Leakage of Signal function with reused keys in RLWE key exchange

Jintai Ding[1], Saed Alsayigh[1], Saraswathy RV[1], Scott Fluhrer[2], and Xiaodong Lin[3]

[1] University of Cincinnati
[2] Cisco Systems
[3] Rutgers University

Abstract. In this paper, we show that the signal function used in Ring-Learning with Errors (RLWE) key exchange could leak information to find the secret s of a reused public key $p = as + 2e$. When RLWE public keys reused for long term, it can be exploited by initiating multiple sessions with the honest party and analyzing the output of the signal function. Experiments have confirmed the success of our attack in recovering the secret.

Keywords: RLWE, key exchange, post quantum, key reuse, active attacks.

1 Introduction

Key exchange is an integral part of cryptography. It is required for establishing secure keys for encryption of data between two parties. The breakthrough idea of Diffie-Hellman in [9] provided an algorithm for secure key exchange and has been used since then in many applications of cryptography. There are also other variants of DH that have been proposed and used over the years including Elliptic curve DH. The necessity to look for an alternative to DH is mainly for security with the existence of quantum computers. With Shor's algorithm [20], the discrete log problem (the hardness of which the security of currently used key exchange protocols are based on) can be solved in polynomial time with the help of a quantum computer compromising the security of the protocols. Recently, NSA has announced plans to transition to quantum resistant cryptographic primitives for its Suite B cryptographic algorithms. One of the potential candidates for post quantum key exchange includes the key exchange from RLWE proposed in [14] and an authenticated version in [21]. Other follow up work on RLWE based key exchange protocols include [18], [6], [4] and [11], which follow the same approach as in [14] with minor modifications. The implementation of [4] uses the centered binomial gaussian for the error terms of the RLWE samples and achieve significant performance improvements and has recently been chosen as a candidate for integration in google chrome canary browser for their post quantum experiment[2].In an RLWE key exchange between two parties, the key computation yields approximately equal values. To derive a shared key using the approximately equal key computation, the responder sends a signal to the initiator indicating the region in which the approximately equal value lies. Using this information, both the parties agree on a shared key. The concern addressed in this work is the ability of an active adversary to perform an attack on the RLWE based key exchange protocols using this signal function to recover the corresponding secret of a reused public key. The signal function and key exchange are reviewed in section 4.

1.1 Previous Work

An attack on RLWE key exchange for reused public keys was described by Fluhrer in [10]. The idea of the attack is to deviate from the protocol in creating the adversary's public key and

extract information about the secret of the other party. The attack relies on finding s_A in an adversary's public key of $p_A = jas_A + ke_A$, (where k, j are integers) such that $as_As[0] = \pm 1$ and once such an s_A is found, the attack aims to find the secret s of the reused key $p = as + 2e$ of the honest party by fixing $k = as_As[0]$ and looking for signal variations when changing j. The signal function region is slightly modified and even though the approach is in the right direction, this does not work in the case of the key exchange [14] discussed in this paper. There is also an attack [13] using the CRT (Chinese Remainder Theorem) basis of R_q on the one pass case of the HMQV key exchange protocol from RLWE. This attack recovers every CRT coefficient of the secret s of a key $p = as + 2e$ in order to recover s. The complexity of the attack is claimed to be $n.q.\frac{q-1}{2n}$.

1.2 Our Contributions

We exploit the leakage of the signal function to present an attack on RLWE based key exchange that finds the exact value of the secret s corresponding to a reused RLWE key of $p = as + 2e$. This work follows the idea in [10] but uses a different approach to using the signal function to complete the attack. We provide a detailed description on how such an attack is performed by analyzing the number of signal changes of each of the coefficients of $p_As + 2g$ where $p_\mathcal{A} = as_\mathcal{A} + ke_\mathcal{A}$ with $s_\mathcal{A}, e_\mathcal{A}$ chosen specifically by the adversary to extract the value of s, and g is sampled from the error distribution. The choice of $s_\mathcal{A}, e_\mathcal{A}$ is explained in section 4.

We also describe how to refine the queries further to eliminate the ambiguity of the $\pm$ sign of the coefficients and recover the exact values. Experiments have verified our estimation for the number of signal changes for different values of the coefficient of s. The goal of the work is to show that RLWE keys when reused in key exchange can be exploited and broken. The success of such attacks comes from the hardness of distinguishing RLWE samples from uniform.

