Analysis of Two Countermeasures against the Signal Leakage Attack

No Author Given

No Institute Given

Abstract. In 2017, a practical attack, referred to as the signal leakage attack, against reconciliation-based RLWE key exchange protocols was proposed. In particular, this attack can recover a long-term private key if a key pair is reused.
Directly motivated by this attack, recently, Ding et al. proposed two countermeasures against the attack. One is the RLWE key exchange protocol with reusable keys (KERK), which is included in the Ding Key Exchange, a NIST submission. The idea for this construction is using zero knowledge proof. The other is the practical randomized RLWE-based key exchange (PRKE) (TOC'18), which mixes more randomization.
We found that the two countermeasures above can effectively prevent malicious Alice from recovering the private key of Bob when keys are reused. However, both countermeasures don't consider the case where malicious Bob tries to recover the private key of Alice. In particular, malicious Bob can recover the private key of Alice by carefully choosing what he sends and observing whether shared keys match. By analyzing the complexities of these attacks, the results show these attacks are practical and effective.
Notice that the key to carry out these attacks is that malicious Bob chooses a RLWE sample with the special structure as his public key. Therefore, other RLWE-based schemes, including KEM (or key exchange) and PKE, are also vulnerable to these attacks. In response to these attacks, we propose a mechanism where one party can construct a new "public key" of the other party, and in order to illustrate the mechanism, we give an improved KERK.

Keywords: RLWE · key exchange · post-quantum · key reuse · analysis · active attacks.

1 Introduction

Key exchange is an important cryptography primitive. It allows two or more parties to agree on the same key, which is used in symmetric ciphers to encrypt and decrypt traffic data. Since the groundbreaking work of Diffie-Hellman key exchange [1], various key exchange protocols following this idea have been designed, implemented and deployed in real-world applications.

In 1994, Shor proposed a quantum algorithm in [2], which can break most current public key cryptosystems based on integer factoring problem, discrete logarithm problem etc. Cryptographic algorithms designed based on these hard problems are no longer secure when large quantum computers are implemented. Fortunately, there are several approaches that can defeat such attacks, including lattice-based, multivariate-based, hash-based, code-based, and others. In particular, lattice-based cryptography is regarded as a very promising one because construction based on lattice problems are extremely hard to solve, even against quantum computers. It also enjoys strong provable security and very high efficiency. An important line of lattice-based cryptography is constructions based on the Ring-Learning with Error(RLWE) problem [4].

RLWE-based key exchange protocol was introduced in 2012 (denoted as DING12)[5], which gives RLWE variant of Diffie-Hellman key exchange. Following the idea of this work, various RLWE-based key exchange protocols have been designed and implemented, including [6]-[12]. One common approach to achieve RLWE-based key exchange is error reconciliation. In particular, DING12, PKT14 [6], NewHope [9] and HILA5 [26] belong to reconciliation-based RLWE key exchange protocol.

Recently, a practical attack, referred to as the signal leakage attack, against reconciliation-based RLWE key exchange protocols was proposed [13]. This attack can recover a long-term private key if a key pair is reused. It is known that in the real world, key reuse is commonly adopted in applications like the Transport Layer Security (TLS) protocol to improve performance. In TLS v1.2, the resumption mode allows key reuse, and this reduces online computations significantly. Another instance of key reuse appears in the Internet Key Exchange (IKE). Currently with classical DH, some implementations of IKE do reuse the keys for improved computational efficiency and latency. If RLWE-based key exchange protocols that are vulnerable to key reuse attack are adopted in TLS and IKE with reused keys, the security of communication is compromised.

Directly motivated by this attack, Ding et al. constructed two countermeasures against the attack. One is called the RLWE key exchange protocol with reusable keys (KERK), which was proposed in the Ding Key Exchange, a NIST submission. The idea for this construction is using zero knowledge proof. In particular, the key exchange is based on the authentication protocol proposed in [27], where they firstly designed a zero knowledge-based authentication protocol. It is a novel application of the signal function used for reconciliation in key exchange to derive a secure authentication protocol. The other is the practical randomized RLWE-based key exchange (PRKE)[15]. The protocol extends DING12 and incorporates an additional ephemeral public error term into key exchange materials so that the practical signal leakage attack does not work. The protocol has two modes: regular mode and key reuse mode. Regular mode is for the fresh key exchange scenario that two parties do not have a prior key negotiation or they do not want to reuse keys. Key reuse mode is a simplified version of regular mode that is for the key reuse case.

1.1 Related Work

In 2015, Kirkwood et al. from National Security Agency(NSA) revealed an issue in reconciliation-based key exchange protocols [16]. They suggested that if a public and private key pairs is reused, current reconciliation-based LWE-based and RLWE-based key exchange protocols may suffer from an attack that can reveal a private key with multiple key exchange executions.

In 2016, Fluhrer [17] gave cryptanalysis on RLWE-based key exchange protocols . This work gives the basic structure of the attack and shows that RLWE-based key exchange can be broken when a key pair is reused. In 2017, Ding et al. [13] presented a signal leakage attack with two extensions. They also showed that attacks are indeed practical with an actual attack example on the RLWE-based key exchange. In 2017, Bernstein et al. [18] demonstrated a key-recovery attack on HILA5 using an active attack on reused keys. The attack applies to the HILA5 key-encapsulation mechanism (KEM), and also to the public-key encryption mechanism (PKE) obtained by NIST's procedure for combining the KEM with authenticated encryption.

In December 2017, NIST announced 69 post-quantum cryptosystems entered the Round 1. As one of the submissions, Ding Key Exchange includes a KERE, which can achieve secure key reuse.In 2018, Ding et al. [19] described a new attack on Ding's one pass case without relying on the signal function output but using only the information of whether the final key of both parties agree. Moreover, they showed that the previous signal leakage attack [13] can be made more efficient with fewer communications and how it can be extended to Peikert's key exchange [6]. In 2018, Ding et al. [15] constructed a new randomized RLWE-based key exchange protocol against the signal leakage attack. In particular, they incorporate an additional ephemeral public error term into key exchange materials.

Recently, Liu et al. [28] describe a new key reuse attack against the NewHope key exchange protocol proposed by Alkim et al. in 2016. They give a detailed analysis of the signal function of the NewHope and describe a new key recovering technique based on the special property of NewHope's signal. In addition, D'Anvers et al. [29] investigate the impact of decryption failures on the chosen-ciphertext security of (Ring/Module)-Learning With Errors and (Ring/Module)-Learning with Rounding based primitives. In particular, they introduce a technique to increase the failure rate of these schemes called failure boosting. They also examine the amount of information that an adversary can derive from failing ciphertexts. Chen Li [30] also takes advantage of the information leakage from the decapsulation feedback and provide an efficient key recovery attack on the Streamlined NTRU Prime.

1.2 Our Contributions

In this work, we analyze two countermeasures against the signal leakage attack: KERK and PRKE. In particular, we develop two attacks on these two countermeasures. In these attacks, malicious Bob can recover the private key of Alice by carefully choosing what he sends to Alice and observing whether the final shared keys match or not. Further, we give analyses of the complexities of these attacks, which shows these attacks are practical and effective.

Notice that the key to carry out these attack is malicious Bob chooses a RLWE sample with the special structure as his public key. Therefore, these attacks also work for other RLWE-based schemes, including KEM (or key exchange) and PKE. In particular, we propose a mechanism which can resist these attacks, and give an improved KERK to illustrate the mechanism.

