{A}_2, \mathbf{U}_1, \mathbf{U}_2$ are public. Since
   $$\mathbf{U}_1 = \mathbf{A}_1\mathbf{S'} + \mathbf{E'} \text{ and } \mathbf{U}_2 = \mathbf{A}_2\mathbf{S'} + \mathbf{E'} \tag{8}$$
   Then
   $$\mathbf{S'} = \frac{\mathbf{U}_1 - \mathbf{U}_2}{\mathbf{A}_1 - \mathbf{A}_2} \text{ and } \mathbf{E'} = \mathbf{A}_1\mathbf{S'} - \mathbf{U}_1 \tag{9}$$

Attack Model 5 *The client Bob generates static private secrets **S'**, **E'**, and **E"**. A malicious server, Eve can initiate key establishment with Bob which respond honestly.*

As shown before, a malicious server (Eve) can retrieve Bob's secrets with only 2 queries if she changes the value of the *seed* for $\mathbf{A}$.

B Proof of Proposition 1

Proof. Let k be an index defined in $[0, n-1]$ and let us fix $\underline{\ell} = (\ell_0, \ell_1, \ell_2, \ell_3) \in [-4, 3]^4$. We show how the explicit values for $(\mathbf{c}, \mathbf{U})$ given in Equation 5 provide the expected result. Let us compute the values $\mathsf{Decompress}(c)[nj]$:

$$\begin{aligned}\mathsf{Decompress}(c)[nj] = &\left\lceil \tfrac{(\ell_j+4 \mod 8)\times q}{8} \right\rfloor = \left\lceil (\ell_j + 4 \mod 8)s + \tfrac{1}{8} \right\rfloor \\ &= (\ell_j + 4 \mod 8)s\end{aligned}$$

Under Hypothesis 1 the equation given by Lemma 1 thus becomes:

$$Sign\left(\sum_{j=0}^{j=3} \left| (\ell_j + 4 \mod 8).s - s\mathbf{S}[k+nj] - 4s \right| - q\right)$$

The next step is to divide the previous equation by s. Since $\frac{q}{s} \approx 8.0007$, $\sum_{j=0}^{j=3} |(\ell_j + 4 \mod 8) - \mathbf{S}[k+nj] - 4| \in \mathbb{Z}$ and by convention $Sign(0) > 0$, we get the following :

$$\mathcal{O}_3(k, \underline{\ell}) = Sign\left(\sum_{j=0}^{j=3} \left| (\ell_j + 4 \mod 8) - \mathbf{S}[k+nj] - 4 \right| - 8 \right)$$

Since $\underline{\ell}$ belong in $[-4, 3]^4$, we have

$$(\ell_j + 4 \mod 8) = \ell_j + 4 \, .$$

Then we get

$$\mathcal{O}_3(k, \underline{\ell}) = Sign\left(\sum_{j=0}^{j=3} \left| \ell_j - \mathbf{S}[k+nj] \right| - 8 \right)$$

$\square$

C Success rate of $\mathcal{A}$

Let us introduce a method to estimate the success rate of $\mathcal{A}$. We still focus on targetting the secret S_0. Since the success depends on S_0 and on the distribution of $(S_j)_{1 \le j \le 3}$, let us suppose that

- $(S_j)_{1 \le j \le 3}$ follow a binomial distribution of parameter 4, ψ_4. According to property 1, $(S_j)_{1 \le j \le 3} \in [-4, 4]$
- $S_0 \in [-3, 2]$

For any $\forall j \in [1, 3]$, the probability of $|\ell - S_j| = w$ when $w \in [0, 8]$ and ℓ uniformly random in $[-4, 3]$ can be seen in Table 4.

Table 4. Probability of $|\ell - S_j| = w$ depending on S_j with ℓ uniformly random in $[-4, 3]$

w	0	1	2	3	4	5	6	7	8
$S_j = 4$	0	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125
$S_j = 3$	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0
$S_j = 2$	0.125	0.25	0.125	0.125	0.125	0.125	0.125	0	0
$S_j = 1$	0.125	0.25	0.25	0.125	0.125	0.125	0	0	0
$S_j = 0$	0.125	0.25	0.25	0.25	0.125	0	0	0	0
$S_j = -1$	0.125	0.25	0.25	0.25	0.125	0	0	0	0
$S_j = -2$	0.125	0.25	0.25	0.125	0.125	0.125	0	0	0
$S_j = -3$	0.125	0.25	0.125	0.125	0.125	0.125	0.125	0	0
$S_j = -4$	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0