2 Preliminaries

2.1 Learning with Errors and RLWE

The Learning with Errors (LWE) problem is a generalization of the parity-learning problem introduced by Oded Regev in 2005 [19]. Regev also showed a quantum reduction from solving worst case Lattice problems such as the Shortest Vector Problem (SVP) and the Shortest Independent Vectors Problem (SIVP) to solving LWE in the average case. In 2009, Peikert showed a classical reduction from variants of the shortest vector problem to corresponding versions of LWE[17]. The LWE problem is parametrized by a modulus q, dimension n and an error distribution χ on $\mathbb{Z}_q$. Then, the decision version of the LWE problem is to distinguish the following two distributions: (**a**, **a**.**s** + e) and (**a**, **b**), where **a**, **s**, **b** $\in \mathbb{Z}_q^n$ are sampled uniformly at random and $e \leftarrow \chi$ from the error distribution. The search version is to find s given $poly(n)$ number of samples $(a_i, a_i.s + e)$. Ring-Learning with Errors (RLWE) is the version of LWE using polynomial rings and is preferred over LWE due to its efficiency and potential for practical implementations. The hardness of RLWE is also established by reductions to solving hard problems in ideal lattices. We provide the definition of the Discrete Gaussian distribution (error distribution) here:

Definition 1. *[21] For any positive real $\alpha \in \mathbb{R}$, and vectors $c \in \mathbb{R}^n$, the continuous Gaussian distribution over $\mathbb{R}^n$ with standard deviation centered at v is defined by the probability function $\rho_{\alpha,c}(x) = (\frac{1}{\sqrt{2\pi\alpha^2}})^n exp(\frac{-\|x-c\|^2}{2\alpha^2})$. For integer vectors $c \in \mathbb{R}^n$, let $\rho_{\alpha,c}(\mathbb{Z}^n) = \sum_{x\in\mathbb{Z}^n} \rho_{\alpha,c}(x)$. Then, we define the discrete Gaussian distribution over $\mathbb{Z}^n$ as $D_{\mathbb{Z}^n,\alpha,c}(x) = \frac{\rho_{\alpha,c}(x)}{\rho_{\alpha,c}(\mathbb{Z}^n)}$, where $x \in \mathbb{Z}^n$. The subscripts s and c are taken to be 1 and 0 (respectively) when omitted.*

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 the ring $R_q = \mathbb{Z}_q[x]/\langle f(x)\rangle$ 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 $\mathbb{Z}_q[x]$). Let χ_α denote the Discrete Gaussian distribution on R_q with parameter α. For any polynomial $p \in R$ (or R_q), let the norm $\|p\|$ be defined as the norm of the corresponding coefficient vector in $\mathbb{Z}$ (or $\mathbb{Z}_q$). Let $p[i]$ denote the i-th coefficient of p, equivalently i-th index in the coefficient vector of p. Below are two lemmas that help ensure the correctness of the key exchange protocol.

Lemma 1 ([21]). *Let $f(x)$ and R be defined as above. Then, for any $s, t \in R$, we have $\|s \cdot t\| \leq \sqrt{n} \cdot \|s\| \cdot \|t\|$ and $\|s \cdot t\|_\infty \leq n \cdot \|s\|_\infty \cdot \|t\|_\infty$.*

Lemma 2 ([12, 16]). *For any real number $\alpha = \omega(\sqrt{\log n})$, we have $\Pr_{\boldsymbol{x} \leftarrow \chi_\alpha}[\|\boldsymbol{x}\| > \alpha\sqrt{n}] \leq 2^{-n+1}$.*

Let $s \leftarrow R_q$ be a uniformly chosen element of the ring R_q, as defined above. We define A_{s,χ_α} to be the distribution of the pair $(a, as + e) \in R_q \times R_q$, where $a \leftarrow R_q$ is uniformly chosen and $e \leftarrow \chi_\alpha$ is independent of a.