2 Organization

In section 3, we introduce some notations, background on RLWE, the error reconciliation mechanism from DING12 and RLWE key exchange. In Section 4 and Section 5, we revisit the signal leakage attack on DING12 and two countermeasures, which consist of KERK and PRKE. In section 6, we analyze these two countermeasures and give corresponding attacks and the complexities of attacks. Finally, we discuss the scope of these attacks, propose a mechanism which can resist these attacks and give an improved KERK in Section 7. In Section 8, we make a conclusion.

3 Preliminaries

Notation Let n be an integer and a power of 2. Define $f(x) = x^n + 1$ and consider the ring $R := \mathbb{Z}[x]/\langle f(x)\rangle$. For any positive integer q, we define $R_q = R/qR \cong \mathbb{Z}_q[x]/\langle f(x)\rangle$ analogously, where the ring of polynomials over $\mathbb{Z}$ (respectively $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$) we denote by $\mathbb{Z}[x]$ (respectively $\mathbb{Z}_q[x]$)). For any polynomial $p \in R$ (or R_q), let $p[i]$ denote the i-th coefficient of p.

Discrete Gaussian Distribution For any positive real $\sigma \in \mathbb{R}$, and vector $c \in \mathbb{R}^n$, the continuous Gaussian distribution over $\mathbb{R}^n$ with standard deviation σ centered at c is defined by the probability function $\rho_{\sigma,c}(x) = (\frac{1}{\sqrt{2\pi\sigma^2}})^n exp(\frac{-\|x-c\|^2}{2\sigma^2})$. For integer vectors $c \in \mathbb{R}^n$, let $\rho_{\sigma,c}(\mathbb{Z}^n) = \sum_{x\in\mathbb{Z}^n} \rho_{\sigma,c}(x)$. Then, we define the discrete Gaussian distribution over $\mathbb{Z}^n$ as $D_{\mathbb{Z}^n,\sigma,c}(x) = \frac{\rho_{\sigma,c}(x)}{\rho_{\sigma,c}(\mathbb{Z}^n)}$, where $x \in \mathbb{Z}^n$. The subscripts σ and c are taken to be 1 and 0 (respectively) when omitted.

Ring Learning with Errors(RLWE) A Lattice $L(b_1, ..., b_n) = \{\sum_{i=1}^n x_i b_i | x_i \in \mathbb{Z}\}$ is formed by integer linear combinations of n linearly independent vectors $b_1, ..., b_n \in \mathbb{R}^n$ called the Lattice Basis. In 1996, Ajtais seminal result[21] heralded the use of lattices for constructing cryptographic systems, with the security based on hardness of problems such as the Shortest Vector Problem (SVP) and Closest Vector Problem (CVP). The Learning with Errors (LWE) problem introduced by Oded Regev in 2005[4] is a generalization of the parity-learning problem. The reduction from solving hard problems in lattices in the worst case to solving LWE in the average case provides strong security guarantees for L-WE based cryptosystems, yet it is not efficient enough for practical applications due to its large key sizes of $O(n^2)$. Ring-Learning with Errors(RLWE) is the version of LWE in the ring setting, that overcomes the efficiency disadvantages of LWE. Similar to LWE, there is a quantum reduction from solving worst case lattice problems in ideal lattices to solving the RLWE problem in average case. The search version of RLWE is to find a secret s in R_q given $(a, as + e)$ for polynomial number of samples, where a is sampled uniform from R_q and e is sampled according to the error distribution $D_{\mathbb{Z}^n,\sigma}$. An equivalent problem of the search version is the decision version which is commonly used for security proof of cryptographic algorithms based on RLWE. Let $A_{s,D_{\mathbb{Z}^n,\sigma}}$ denote the distribution of the pair $(a, as + e)$, where a, s is sampled uniformly from R_q and e is sampled according to the error distribution $D_{\mathbb{Z}^n,\sigma}$. The decision version of the RLWE problem is to distinguish $A_{s,D_{\mathbb{Z}^n,\sigma}}$ from the uniform distribution on $R_q \times R_q$ with polynomial number of samples. The *normal form* [22, 23] of the RLWE problem is by modifying the above definition by choosing s from the error distribution $D_{\mathbb{Z}^n,\sigma}$ rather than uniformly. It has been proven that the ring-LWE assumption still holds even with this variant [24, 4].

The Error Reconciliation Mechanism from DING12 The error reconciliation mechanism in DING12 mainly consists of a signal function and a robust extractor. More specifically, in the protocol, two parties will compute two very close values in $\mathbb{Z}_q$. In order to agree on a common value, one party additionally sends a signal of his value. Then both parties compute their shared keys using the robust extractor.

Signal Function For the Key Exchange from RLWE presented in [5], the signal function is required for the two parties in the key exchange to derive a final shared key. The signal function is usually sent by the responding party to the initiator of the key exchange, which gives additional information about whether the responder's key computed lies in a specific region.

For prime $q > 2$, hint functions $\sigma_0(x), \sigma_1(x)$ from $\mathbb{Z}_q$ to $\{0, 1\}$ are defined as:

$$\sigma_0(x) = \begin{cases} 0, & x \in [-\lfloor \frac{q}{4} \rfloor, \lfloor \frac{q}{4} \rfloor] \\ 1, & otherwise \end{cases}$$

$$\sigma_1(x) = \begin{cases} 0, & x \in [-\lfloor \frac{q}{4} \rfloor + 1, \lfloor \frac{q}{4} \rfloor + 1] \\ 1, & otherwise \end{cases}$$

Signal function Cha() is defined as: For any $y \in \mathbb{Z}_q$, $\text{Cha}(y) = \sigma_b(y)$, where $b \overset{\$}{\leftarrow} \{0, 1\}$.

Robust Extractor Informally, a robust extractor enables two parties to extract an identical information from two close elements with some additional hint. The robust extractor is defined as:

$$\text{Mod}_2(x, w) = (x + w \cdot \frac{q-1}{2} \text{mod } q) \text{mod } 2$$

$\text{Mod}_2()$ is a robust exactor on $\mathbb{Z}_q$ with error tolerance δ with respect to signal function, if the following holds:

– The deterministic algorithm $\text{Mod}_2()$ takes input $x \in \mathbb{Z}_q$ and signal w, output $k = \text{Mod}_2(x, w) \in \{0, 1\}$.

– $\text{Cha}()$ takes an input $y \in \mathbb{Z}_p$ and outputs signal $w \leftarrow \text{Cha}(y) \in \{0, 1\}$.

For any $x, y \in \mathbb{Z}_q$ such that $x - y$ is even and $\|x - y\| \leq \delta$, then it holds that $\text{Mod}_2(x, w) = \text{Mod}_2(y, w)$, where $w \leftarrow \text{Cha}(y)$.

Let $q > 8$ be an odd integer, the function $\text{Mod}_2()$ defined above is a robust extractor with respect to Mod with error tolerance $\frac{q}{4} - 2$.

For any odd $q > 2$, if x is uniformly random in $\mathbb{Z}_q$, then $\text{Mod}_2(x)$ is uniformly random conditioned on w, where $w \leftarrow \text{Cha}(x)$.