From Table 4, we derive the probability for w :

$\mathcal{P}(w = 0) = 0.125(1 - \mathcal{P}[S_j = 4])$
$\mathcal{P}(w = 1) = 0.25 - 0.125\mathcal{P}[S_j \in \{4, 3, -4\}]$

$\mathcal{P}(w = 2) = 0.25 - 0.125\mathcal{P}[S_j \in \{4, 3, 2, -3, -4\}]$
$\mathcal{P}(w = 3) = 0.125(1 + \mathcal{P}[S_j \in \{0, -1\}])$
$\mathcal{P}(w = 4) = 0.125$
$\mathcal{P}(w = 5) = 0.125(1 - \mathcal{P}[S_j \in \{0, -1\}])$
$\mathcal{P}(w = 6) = 0.125\mathcal{P}[S_j \in \{4, 3, 2, -3, -4\}]$
$\mathcal{P}(w = 7) = 0.125\mathcal{P}[S_j \in \{4, 3, -4\}]$
$\mathcal{P}(w = 8) = 0.125\mathcal{P}[S_j = 4]$

Then, for $v - 8 \in [-8, 16]$, we can compute the probability, denoted P_v, of getting $|\ell_1 - S_1| + |\ell_2 - S_2| + |\ell_3 - S_3| = v - 8$ with ℓ_j varying in $[4, 3]$ and $S_j \in [-4, 4]$ (for $j = 1$ to 3).

$$P_v = \mathcal{P}_{l_j \in [-4,3]}[|\ell_1 - S_1| + |\ell_2 - S_2| + |\ell_3 - S_4| = v - 8]$$

$$= \sum_{x \in [0,7]} \sum_{y \in [0,7]} \mathcal{P}(w = x)\mathcal{P}(w = y)\mathcal{P}(w = v - 8 - x - y)$$

According to the description of algorithm $\mathcal{A}$, the probability of success of one iteration for finding S_0 depends on the value of $v - 8$.

Table 5. Probability of the favorable case

S_0	-3	-2	-1	0	1	2
Success Probability (%)	P_{-1}	$P_{-1} + P_{-2}$	$P_{-1} + P_{-2} + P_{-3}$	$P_{-1} + P_{-2} + P_{-3}$	$P_{-1} + P_{-2}$	P_{-1}

Table 1 is finally computed from Table 5 assuming that S_1, S_2, S_3 follow a binomial distribution ψ_4.

D Proof of Corollary 1

Let us define $\mathcal{A}'$ as algorithm $\mathcal{A}$ iterated until it finds a favorable case. This algorithm is probabilitic, and when it finishes, the last iteration corresponds to a favorable case. We still focus on targetting the secret S_0. Actually, one can lower down the number of queries of Proposition 2 with several techniques.

– If the first and the eighth queries are not (+), the iteration is restarted because a favorable case in impossible.
– If τ_1 and τ_2 are already found, the queries can be stopped.

The expected number of queries for finding S_0 can be computed using the success rate of $\mathcal{A}$ which depends on the distribution of (S_j). There are several cases depending on the value $v - 8$ (see Table 6)

– If $v - 8 \geq 0$, eight queries are used. The probability of this event is $\sum_{i=0}^{16} P_i$
– If $v - 8 \leq -4$, only two queries at the border are needed to be sure that the result will not be found. The probability of this event is $\sum_{i=-8}^{-4} P_i$
– For the other cases, the number of queries depends on the values of S_0 and can be derived with algorithm $\mathcal{A}$ description.

Table 6. Number of queries needed depending on the value of $v - 8$

S_0	-3	-2	-1	0	1	2
$v - 8 = -1$ (probability P_{-1})	3	4	5	6	7	8
$v - 8 = -2$ (probability P_{-2})	2	5	6	7	8	2
$v - 8 = -3$ (probability P_{-3})	2	2	7	8	2	2

Assuming that S_1, S_2, S_3 follow a binomial distribution ψ_4, the number of queries are stored in Table 7.

Table 7. Expected number of queries of $\mathcal{A}'$ to find S_0 when $(S_j)_{1 \leq j \leq 3}$ following ψ_4 distribution

S_0	-3	-2	-1	0	1	2
Expected number of queries for one iteration	4.6	5.1	$6,0$	6.4	5.9	5.3
Expected number of iterations	7.1	3.7	2.6	2.6	3.7	7.1
Expected number of queries before termination	33	19	16	17	22	38