Definition 2 (Ring-LWE Assumption[15]). *Let R_q, χ_α be defined as above, and let $s \leftarrow R_q$ be uniformly chosen. The (special case) ring-LWE assumption $RLWE_{q,\alpha}$ states that it is hard for any PPT algorithm to distinguish A_{s,χ_α} from the uniform distribution on $R_q \times R_q$ with only polynomial samples.*

The search version of RLWE is for a PPT algorithm to find s rather than distinguish the two distributions. For certain parameter choices, the two forms are polynomially equivalent [15]. The *normal form* [7, 8] of the RLWE problem is by modifying the above definition by choosing s from the error distribution χ_α rather than uniformly. It has been proven that the ring-LWE assumption still holds even with this variant [5, 15].

Proposition 1 ([15]). *Let n be a power of 2, let α be a real number in $(0, 1)$, and q a prime such that $q \bmod 2n = 1$ and $\alpha q > \omega(\sqrt{\log n})$. Define $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$ as above. Then there exists a polynomial time quantum reduction from $\tilde{O}(\sqrt{n}/\alpha)$-SIVP (Short Independent Vectors Problem) in the worst case to average-case RLWE$_{q,\beta}$ with ℓ samples, where $\beta = \alpha q \cdot (n\ell/\log(n\ell))^{1/4}$.*

3 Transport Layer Security (TLS)

An important application of key exchange protocols is in the Transport Layer Security (TLS) which is used to secure http traffic in https websites and SSH (Secure Shell). The Transport Layer security and SSL (Secure Sockets Layer, predecessor of TLS) are cryptographic protocols in the application layer of TCP/IP reference model and presentation layer in the OSI model to provide security in a communication network. The objective of the TLS protocol is to ensure privacy and data integrity between communicating applications. The protocol consists of 2 layers - TLS record protocol and the other layer being protocols that are designed to establish a secure connection (Handshake Protocol and the Alert Protocol). The handshake protocol is run before any application data is transmitted and enables the client and server to establish the algorithm and shared key for encrypted communication.

The TLS 1.3 draft [1] includes support to a 0-RTT mode in which the server maintains a long term public key which is sent through a `ServerConfiguration` message to the client. The client can use this static key to secure communication of application data in future connections. This is the motivation behind an attack on RLWE key exchange for reused public keys, although this has been changed in the later draft 13 of TLS 1.3 [3]. The draft 13 of TLS 1.3 proposes to use a PSK(preshared key) identity that is sent to the client on the initial handshake to be used for encrypting early data on future handshakes.

4 The Attack

Before describing the attack on an RLWE based key exchange, we briefly recall the simple key exchange protocol in [14]. Let the notations be as defined in section 2.

4.1 Signal function (Reconciliation)

We define the Signal function used in the key exchange protocol discussed in this work (refer [21]). Given $\mathbb{Z}_q = \{-\frac{q-1}{2}, \ldots, \frac{q-1}{2}\}$ and the middle subset $E := \{-\lfloor \frac{q}{4} \rfloor, \ldots, \lfloor \frac{q}{4} \rceil\}$, we define Sig as the characteristic function of the *complement* of E: $Sig(v) = 0$ if $v \in E$ and 1 otherwise.

Another important function used for deriving the final shared key in the key exchange is defined as follows, $\mathsf{Mod}_2 \colon \mathbb{Z}_q \times \{0,1\} \to \{0,1\}$:

$$\mathsf{Mod}_2(v, w) = (v + w \cdot \frac{q-1}{2}) \bmod q \bmod 2.$$

Then the key exchange protocol is as described below:

Init: Party A chooses a uniformly random from R_q and a secret $s_A \leftarrow \chi_\alpha$ and computes $p_A = as_A + 2e_A$, where $e_A \leftarrow \chi_\alpha$. Party A then sends p_A to party B.

Response: On receiving p_A, party B chooses a secret element s_B and $e_B \leftarrow \chi_\alpha$. Party B then computes $p_B = as_B + 2e_B$, $k_B = p_A s_B + 2g_A$ and $w_B = Sig(k_B)$, sends p_B, w_B. party B obtains a shared key $sk_B = \mathsf{Mod}_2(k_B, w_B)$.

Finish: On receiving p_B, w_B from party B, party A computes $k_A = s_A p_B + 2g_A$, where $g_A \leftarrow \chi_\alpha$ and obtains the shared key $sk_A = \mathsf{Mod}_2(k_A, w_B)$.