RLWE-Based Key Exchange Protocol In 2012, Ding et al. introduced the the first RLWE-based key exchange protocols [5]. This is the first work that gives practical and provable secure RLWE-based key exchange protocols. DING12 can be regarded as RLWE variant of the classical Diffie-Hellman key exchange protocol. They introduced robust extractor, allowing both sides to reconcile errors between approximately equal values to agree on the same key using an additional signal value. DING12 is proven secure under attacks from passive probabilistic polynomial time (PPT) adversaries. In 2014, Peikert presented a slightly modified reconciliation mechanism (denoted as PKT14) [6]. In 2015, Bos et al. instantiated PKT14 with practical parameter choice, implementation and integration into OpenSSL as a post-quantum TLS ciphersuite (denoted as BCNS15) [7]. In 2015, Zhang et al. introduced a RLWE-based AKE protocol (denoted as AKE15) [8]. AKE15 is a RLWE variant of HMQV and it is proven secure under the Bellare-Rogaway model. In 2016, Alkim et al. improved the efficiency and security of BCNS15 with a new error reconciliation mechanism, detailed security analysis, removal of fixed parameter a and optimized implementation (denoted as NewHope) [9]. In late 2016, a variant of NewHope without using the error reconciliation mechanism was introduced (denoted as NewHope-Simple) [11]. This work gives an encryption-based approach to realize key exchange. In 2017, Ding et al. introduced two proven secure password-based RLWE key exchange protocols that are RLWE analogue of PAK and PPK [12]. Gao et al. presented a much optimized implementation of [12] and integration into TLS as post-quantum TLS ciphersuite [25]. Gao et al. introduced a RLWE variant of Secure Remote Password(SRP) protocol [20], which is an augmented PAKE protocol.

Reconciliation-based Key Exchange It is an approach to construct a passively secure lattice-based Key Exchange scheme [11]. In particular, we take NewHope for example. As shown in the Table 1, in Reconciliation-based Key Exchange, the final shared key sk_B (sk_A) is derived from k_B (k_A) and w, where w is also derived from k_B too.

KEM.Setup(): a $\overset{\$}{\leftarrow} R_q$		
Alice		Bob
KEM.Gen(a):		KEM.Encaps(a, b):
$s_A, e_A \overset{\$}{\leftarrow} D_{\mathbb{Z}^n,\sigma}$		$s_B, e_B, e'_B \overset{\$}{\leftarrow} D_{\mathbb{Z}^n,\sigma}$
$p_A \leftarrow a s_A + e_A$	$\xrightarrow{p_A}$	$p_B \leftarrow a s_B + e_B$
		$k_B \leftarrow p_A s_B + e'_B$
KEM.Decaps($s_A, (p_B, w)$)	$\xleftarrow{p_B, w}$	$w \overset{\$}{\leftarrow}$ HelpRec(k_B)
$k_A \leftarrow p_B s_A$		$sk_B \leftarrow$ Rec(k_B, w)
$sk_A \leftarrow$ Rec(k_A, w)		

Table 1. Reconciliation-based key exchange

4 Revisit the Signal Leakage Attack on DING12

Alice		**Bob**
Public key: $p_A = a s_A + 2e_A \in R_q$		Public key: $p_B = a s_B + 2e_B \in R_q$
Private key: $s_A \in R_q$		Private key: $s_B \in R_q$
where $s_A, e_A \leftarrow D_{\mathbb{Z}^n,\sigma}$		where $s_B, e_B \leftarrow D_{\mathbb{Z}^n,\sigma}$
	$\xrightarrow{p_A}$	
		$k_B = p_A s_B + 2e'_B$
		where $e'_B \leftarrow D_{\mathbb{Z}^n,\sigma}$
		$w = \mathrm{Cha}(k_B) \in \{0,1\}^n$
	$\xleftarrow{p_B, w}$	
$k_i = p_B s_A + 2e'_A$		$\sigma_B = \mathrm{Mod}_2(k_B, w) \in \{0,1\}^n$
where $e'_A \leftarrow D_{\mathbb{Z}^n,\sigma}$		$sk_B = \mathrm{SHA2} - 256(\sigma_B)$
$\sigma_A = \mathrm{Mod}_2(k_A, w) \in \{0,1\}^n$		
$sk_A = \mathrm{SHA2} - 256(\sigma_A)$		

Table 2. DING12 key exchange

In this section, we briefly recall the signal leakage attack [13] on DING12 (Table 2). If Alice is malicious, this attack can recover Bob's private key within multiple queries if key pair (s_B, p_B) is reused.

– In step 1 of the attack, malicious Alice chooses secret key $s_A = 0$ and error term $e_A = 1$, therefore public key $p_A = a s_A + k e_A = k$ and signal value [1]

$$w[i] = \mathrm{Cha}(k_B[i]) = \mathrm{Cha}(p_A s_B[i]) = \mathrm{Cha}(k s_B[i]). \tag{1}$$

According to the definition of signal function Cha, signal value $w[i]$ flips when $k s_B[i]$ enters or exits inner region $[-q/4, q/4]$. When k loops from 0

[1] First consider a simple case without adding $2e'_B$, where $k_B = p_A s_B$.

to $q-1$, signal value $w[i]$ flips exactly $2s_B[i]$ times. By communicating with Bob with k looping from 0 to $q-1$, Alice can get the value of $s_B[i]$ based on the number of times that signal $w[i]$ changes, therefore, Alice can find the absolute value of $2s_B[i]$ without knowing the sign.
- In step 2, Alice sends $(1+x)p_A$ to Bob. By looping k from 0 to $q-1$ once again, Alice can get the absolute value of $s_B[0]-s_B[n-1], s_B[1]+s_B[2], s_B[2]+s_B[3], ..., s_B[n-2]+s_B[n-1]$ without knowing the actual signs.
- In steps 3 and 4, Alice can reveal the relationship between the signs of neighboring coefficients using all information from previous steps.
- In step 5, Alice verifies his guess on signs by checking the distribution of $p_B - as_B$. If it follows the distribution of e_B (normally is discrete Gaussian distribution), then the guess is correct and the secret key is successfully recovered. If not, then flips the signs and obtain the correct s_B.

The above attack does not take account of additional errors $2e'_B$, but this is not an issue. When adding $2e'_B$, the attack steps remain the same. Although the error term causes some fluctuations, it is relatively small and can be ignored. The private key can also be recovered successfully.

More generally, the attack still works if p_A is chosen with $s_A \neq 0$. In this case,

$$w[i] = \mathrm{Cha}(k_B[i]) = \mathrm{Cha}(p_A s_B[i]) = \mathrm{Cha}(as_A s_B[i] + ks_B[i]). \tag{2}$$

The value $as_A s_B[i]$ is constant when looping over k value. So, Alice still can get the value of $s_B[i]$ by looking at the number of times that signal $w[i]$ changes. Alice only needs to sample s_A according to the distribution defined in key exchange protocol and the rest of the attack remains the same. This attack can be done within $2q$ online queries with Bob.

5 Revisit Two Countermeasures against the Signal Leakage Attack

In this section, we revisit two countermeasures against the signal leakage attack. The first one was abbreviated as KERK and proposed in the Ding Key Exchange, a NIST submission. The idea for this construction is using zero knowledge proof [27]. The other is PRKE in [15], which mixes more randomization.