Party A		Party B
Sample $s_A, e_A \leftarrow \chi_\alpha$ Secret Key: $s_A \in R_q$ Public Key: $a, p_A = as_A + 2e_A \in R_q$	$\xrightarrow{p_A}$	Sample $s_B, e_B \leftarrow \chi_\alpha$ Secret Key: $s_B \in R_q$ Public Key: $a, p_B = as_B + 2e_B \in R_q$
	$\xleftarrow{p_B, w_B}$	Sample $g_B \leftarrow \chi_\alpha$ Set $k_B = p_A s_B + 2g_B$ Find $w_B = Sig(k_B) \in \{0,1\}^n$
Sample $g_A \leftarrow \chi_\alpha$ Set $k_A = p_B s_A + 2g_A$ Find $sk_A = \mathsf{Mod}_2(k_A, w_B) \in \{0,1\}^n$		Find $sk_B = \mathsf{Mod}_2(k_B, w_B) \in \{0,1\}^n$

Fig. 1. Original Protocol

4.2 Simplified Attack

Suppose that $\mathcal{A}$ is an active adversary with the knowledge of p_B and with the ability to initiate any number of key exchange sessions with party B to query for recovering the secret s_B. In performing the attack, an adversary plays the role of party A in the protocol and initiates key exchange sessions with party B. $\mathcal{A}$ creates p_A by deviating from the protocol; we denote the

deviated public key of the adversary as $p_{\mathcal{A}}$ and the corresponding secret and error terms of the adversary as $s_{\mathcal{A}}$ and $e_{\mathcal{A}}$ respectively.

First, we describe a simplified version of the attack when the error terms g_A, g_B are not added to the key computation of k_A, k_B of parties A and B respectively.The adversary chooses secret $s_{\mathcal{A}}$ to be 0 and $e_{\mathcal{A}}$ to be the identity element 1 in R_q,and computes $p_{\mathcal{A}} = as_{\mathcal{A}} + ke_{\mathcal{A}} = ke_{\mathcal{A}}$, k takes values in $\mathbb{Z}_q$. This results in the key computation of B to be $k_B = ks_B$. Hence the signal w_B sent by the party B for each coefficient is the signal of each coefficient of s_B and leaks its value.

Simulating party B's response: We build an oracle $\mathcal{S}$ that simulates party B's action in the protocol on receiving a given input public key. We assume that p_B is fixed for party B and $\mathcal{S}$ has access to the secret s_B. On receiving $p_{\mathcal{A}}$ from $\mathcal{A}$, $\mathcal{S}$ computes $k_B = p_{\mathcal{A}} s_B$ according to the protocol. Then, $\mathcal{S}$ computes the signal $w_B = Sig(k_B)$ and outputs w_B.

Then, the attack is executed as follows:

Step 1**:** The Adversary $\mathcal{A}$ invokes the oracle $\mathcal{S}$ with input $p_{\mathcal{A}} = ke_{\mathcal{A}}$ for $e_{\mathcal{A}} = 1$ in R_q. Here, k takes values from 0 to $q-1$. As we change k value from 0 to $q-1$, $\mathcal{A}$ can make a correct guess of the value of $s_B[i]$ based on the number of times the signal $w_B[i]$ changes, for each coefficient i of s_B. As k takes values from 0 to $q-1$, the value of $k_B[i]$ changes in k multiples of $s_B[i]$ (refer figure 2) and there are changes in the signal value when $ks_B[i]$ is near the boundary values of the signal region E defined in section 4.1. Hence, there will be exactly $2s_B[i]$ number of changes in signal for any i-th coefficient of s_B. For example, $w_B[0]$ remains the same if $s_B[0]$ is 0 for different k, changes twice if $s_B[0]$ is 1 and so on. But the adversary can only guess the value upto the $\pm$ sign, since a value of 1 or -1 gives the same number of signal changes. This is because for a value of $-s_B[i]$, the value of $k_B[i]$ still changes in k multiples of $s_B[i]$ but in the reverse direction.

We query party B again to resolve the ambiguity of the sign in the s_B coefficients and determine the exact value of s_B. Suppose that the adversary $\mathcal{A}$ has performed the above steps and has determined the value of each coefficient of s_B upto sign.