5.1 KERK

Firstly, we revisit KERK (Table 3). The construction of the key exchange is based on the zero knowledge-based authentication protocol proposed in [27]. It can prevent malicious Alice from recovering the private key of Bob when keys are reused. In particular, if malicious Alice chooses secret key $s_A = 0$ and error term $e_A = 1$ such that public key $p_A = as_A + ke_A = k$, then

$$\begin{aligned} w = \mathrm{Cha}(k_B) &= \mathrm{Cha}(\overline{p_A} s_B + 2e''_B) \\ &= \mathrm{Cha}((a \cdot \mathrm{Samp}(\mathrm{H}(p_A)) + 2e'_B + p_A)s_B + 2e''_B) \\ &= \mathrm{Cha}((a \cdot \mathrm{Samp}(\mathrm{H}(k)) + 2e'_B + k)s_B + 2e''_B), \end{aligned} \tag{3}$$

Alice		**Bob**
Public key: $p_A = as_A + 2e_A \in R_q$		Public key: $p_B = as_B + 2e_B \in R_q$
Private key: $s_A \in R_q$		Private key: $s_B \in R_q$
where $s_A, e_A \leftarrow D_{\mathbb{Z}^n,\sigma}$		where $s_B, e_B \leftarrow D_{\mathbb{Z}^n,\sigma}$
	$\xrightarrow{p_A}$	
		Sample $e'_B, e''_B \leftarrow D_{\mathbb{Z}^n,\sigma}$
		Samp(H(p_A))$\leftarrow D_{\mathbb{Z}^n,\sigma}$
		$\overline{p_A} = a\cdot$ Samp(H(p_A)) $+ 2e'_B + p_A$
Sample $e'_A \leftarrow D_{\mathbb{Z}^n,\sigma}$		$k_B = \overline{p_A} s_B + 2e''_B$
Samp(H(p_A))$\leftarrow D_{\mathbb{Z}^n,\sigma}$		$w = \text{Cha}(k_B) \in \{0,1\}^n$
	$\xleftarrow{p_B, w}$	
$k_A = p_B \cdot (s_A +$ Samp(H(p_A))) $+ 2e'_A$		$sk_B = \text{Mod}_2(k_B, w) \in \{0,1\}^n$
$sk_A = \text{Mod}_2(k_A, w) \in \{0,1\}^n$		

Table 3. KERK

where Samp() is a function which generates polynomial in R_p using output of H according to distribution $D_{\mathbb{Z}^n,\sigma}$, Samp(H(k)) is indistinguishable from uniform and e'_B is chosen by Bob. Therefore, $a \cdot$ Samp(H(k)) $+ 2e'_B$ can be regarded as a RLWE sample and is indistinguishable from uniform. By looping k from 0 to $q-1$, Alice can't get the value of $s_B[i]$ based on the number of times that signal $w[i]$ changes.

5.2 PRKE

In this section, we revisit PRKE in [15]. As shown in Table 3 and Table 4, it has two modes: regular mode and key reuse mode, which share the same structure. Key reuse mode is a simplified version of regular mode, which reduces computation and communication costs with reused keys.

Alice		**Bob**
Reused public key: $p_A = as_A + 2e_A$		Reused public key: $p_B = as_B + 2e_B$
Reused private key: s_A		Reused private key: s_B
	$\xrightarrow{Session\ ID}$	
		$k_B = (p_A s_B + 2e_p) \cdot e_p + 2e'_B$
		where $e_p, e'_B \leftarrow D_{\mathbb{Z}^n,\sigma}$
		$w = \text{Cha}(k_B) \in \{0,1\}^n$
	$\xleftarrow{w, e_p}$	
$k_A = (p_B s_A + 2e_p) \cdot e_p + 2e'_A$		$\sigma_B = \text{Mod}_2(k_B, w) \in \{0,1\}^n$
where $e'_A \leftarrow D_{\mathbb{Z}^n,\sigma}$		$sk_B = \text{SHA2} - 256(\sigma_B)$
$\sigma_A = \text{Mod}_2(k_A, w) \in \{0,1\}^n$		
$sk_A = \text{SHA2} - 256(\sigma_A)$		

Table 4. Key reuse mode

Alice		**Bob**
Public key: $p_A = as_A + 2e_A \in R_q$		Public key: $p_B = as_B + 2e_B \in R_q$
Private key: $s_A \in R_q$		Private key: $s_B \in R_q$
where $s_A, e_A \leftarrow D_{\mathbb{Z}^n,\sigma}$		where $s_B, e_B \leftarrow D_{\mathbb{Z}^n,\sigma}$
	$\xrightarrow{p_A}$	
		$k_B = (p_A s_B + 2e_p) \cdot e_p + 2e'_B$
		where $e_p, e'_B \leftarrow D_{\mathbb{Z}^n,\sigma}$
		$w = \text{Cha}(k_B) \in \{0,1\}^n$
	$\xleftarrow{p_B, w, e_p}$	
$k_i = (p_B s_A + 2e_p) \cdot e_p + 2e'_A$		$\sigma_B = \text{Mod}_2(k_B, w) \in \{0,1\}^n$
where $e'_A \leftarrow D_{\mathbb{Z}^n,\sigma}$		$sk_B = \text{SHA2} - 256(\sigma_B)$
$\sigma_A = \text{Mod}_2(k_A, w) \in \{0,1\}^n$		
$sk_A = \text{SHA2} - 256(\sigma_A)$		

Table 5. Regular mode

Regular mode is designed for common key exchange between two parties without reused keys. In this mode, two parties exchange their public keys, signal w and a one-time public error e_p to agree on the same session keys. There are several preferable scenarios for this mode: (1)The two parties have never communicated before, and Alice dose not share Bob's public key. (2)The two parties performed key exchange sometime before, but it has been too long since their last communication, therefore they need to generate new public and private key pairs and start key exchange.

Key reuse mode is designed for both parties wanting to reuse a key pair and it is directly derived from the regular mode. The main motivation for key reuse is for better performance because generating the key pair is somewhat expensive. Alice needs to send a 256-bit session id to Bob, notifying Bob that he wants to reuse the key from a previous session. Bob retrieves keys of Alice from the database, generates a fresh public error e_p and returns signal w and e_p to Alice to generate a fresh session key.

The key exchange in key reuse mode can prevent malicious Alice from recovering the private key of Bob when keys are reused. In particular, if malicious Alice chooses secret key $s_A = 0$ and error term $e_A = 1$ such that public key $p_A = as_A + ke_A = k$, then

$$\begin{aligned} w = \text{Cha}(k_B) &= \text{Cha}((p_A s_B + 2e_p) \cdot e_p + 2e'_B) \\ &= \text{Cha}((ks_B + 2e_p) \cdot e_p + 2e'_B) \\ &= \text{Cha}(ks_B e_p + 2e_p^2 + 2e'_B), \end{aligned} \tag{4}$$

where e_p is chosen by Bob to mix more randomization. By looping k from 0 to $q-1$, Alice can't get the value of $s_B[i]$ based on the number of times that signal

$w[i]$ changes. In fact, the approach of mixing more randomization is a common and practical solution, for example, using public random one-time initialization vector (IV) in encryption, nonce in various cryptography protocols, long and random salt in password-based key derivation function.

6 Analysis of Two Countermeasures

The two countermeasures above can effectively prevent malicious Alice from recovering the private key of Bob when keys are reused. However, both countermeasures don't consider the case where malicious Bob tries to recover the private key of Alice. In particular, malicious Bob can recover the private key s_A of Alice by carefully choosing what he sends and observing whether shared keys match.

6.1 Attack on the KERK

Firstly, we consider the KERK, where malicious Bob deliberately chooses p_B, and w with special structure to recover the private key of Alice. In the attack, Bob is an active adversary intending to recover the private key s_A of the Alice. In every communication, Alice executes the key exchange honestly and Bob carefully chooses p_B and w.

Alice		**Bob**
Public key: $p_A = as_A + 2e_A \in R_q$		Public key: $p_B = as_B + 2e_B \in R_q$
Private key: $s_A \in R_q$		Private key: $s_B \in R_q$
where $s_A, e_A \leftarrow D_{\mathbb{Z}^n,\sigma}$		where $s_B, e_B \leftarrow D_{\mathbb{Z}^n,\sigma}$
	$\xrightarrow{p_A}$	
		Sample $e'_B, e''_B \leftarrow D_{\mathbb{Z}^n,\sigma}$
		Samp(H(p_A))$\leftarrow D_{\mathbb{Z}^n,\sigma}$
		$\overline{p_A} = a \cdot \overline{s_A} + 2\overline{e'_A}$
Sample $e'_A \leftarrow D_{\mathbb{Z}^n,\sigma}$		$k_B = \overline{p_A} s_B + 2e''_B$
Samp(H(p_A))$\leftarrow D_{\mathbb{Z}^n,\sigma}$		$w = \mathrm{Sig}(k_B) \in \{0,1\}^n$
	$\xleftarrow{p_B, w}$	
$k_A = p_B \cdot \overline{s_A} + 2e'_A$		$sk_B = \mathrm{Mod}_2(k_B, w) \in \{0,1\}^n$
$sk_A = \mathrm{Mod}_2(k_A, w) \in \{0,1\}^n$		

Table 6. new expression of Table 3

For convenience, we denote $s_A + \mathrm{Samp(H}(p_A))$ by $\overline{s_A}$, then in the key exchange (Table 6), k_A and $\overline{p_A}$ can be expressed as follow:

$$k_A = p_B \cdot (s_A + \mathrm{Samp(H}(p_A))) + 2e'_A = p_B \cdot \overline{s_A} + 2e'_A. \qquad (5)$$

$$\begin{aligned}\overline{p_A} &= a \cdot \mathrm{Samp}(\mathrm{H}(p_A)) + 2e'_B + p_A \\ &= a \cdot \mathrm{Samp}(\mathrm{H}(p_A)) + 2e'_B + as_A + 2e_A \\ &= a \cdot (\mathrm{Samp}(\mathrm{H}(p_A)) + s_A) + 2e'_B + 2e_A \\ &= a \cdot \overline{s_A} + 2(e'_B + e_A) \\ &= a \cdot \overline{s_A} + 2\overline{e'_A}. \end{aligned} \tag{6}$$

where we denote $e'_B + e_A$ by $\overline{e'_A}$, and Table 3 can be expressed as Table 6.

In particular, in order to recover s_A, Bob firstly recovers $\overline{s_A}$. Here we introduce two methods to recover $\overline{s_A}$.

Method 1: We first adopt the method proposed in [19], where in order to recover the i-th coefficient $\overline{s_A}[i]$, Bob chooses

- s_B to be 0
- e_B such that $e_B[t] = 0$ for all $t = 0, ..., n-1$ except $t = n-1-i, n-1-j$ and $e_B[n-1-i] = 1, e_B[n-1-j] = k$, namely, $e_B = e_B[n-1-i]x^{n-1-i} + e_B[n-1-j]x^{n-1-j}$
- j such that $\overline{s_A}[j] = 1$ [2]
- k is a small integer

He then performs the protocol honestly, except that he deliberately flips bit $w[n-1]$ to be 1.

In this way [3],

$$\begin{aligned} k_A = p_B\overline{s_A} = e_B\overline{s_A} & \\ &= (2e_B[n-1-i]x^{n-1-i} + 2e_B[n-1-j]x^{n-1-j})\overline{s_A} \\ &= (2x^{n-1-i} + 2kx^{n-1-j})\overline{s_A}. \end{aligned} \tag{7}$$

and $k_A[n-1]x^{n-1}$ only depends on two terms of $\overline{s_A}$, and they are $\overline{s_A}[i]x^i$ and $\overline{s_A}[j]x^j$, which results in

$$k_A[n-1] = 2\overline{s_A}[i] + 2k\overline{s_A}[j] = 2\overline{s_A}[i] + 2k. \tag{8}$$

$$\begin{aligned} sk_A[n-1] = \mathrm{Mod}_2(k_A[n-1], w) & \\ &= k_A[n-1] + w[n-1] \cdot \frac{q-1}{2} \bmod q \bmod 2 \\ &= 2\overline{s_A}[i] + 2k + \frac{q-1}{2} \bmod q \bmod 2. \end{aligned} \tag{9}$$

[2] [19] proposed a method of hypothesis verification to choose j such that $s_A[j] = \pm 1$ and this method can be applied to other values except ± 1. Here we use the same method and take $\overline{s_A}[j] = 1$ for example. In fact, it is possible that no coefficient of $\overline{s_A}$ is equal to one and our analysis is applicable other values as well.

[3] Similarly, first consider a simple case without adding $2e'_A$, where $k_A = p_B\overline{s_A}$.

Next, we show that the k value reveals the value of $\overline{s_A}[i]$ when there is a change in the value of $sk_A[n-1]$. Notice that the terms $2\overline{s_A}[i]+2k$ is even and also from the usual choice of parameters for RLWE such that $q = 1 \bmod 2n$, we have $\frac{q-1}{2}$ to be even.

– First consider the sign of $\overline{s_A}[i]$: Alice chooses $e_B[n-1-j] = k = 0$, which will result in $k_A[n-1] = 2\overline{s_A}[i]$ and.

$$sk_A[n-1] = 2\overline{s_A}[i] + \frac{q-1}{2} \bmod q \bmod 2. \tag{10}$$

If $\overline{s_A}[i]$ is negative, $2\overline{s_A}[i] + \frac{q-1}{2} \bmod q$ is even and $sk_A[n-1] = 0$; If $\overline{s_A}[i]$ is positive, $2\overline{s_A}[i] + \frac{q-1}{2} \bmod q$ is odd since the addition of $\frac{q-1}{2}$ to a positive value changes its parity by the representation of $\mathbb{Z}_q$ to be $\{-\frac{q-1}{2} ... \frac{q-1}{2}\}$, thus $sk_A[n-1] = 1$. In addition, Alice can choose $e_B[n-1-i] = -1$ to identify whether $\overline{s_A}[i]$ is 0, which results in $k_A[n-1] = -2\overline{s_A}[i]$ and $sk_A[n-1] = -2\overline{s_A}[i] + \frac{q-1}{2} \bmod q \bmod 2$.

– Next consider that case where $\overline{s_A}[i]$ is negative: notice that

$$sk_A[n-1] = 2\overline{s_A}[i] + 2k + \frac{q-1}{2} \bmod q \bmod 2. \tag{11}$$

and when k is small, $sk_A[n-1] = 0$ as long as $2\overline{s_A}[i] + 2k$ is negative. As k increases, $2\overline{s_A}[i] + 2k$ changes from negative to positive. As this happens, we have $sk_A[n-1] = 1$. Thus, the k value reveals the value of $\overline{s_A}[i]$ when there is a change in the value of $sk_A[n-1]$.