Step 2**:** $\mathcal{A}$ invokes the oracle $\mathcal{S}$ to query with input $(1+x)p_{\mathcal{A}}$. By doing this, $\mathcal{A}$ is able to see the signal function value of $p_{\mathcal{A}}((1+x)s_B)$ that is output by $\mathcal{S}$. Thus, again by checking the number of signal changes, $\mathcal{A}$ can now find the values of the coefficients of $(1+x)s_B$, which are $s_B[0] - s_B[n-1], s_B[1] + s_B[2], \ldots, s_B[n-2] + s_B[n-1]$ upto $\pm$ sign.

So, with the additional information about these coefficients, we can determine if each pair of coefficients $s_B[i], s_B[j]$ have equal or opposite sign, hence narrowing down to only two possibilities of s_B and $-s_B$.

Step 3**:** Consider the pair of coefficients $s_B[0], s_B[n-1]$, then by recovering the value of $s_B[0] - s_B[n-1]$ upto $\pm$ sign in Step 2, and already knowing $s_B[0], s_B[n-1]$ values upto sign from Step 1, $\mathcal{A}$ determines if $s_B[0]$ and $s_B[n-1]$ have the same or opposite sign.

Step 4: Repeat Step 3 for every pair of coefficients $s_B[i], s_B[j]$, i from 0 to $n-2$, j from 1 to $n-1$ with the value of $s_B[i] + s_B[j]$ upto $\pm$ sign from Step 2 to determine if they have equal or opposite signs.

Once the adversary reaches this stage, he only has to guess the sign of $s_B[0]$ and the rest of the coefficients follow since we have determined if every pair of coefficients $s_B[i], s_B[j]$ have equal or opposite signs.

Step 5: Since a and p_B are public, $\mathcal{A}$ computes $p_B - as_B$ and verifies the distribution of the result. If $\mathcal{A}$ correctly guesses the sign of $s_B[0]$ and hence all the coefficients, then the resulting distribution of $p_B - as_B$ is the distribution of e_B, which is discrete gaussian. Otherwise, $\mathcal{A}$ knows that the guess for the sign of $s_B[0]$ is incorrect and can flip the sign to obtain the correct s_B value.

Thus, the adversary is able to determine the exact value of s_B without any ambiguity at the end of the execution by querying party B when B re-uses the same key for every query. The success of the attack also shows the significance of the role of the signal function in the key exchange protocol.

4.3 Extension of the Attack

In this section, we describe how to extend the attack in section 4.2 described to the actual protocol described in figure 1 that includes addition of an error term g_A, g_B to the key computation k_A, k_B of parties A and B respectively. The choice of $s_{\mathcal{A}}$ and $e_{\mathcal{A}}$ remain the same as in section 4.2 but in this case, there is a difference in counting the number of signal changes to identify the value of s_B coefficients. This is because of the fluctuations caused by the addition of g_B by party B in the computation of k_B.
Simulating party B's response: We build an oracle $\mathcal{S}$ that simulates party B's action in the protocol on receiving a given input p. We assume that p_B is fixed for party B and $\mathcal{S}$ has access to the secret s_B. On receiving $p_{\mathcal{A}}$ from $\mathcal{A}$, $\mathcal{S}$ samples $g_B \leftarrow \chi_\alpha$ and computes $k_B = p_{\mathcal{A}} s_B + 2g_B$ according to the protocol. Then, $\mathcal{S}$ computes the signal $w_B = Sig(k_B)$ and outputs w_B.

Effect of g_B on the signal changes: In the previous section, it is easy to see that as we loop k values in $\mathbb{Z}_q$, there are changes in the signal exactly $2s_B[i]$ times for any i-th coefficient of s_B. When the error term g_B is added to k_B, there are some frequent changes, which we call fluctuations in the signal value at the boundary values of set E. This is because the error term g_B pulls the value of the key k_B back and forth when $k_B[i]$ is near the boundary, for each i. This stabilizes as k becomes larger and $k_B[i]$ moves away from the boundary to resist impact by $g_B[i]$. In this case, we ignore the fluctuations, not counting them as a signal change. This is illustrated in section 5 with the example.