– Then when $\overline{s_A}[i]$ is positive, the situation is the same except that Bob chooses $e_B[n-1-j] = -k$ such that $k_A[n-1] = 2\overline{s_A}[i] - 2k$, which results in

$$sk_A[n-1] = 2\overline{s_A}[i] - 2k + \frac{q-1}{2} \bmod q \bmod 2. \tag{12}$$

In this case, when k is small, $sk_A[n-1] = 1$ as long as $2\overline{s_A}[i] - 2k$ is positive. As k increases, $2\overline{s_A}[i] - 2k$ changes from positive to negative. As this happens, we have $sk_A[n-1] = 0$.

– Therefore, looking for the k value when $sk_A[n-1]$ changes from 0 to 1 reveals the exact value of a negative $\overline{s_A}[i]$ and a change from 1 to 0 reveals the value of a positive $\overline{s_A}[i]$.

As Bob performs the protocol mostly honestly (and both s_B and e_B qualify as small vectors until k remains small), Bob can compute the value sk_A, except for index $n-1$, for which he flips the signal bit. Bob can guess a $sk_A[n-1]$ $= 0$ and communicating with Alice to see if his guess was correct. Finally, the recovered secret $\overline{s_A}$ can be verified by checking the distribution of $p_A - as_A$.

When Alice adds the error term $2e'_A$ to the key k_A, the number of queries required to recover $\overline{s_A}[i]$ increases, due to the complexity involved in eliminating the effect of the noise $2e'_A$. Here we adopt the strategy in [19], which is to run the attack on the same coefficient of $\overline{s_A}$ multiple times and look at the distribution of k.

More generally, Bob can choose public key p_B with $s_B \neq 0$[4]. In this way, $k_A = p_B\overline{s_A}$, which results in $k_A[n-1] = as_B\overline{s_A}[n-1] + 2\overline{s_A}[i] + 2k$. In particular, Bob firstly chooses $e_B[n-1-i] = 1$ to recover the value of $\frac{1}{2}as_B\overline{s_A}[n-1] + \overline{s_A}[i]$, then chooses $e_B[n-1-i] = 2$ to recover the value of $\frac{1}{2}as_B\overline{s_A}[n-1] + 2\overline{s_A}[i]$, thus recover $\overline{s_A}[i]$.

To compute the query complexity of the method 1, we compute the query complexity of each phase of the attack:

- 1) When $s_B \neq 0$ and the error term $2e'_A$ is added to the key k_A, determining the sign of each coefficient requires querying with p_B, where $e_B[n-1-i] = \pm 4$ [5] and $e_B[n-1-j] = k = 0$, a small constant number of times. Thus, the query complexity is constant for each coefficient, and the query complexity to recover the signs of all the coefficients of $\overline{s_A}$ is $2c'n$, where c' is a constant.
- 2) When $s_B = 0$ and the error term $2e'_A$ is added to the key k_A, recovering complete $\overline{s_A}$ needs $nt\alpha$ queries[6]. In addition, in order to eliminate the effect of the noise $2e'_A$, Bob needs to run the attack on the same coefficient of $\overline{s_A}$ multiple times. Thus, the attack complexity in this case would be $Cn\alpha$, where C is the constant.
- 3) When $s_B \neq 0$, recovering the secret $\overline{s_A}$ is run twice with different e_B. So, the number of queries required is $2Cn\alpha$.
- 4) Determining index j such that $\overline{s_A}[j] = 1$ requires running the attack for every coefficient i assuming that $\overline{s_A}[j] = 1$ starting with the first positive coefficient until such a j is found. So, the best case query complexity is $2Cn\alpha$, when the first positive coefficient turns out to be the required index with $\overline{s_A}[j] = 1$. The worst case query complexity is $2Cn^2\alpha$.

Thus the query complexity of the complete attack would be $2c'n + 2Cn^2\alpha \approx O(n^2\alpha)$ in the worst case and $2c'n + 2Cn\alpha \approx O(n\alpha)$ in the best case.

Method 2: In addition to the above method, we propose another simple method to recover the coefficients $\overline{s_A}$, where Bob chooses $p_B = c(kas_B + e_B)$, where s_B

[4] In particular, Bob chooses s_B such that $\overline{p_A}s_B[n-1] = 0$. This way Bob can obtain a s_B such that the value of $as_B\overline{s_A}[n-1]$ is small since $\overline{p_A}s_B = as_B\overline{s_A} + 2\overline{e_A}s_B$, where $\overline{e_A}s_B$ is small. We require such a s_B so that the term $as_B\overline{s_A}[n-1]$ in $k_A[n-1]$ cannot override $2\overline{s_A}[i] + 2k$ and the attack strategy can still be used. Since $\overline{p_A}$ is known to the Bob, he can solve the polynomial equation to find s_B such that $\overline{p_A}s_B[n-1] = 0$.

[5] when $s_B = 0$ and the error term $2e'_A$ is added to the key k_A, Choosing $k = 0$ and $e_B[n-1-i] = \pm 4$ will result in $k_A[n-1] = as_B\overline{s_A}[n-1] + 8\overline{s_A}[i] + 2e'_A[n-1]$, where $as_B\overline{s_A}[n-1]$ and $2e'_A[n-1]$ cannot override $8\overline{s_A}[i]$.

[6] $\overline{s_A}$ consists of s_A and Samp(H(p_A)), where s_A is sampled from the error distribution that has standard deviation α. Determining a coefficient value $\overline{s_A}[i]$ needs at most $t\alpha$ queries, where t is a constant.

and e_B satisfy that $(as_B\overline{s_A})[0] = 1$ [7] and $e_B\overline{s_A}[0] = \overline{s_A}[i]$ [8]. Thus, $(p_B\overline{s_A})[0] = c(kas_B\overline{s_A}[0] + e_B\overline{s_A}[0]) = c(\overline{s_A}[i] + k)$. He then performs the protocol honestly, except that he deliberately flips bit $w[0]$ to be 0.

In this way, $k_A = p_B\overline{s_A}$, which results in

$$k_A[0] = (p_B\overline{s_A})[0] = c(\overline{s_A}[i] + k). \tag{13}$$

$$sk_A[0] = c(\overline{s_A}[i] + k) \bmod q \bmod 2. \tag{14}$$

Notice that if $sk_A[0] = 0$ for all different c's, then $\overline{s_A}[i] + k = 0 \bmod q$. Thus, the k value reveals the value of $\overline{s_A}[i]$.

Similarly, as Bob performs the protocol mostly honestly (and both s_B and e_B qualify as small vectors until k and c remains small), Bob can compute the value sk_A, except for index 0, for which he flips the signal bit. Bob can guess a $sk_A[0] = 0$ and communicating with Alice to see if his guess was correct.

Analysis of the Complexity of the Attack Bob can adopt the technology in [17] to choose s_B and e_B, which can be done off-line. Therefore, the complexity of the attack is to query $\overline{s_A}[i]$ for each i. In particular, we can adopt the analysis in [19] to estimated complexity and the attack complexity is $C'n\alpha \approx O(n\alpha)$, where C' is the constant.

6.2 Attack on the PRKE in Key Reuse Mode

In this section, we consider PRKE in key reuse mode, where malicious Bob deliberately chooses p_B, w, and e_p with special structure to recover the private key s_A of Alice. In the attack, Bob is an active adversary intending to recover the private key s_A of the Alice. In every communication, Alice executes the key exchange honestly and Bob carefully chooses p_B, w and e_p.