The attack works the same way when we use the randomized signal function in [14], which is used to remove any bias in the shared key generated from k_B, for odd q. This has also been verified in our experiments.

4.4 Attack Improved

In the above attack, the public key of the attacker is only k times the identity 1 in R_q. It is possible for party B to defend against such an attack by verifying if the public key it receives is a constant polynomial in R_q. To overcome this, the adversary can sample $s_{\mathcal{A}}$ from χ_α, choose $e_{\mathcal{A}}$ to be 1 as before and compute $p_{\mathcal{A}} = as_{\mathcal{A}} + ke_{\mathcal{A}}$ as his public key in the RLWE form. Now, party B cannot distinguish the public key of the attacker from uniform. The key computation of party B in this case is $k_B = as_{\mathcal{A}} s_B + ks_B + 2g_B$. The value $as_{\mathcal{A}} s_B$ is constant as we loop over k values. So, we are still looking at the signal changes of $s_B[i]$ as we loop around k values 0 to $q-1$. The difference is that in the case of the simplified attack, when we start looping k values from 0, k_B changes in multiples of $s_B[i]$ starting with 0 while in this case, we start with the constant value of $as_{\mathcal{A}} s_B[i]$ for each coefficient i. The algorithm for the attack is the same as described in section 4.2.

4.5 Adversary query complexity

From the above description of the attack, it is clear that the adversary $\mathcal{A}$ needs q queries (varying k from 0 to $q-1$) for determining all the coefficients of the secret s_B upto $\pm$ sign. Again, by using q queries for the adversary's public key of $(1+x)p_{\mathcal{A}}$, he can resolve the ambiguity of the sign of the coefficients. Hence, the number of queries required for the attack to find the exact value of the secret is $2q$. Since q is $poly(n)$, we claim that the key is compromised in $poly(n)$ queries.

5 Toy Example

We demonstrate the attack described above with the help of a toy example in this section. The example shows the steps to recover the 0-th coefficient of s_B. The other coefficients of s_B can be recovered by following the same steps. First, we show the attack for the simplified case when g_B is not added to party B's key.
Let $n = 4$, $q = 17$, $\alpha = 1.6$ (Choosing such a value of α for the sake of the example to obtain reasonable sample values). Sampling a uniformly random from R_q and s_B, e_B from χ_α, let $a = (9, 4, 9, 3)$, $s_B = (-1, 0, 0, 2)$ and $e_B = (1, -1, -1, 0)$. Then $p_B = as_B + 2e_B = (2, -7, 0, -2)$.
The signal region $E \coloneqq \{-\lfloor \frac{q}{4} \rfloor, \ldots, \lfloor \frac{q}{4} \rceil\}$ for $q = 17$ is $E \coloneqq \{-4, -3, -2, -1, 0, 1, 2, 3, 4\}$
Choose $s_\mathcal{A}, e_\mathcal{A}$ of the adversary to be $0, 1$ respectively. So $p_\mathcal{A} = k$.
Oracle $\mathcal{S}$: On input of $p_\mathcal{A} = k$ from the adversary, $\mathcal{S}$ computes $p_\mathcal{A}s_B = ks_B$ and $w_B = Sig(ks_B)$, outputs w_B.

Step 1: Below are the values of the 0-th coefficient of w_B output from the oracle $\mathcal{S}$ on invocation by the adversary $\mathcal{A}$ for k ranging from 0 to $q - 1 = 16$.