Attack Principle In general, let's consider recovering coefficient $s_A[i], i \in [0, n-1]$. In the key exchange, if Bob chooses c (c is a non-zero constant) and k (k is a small integer) such that his public key $p_B = 2c$ and $e_p = kx^ic$, flips bit $w[2i]$ to be 0, then[9]

$$\begin{aligned} k_A = (p_Bs_A + 2e_p) \cdot e_p &= (2cs_A + 2kx^ic) \cdot kx^ic \\ &= (s_A + kx^i) \cdot 2kx^ic^2 \\ &= [(\sum_{j=0, j\neq i}^{n-1} s_A[j]x^j) + (s_A[i] + k)x^i] \cdot 2kx^ic^2 \end{aligned} \tag{15}$$

[7] [17] proposed an offline method to search $\overline{s_A}$ and here we adopt the same method.

[8] Specifically, he can choose e_B such that $e_B[t] = 0$ for all $t = 0, ..., n-1$ except $t = n-i$ and $e_B[n-i] = 1$, namely, $e_B = e_B[n-i]x^{n-i}$

[9] Similarly, first consider a simple case without adding $2e'_A$, where $k_A = (p_Bs_A + 2e_p) \cdot e_p$.

$$k_A[2i] = (s_A[i] + k) \cdot 2kc^2 \tag{16}$$

$$\begin{aligned}\sigma_A[2i] = \mathrm{Mod}_2(k_A[2i], w[2i]) &= (k_A[2i] + w[2i] \cdot \frac{q-1}{2} \mathrm{mod}\ q) \mathrm{mod}\ 2 \\ &= (((s_A[i] + k) \cdot 2kc^2) + w[2i] \cdot \frac{q-1}{2} \mathrm{mod}\ q) \mathrm{mod}\ 2 \\ &= (((s_A[i] + k) \cdot 2kc^2) \mathrm{mod}\ q) \mathrm{mod}\ 2 \end{aligned} \tag{17}$$

Notice that if $\sigma_A[2i] = 0$ for different c's, then $s_A[i] + k = 0 \bmod q$. Thus, the k value reveals the value of $\overline{s_A}[i]$.

Similarly, as Bob performs the protocol mostly honestly, Bob can compute the value σ_A, except for index $2i$, for which he flips the signal bit. Bob can guess a $\sigma_A[2i] = 0$ and communicating with Alice to see if his guess was correct.

Remark: In fact, Bob has many choices to choose p_B and e_p.[10]

Next, we describe how to extend the attack in simple case to general case that includes addition of an error term $2e'_A$, namely, $k_A = (p_B s_A + 2e_p) \cdot e_p + 2e'_A$. In previous section, it is easy to see that as Bob tries different k's, Bob can always determine the coefficient $s_A[i]$ from $s_A[i] + k = 0 \bmod q$. However, when the error term $2e'_A[2i]$ is added to $k_A[2i]$,

$$k_A[2i] = (s_A[i] + k) \cdot 2kc^2 + 2e'_A[2i] \tag{18}$$

$$\sigma_A[2i] = \mathrm{Mod}_2(k_A[2i], w[2i]) = (((s_A[i] + k) \cdot 2kc^2 + 2e'_A[2i]) \mathrm{mod}\ q) \mathrm{mod}\ 2 \tag{19}$$

if Bob dose as before, what he recovers is $s_A[i]$ with perturbations.

In this case, the choice of p_B, e_p and σ_B remain the same as in simple case but there is difference in determining the value of $s_A[i]$. The strategy here is to run the attack on the same coefficient $s_A[i]$ multiple times and look at the distribution of$s_A[i]$. Compared to the simple case above, the number of communications increases due to the complexity involved in eliminating the effect of the noise $2e'_A[2i]$.

Analysis of the Complexity of the Attack The complexity of the attack is to query $s_A[i]$ for each i. In particular, to determine $s_A[i]$, firstly, Bob needs to try different k's. And for the same k, Bob needs to try different c's. In addition, Bob needs to repeat a certain number of communications to eliminate the effect of the noise $2e'_A[2i]$. Thus, to recover the coefficients of s_A, Bob needs almost $tn\alpha$ communications , where t is a constant, and the attack complexity is $O(n\alpha)$.

[10] In particular, to recover coefficient $s_A[i]$, Bob chooses p_B such that $p_B[t] = 0$ for all $t = 0, ..., n-1$ except $t = 0$ and $p_B[0] = 1$, and e_p such that $e_p[t] = 0$ for all $t = 0, ..., n-1$ except $t = 0, t = 1, t = i-1$ and $e_p[0] = 2c$, $e_p[1] = 1$, $e_p[i-1] = kc$. Thus, $k_A[i] = (p_B s_A e_p)[i] + (2e_p e_p)[i] = 2cs_A[i] + 2ck = 2c(s_A[i] + k)$.
Bob can also chooses s_B such that $as_A s_B[0] = 1$ [17], and e_B such that $e_B[t] = 0$ for all $t = 0, ..., n-1$ except for $n-i$ and $e_B[n-i] = 1$, and e_p such that $e_p[t] = 0$ for all $t = 0, ..., n-1$ except $t = 0$ and $e_p[0] = 1$. In addition, Bob chooses p_B such that $p_B = 2kcas_B + 2e_B$. Thus, $k_A[0] = (p_B s_A e_p)[0] + (2e_p e_p)[0] = 2c(k + s_A[i])$.

7 An Improved KERK

In RLWE key exchange, the public key is required to be a RLWE sample. If one party chooses a RLWE sample with special structure, it could cause trouble to the other party. In fact, the reason Bob can successfully recover the private key of Alice is that Alice can't check Bob's public key. And the key to carry out the signal leakage attack [15] is that Bob can't check Alice's public key. Although we discuss the Ding12 in this paper, we believe the same goes for other other RLWE-based schemes, including KEM (or key exchange) and PKE.

Alice		**Bob**
Public key: $p_A = as_A + 2e_A \in R_q$		Public key: $p_B = as_B + 2e_B \in R_q$
Private key: $s_A \in R_q$		Private key: $s_B \in R_q$
where $s_A, e_A \leftarrow D_{\mathbb{Z}^n,\sigma}$		where $s_B, e_B \leftarrow D_{\mathbb{Z}^n,\sigma}$
	$\xrightarrow{p_A}$	
		Sample $e'_B, e''_B \leftarrow D_{\mathbb{Z}^n,\sigma}$
		Samp(H(p_A))$\leftarrow D_{\mathbb{Z}^n,\sigma}$
		Samp(H(p_B))$\leftarrow D_{\mathbb{Z}^n,\sigma}$
Sample $e'_A, e''_A \leftarrow D_{\mathbb{Z}^n,\sigma}$		$\overline{p_A} = a\cdot$ Samp(H(p_A)) $+ 2e'_B + p_A$
Samp(H(p_A))$\leftarrow D_{\mathbb{Z}^n,\sigma}$		$k_B = \overline{p_A}(s_B +$ Samp(H(p_B))) $+ 2e''_B$
Samp(H(p_B))$\leftarrow D_{\mathbb{Z}^n,\sigma}$		$w = \mathrm{Sig}(k_B) \in \{0,1\}^n$
	$\xleftarrow{p_B, w}$	
$\overline{p_B} = a\cdot$ Samp(H(p_B)) $+ 2e'_A + p_B$		$sk_B = \mathrm{Mod}_2(k_B, w) \in \{0,1\}^n$
$k_A = \overline{p_B} \cdot (s_A +$ Samp(H(p_A))) $+ 2e''_A$		
$sk_A = \mathrm{Mod}_2(k_A, w) \in \{0,1\}^n$		

Table 7. an improved KERK

Therefore, in order to solve the trouble, we need a mechanism where any party can check the public key of the other party, or can construct a new "public key" of the other party, which is a good RLWE sample. In particular, in this section we provide a mechanism where any party can construct a new "public key", which is a good RLWE sample.

Similarly, we also take DING12 as an example. Recall in the key exchange (Table 3), Bob constructs a new "public key" of Alice, $\overline{p_A} = a\cdot$ Samp(H(p_A)) $+ 2e'_B + p_A$, where Samp(H(p_A)) is indistinguishable from uniform and $2e'_B$ is chosen by Bob. Therefore, $\overline{p_A}$ can be regarded as a RLWE sample and is indistinguishable from uniform. Although malicious Alice chooses a RLWE sample p_A with special structure, $\overline{p_A}$ is a good RLWE sample. Therefore, it can effectively prevent malicious Alice from recovering the private key of Bob when keys are reused.

Based on the Table 2, we propose an improved KERK (Table 5), which can prevent malicious Bob from recovering the private key of Alice. In the improved key exchange, Alice constructs a new "public key" of Bob, $\overline{p_B} = a\cdot$ Samp(H(p_B)) $+ 2e'_A + p_B$, which can be regarded as a RLWE sample and is indistinguishable from uniform. Although malicious Bob chooses a bad RLWE sample p_B with special structure, $\overline{p_B}$ is a good RLWE sample. Therefore, it can effectively prevent any malicious party from recovering the private key of the other party when keys are reused.

8 Conclusion

In this work, we have analyzed two countermeasures against the signal leakage attack: KERK and PRKE. In particular, we develop two attacks on these two countermeasures. In these attacks, malicious Bob can recover the private key of Alice by carefully choosing what he sends and observing whether the final shared keys match or not. After analyzing the complexities, we get the conclusion that these attacks are practical and effective. In fact, other RLWE-based schemes, including KEM (or key exchange) and PKE, are also vulnerable to these attacks. In response to these attacks, we propose a mechanism which can resist these attacks and give an improved KERK to illustrate the mechanism.

References

1. Diffie W, Hellman M. New directions in cryptography[J]. IEEE transactions on Information Theory, 1976, 22(6): 644-654.
2. Shor P W. Algorithms for quantum computation: Discrete logarithms and factoring[C]//Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on. Ieee, 1994: 124-134.
3. National Institute of Standards and Technology: Announcing request for nominations for public-key post-quantum cryptographic algorithms (2016),https://csrc:nist:gov/news/2016/public-key-post-quantum-cryptographic-algorithms
4. Lyubashevsky V, Peikert C, Regev O. On ideal lattices and learning with errors over rings[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 2010: 1-23.
5. Ding J, Xie X, Lin X. A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem[J]. IACR Cryptology EPrint Archive, 2012, 2012: 688.
6. Peikert C. Lattice cryptography for the internet[C]//International Workshop on Post-Quantum Cryptography. Springer, Cham, 2014: 197-219.
7. Bos J W, Costello C, Naehrig M, et al. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem[C]//Security and Privacy (SP), 2015 IEEE Symposium on. IEEE, 2015: 553-570.
8. Zhang J, Zhang Z, Ding J, et al. Authenticated key exchange from ideal lattices[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 2015: 719-751.
9. Alkim E, Ducas L, Pöppelmann T, et al. Post-quantum Key Exchange-A New Hope[C]//USENIX Security Symposium. 2016, 2016.
10. Bos J, Costello C, Ducas L, et al. Frodo: Take off the ring! practical, quantum-secure key exchange from LWE[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016: 1006-1018.

11. Alkim E, Ducas L, Pöppelmann T, et al. NewHope without reconciliation[J]. IACR Cryptology ePrint Archive, 2016, 2016: 1157.
12. Ding J, Alsayigh S, Lancrenon J, et al. Provably secure password authenticated key exchange based on RLWE for the post-quantum world[C]//Cryptographers Track at the RSA Conference. Springer, Cham, 2017: 183-204.
13. Ding J, Alsayigh S, Saraswathy R V, et al. Leakage of signal function with reused keys in RLWE key exchange[C]//Communications (ICC), 2017 IEEE International Conference on. IEEE, 2017: 1-6.
14. Rescorla E. The transport layer security (TLS) protocol version 1.3[R]. 2018.
15. Gao X, Ding J, Li L, et al. Practical randomized RLWE-based key exchange against signal leakage attack[J]. IEEE Transactions on Computers, 2018 (1): 1-1.
16. Kirkwood D, Lackey B C, McVey J, et al. Failure is not an option: standardization issues for post-quantum key agreement[C]//Talk at NIST workshop on Cybersecurity in a Post-Quantum World: http://www. nist. gov/itl/csd/ct/post-quantum-crypto-workshop-2015. cfm. 2015, 2.
17. Fluhrer S R. Cryptanalysis of ring-LWE based key exchange with key share reuse[J]. IACR Cryptology ePrint Archive, 2016, 2016: 85.
18. Bernstein D J, Bruinderink L G, Lange T, et al. HILA5 pindakaas: on the CCA security of lattice-based encryption with error correction[C]//International Conference on Cryptology in Africa. Springer, Cham, 2018: 203-216.
19. Ding J, Fluhrer S, Rv S. Complete Attack on RLWE Key Exchange with Reused Keys, Without Signal Leakage[C]//Australasian Conference on Information Security and Privacy. Springer, Cham, 2018: 467-486.
20. Gao X, Ding J, Liu J, et al. Post-Quantum Secure Remote Password Protocol from RLWE Problem[C]//International Conference on Information Security and Cryptology. Springer, Cham, 2017: 99-116.
21. Ajtai M. Generating hard instances of lattice problems[C]//Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. ACM, 1996: 99-108.
22. Brakerski Z, Gentry C, Vaikuntanathan V. (Leveled) fully homomorphic encryption without bootstrapping[J]. ACM Transactions on Computation Theory (TOC-T), 2014, 6(3): 13.
23. Brakerski Z, Vaikuntanathan V. Fully homomorphic encryption from ring-LWE and security for key dependent messages[C]//Annual cryptology conference. Springer, Berlin, Heidelberg, 2011: 505-524.
24. Applebaum B, Cash D, Peikert C, et al. Fast cryptographic primitives and circular-secure encryption based on hard learning problems[M]//Advances in Cryptology-CRYPTO 2009. Springer, Berlin, Heidelberg, 2009: 595-618.
25. Gao X, Ding J, Li L, et al. Efficient Implementation of Password-Based Authenticated Key Exchange from RLWE and Post-Quantum TLS[R]. Cryptology ePrint Archive, Report 2017/1192, 2017. http://eprint. iacr. org/2017/1192, 2017.
26. Saarinen M J O. HILA5: on reliability, reconciliation, and error correction for ring-LWE encryption[C]//International Conference on Selected Areas in Cryptography. Springer, Cham, 2017: 192-212.
27. Ding J, Saraswathy R V, Alsayigh S, et al. How to validate the secret of a Ring Learning with Errors (RLWE) key[J].
28. Liu C, Zheng Z, Zou G. Key Reuse Attack on NewHope Key Exchange Protocol[J].
29. D'Anvers J P, Vercauteren F, Verbauwhede I. On the impact of decryption failures on the security of LWE/LWR based schemes[J].
30. Li C. A Key Recovery Attack on Streamlined NTRU Prime[J].