$$
\begin{array}{lll}
k = 0: & w_B[0] = 0 & p_\mathcal{A}s_B[0] = 0.s_B[0] = 0 \\
k = 1: & w_B[0] = 0 & p_\mathcal{A}s_B[0] = s_B[0] = -1 \\
k = 2: & w_B[0] = 0 & p_\mathcal{A}s_B[0] = 2s_B[0] = -2 \\
k = 3: & w_B[0] = 0 & p_\mathcal{A}s_B[0] = 3s_B[0] = -3 \\
k = 4: & w_B[0] = 0 & p_\mathcal{A}s_B[0] = 4s_B[0] = -4 \\
k = 5: & w_B[0] = 1 & p_\mathcal{A}s_B[0] = 5s_B[0] = -5 \\
k = 6: & w_B[0] = 1 & p_\mathcal{A}s_B[0] = 6_B[0] = -6 \\
k = 7: & w_B[0] = 1 & p_\mathcal{A}s_B[0] = 7s_B[0] = -7 \\
k = 8: & w_B[0] = 1 & p_\mathcal{A}s_B[0] = 8s_B[0] = -8 \\
k = 9: & w_B[0] = 1 & p_\mathcal{A}s_B[0] = 9s_B[0] = 8 \\
k = 10: & w_B[0] = 1 & p_\mathcal{A}s_B[0] = 10s_B[0] = 7 \\
k = 11: & w_B[0] = 1 & p_\mathcal{A}s_B[0] = 11s_B[0] = 6 \\
k = 12: & w_B[0] = 1 & p_\mathcal{A}s_B[0] = 12s_B[0] = 5 \\
k = 13: & w_B[0] = 0 & p_\mathcal{A}s_B[0] = 13s_B[0] = 4 \\
k = 14: & w_B[0] = 0 & p_\mathcal{A}s_B[0] = 14s_B[0] = 3 \\
k = 15: & w_B[0] = 0 & p_\mathcal{A}s_B[0] = 15s_B[0] = 2 \\
k = 16: & w_B[0] = 0 & p_\mathcal{A}s_B[0] = 16s_B[0] = 1
\end{array}
$$

Since there are 2 changes in the signal values $w_B[0]$, the adversary $\mathcal{A}$ now knows that the value of $s_B[0] = \pm 1$. The adversary can determine all the other coefficients of s_B this way upto $\pm$ sign. Thus, the adversary extracts the value of all the coefficients upto $\pm$ sign as $s_B[0] = \pm 1$, $s_B[1] = 0$, $s_B[2] = 0$, $s_B[3] = \pm 2$.

Step 2: Repeat above step for public key $p_\mathcal{A} = (1 + x)p_\mathcal{A}$ of the adversary. This yields the coefficients as below:
$s_B[0] - s_B[3] = \pm 1$, $s_B[0] + s_B[1] = \pm 1$, $s_B[1] + s_B[2] = 0$, $s_B[2] + s_B[3] = \pm 2$

Step 3: Using $s_B[0] - s_B[3] = \pm 1$, $s_B[0] = \pm 1$ and $s_B[3] = \pm 2$, the adversary can determine that $s_B[0]$ and $s_B[3]$ are of opposite sign. So, if $s_B[0] = 1$, then $s_B[3] = -2$ and vice versa.

Step 4: Step 3 can be repeated for every pair of coefficients. But in this example, we have $s_B[1] = s_B[2] = 0$. So, this step is not required.

Thus, the adversary $\mathcal{A}$ recovers $s_B = (1, 0, 0, -2)$ or $s_B = (-1, 0, 0, 2)$ depending on the sign of $s_B[0]$. Now, choose $s_B[0] = 1$, then $s_B[3] = -2$ and compute $p_B - as_B = (2, -6, 1, -2) - (0, 5, -2, 2) = (2, 6, 3, -4)$. Since n is small in this toy example, we only have a small number of samples to verify the distribution of $p_B - as_B$ but since parameters proposed

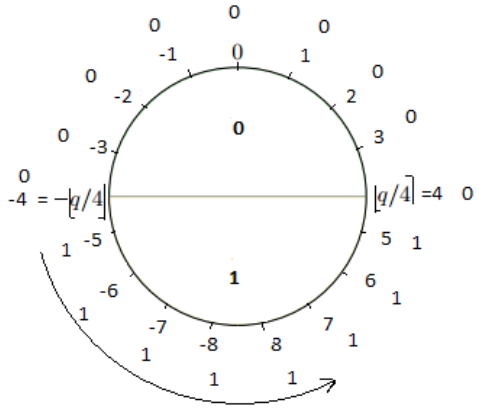

Fig. 2. Signal changes of $s_B[0]$ in the example. This figure shows the number of signal changes while looping over k values.

for the key exchange requires higher value of n, we can easily determine if the distribution follows a discrete gaussian from the samples. This example shows that from the values of $p_B - as_B$, the adversary can easily determine that $s_B[0] = -1$ and hence obtains the value of $s_B = (-1, 0, 0, 2)$ succeeding in the attack.

Fluctuations: For the same example, consider the case when we add g_B to k_B. Suppose $g_B^{(0)}$ values are $0, 1, 1, 0, -1, 1, 0, 0, -1, -1, -2, 1, 0, 0, 2, 0, 0$, sampled from the error distribution for k from 0 to $q - 1 = 16$.

$k = 0:$ $w_B[0] = 0$ $k_B[0] = 0$
$k = 1:$ $w_B[0] = 0$ $k_B[0] = 1$
$k = 2:$ $w_B[0] = 0$ $k_B[0] = 0$
$k = 3:$ $w_B[0] = 0$ $k_B[0] = -3$
$k = 4:$ $w_B[0] = 1$ $k_B[0] = -6$
$k = 5:$ $w_B[0] = 0$ $k_B[0] = -3$
$k = 6:$ $w_B[0] = 1$ $k_B[0] = -6$
$k = 7:$ $w_B[0] = 1$ $k_B[0] = -7$
$k = 8:$ $w_B[0] = 1$ $k_B[0] = 7$
$k = 9:$ $w_B[0] = 1$ $k_B[0] = 6$
$k = 10:$ $w_B[0] = 0$ $k_B[0] = 3$
$k = 11:$ $w_B[0] = 1$ $k_B[0] = 8$
$k = 12:$ $w_B[0] = 1$ $k_B[0] = 5$
$k = 13:$ $w_B[0] = 0$ $k_B[0] = 4$
$k = 14:$ $w_B[0] = 1$ $k_B[0] = 7$
$k = 15:$ $w_B[0] = 0$ $k_B[0] = 2$
$k = 16:$ $w_B[0] = 0$ $k_B[0] = 1$

We can see that when $k = 4, k_B[0] = -6$ which is near the boundary region of E and so we can see frequent changes (fluctuations) in signal for consecutive k values till $k_B[0]$ stabilizes starting with $k = 6$. We ignore these fluctuations as count of signal change and only count 1 signal change from $k = 3$ to $k = 6$. Similarly, we ignore the signal fluctuations corresponding to $k = 9$ to $k = 14$. Thus, the stabilized number of signal changes is 2 which helps to guess the value of $s_B[0]$ to be ± 1. It is difficult to identify the fluctuations for small q since there are not

enough values for the key k_B to stabilize but the parameters for the key exchange provide for a reasonable q for the attack to work.

6 Experiments

We have verified experimentally the number of signal changes for the coefficients of s_B is as mentioned above. The execution was performed in C++ with NTL library using an Windows 10 64 bit system equipped with a 2.40 GHz Intel(R) Core(TM) i7-4700MQ CPU and 8 GB RAM. The multiplication was performed without using any optimized algorithms like FFT.

Also, since only $ke_{\mathcal{A}}$ changes for each query, we can fix $as_{\mathcal{A}}$ for the attack execution in the extended case, thus saving on one multiplication for every query. For preliminary testing purposes, we used parameter values of $n = 1024$, $q = 2^14 + 1$, $\alpha = 3.197$. This choice of parameters is close to the values used in [4] implementation. The time taken for this choice of parameters for running q queries to recover all the coefficients upto sign is 3.8 hours, without optimization.

7 Conclusion

In this work, we have presented an attack on the RLWE key exchange and run experiments to verify the correctness of the attack in recovering the secret of a reused public key with $2q$ queries to the honest party. This is to show that when the public key is fixed for a long term, the signal of the resulting key leaks information about the secret and can be exploited by an active adversary, whose public key is not created according to the protocol. We believe that this attack can be adapted even to [4] which also uses an unauthenticated version of key exchange and a different reconciliation mechanism. It would also be useful to analyze if such an attack can be extended for authenticated version of RLWE key exchange.

Acknowledgment: Jintai Ding, Saraswathy RV and Xiaodong Lin would like to thank NSF for its partial support.

References

1. The transport layer security (tls) protocol version 1.3 (December 2015), https://tools.ietf.org/html/draft-ietf-tls-tls13-07
2. Experimenting with post-quantum cryptography (July 2016), https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html
3. The transport layer security (tls) protocol version 1.3 (May 2016), https://tools.ietf.org/html/draft-ietf-tls-tls13-13
4. Alkim, E., Ducas, L., Pppelmann, T., Schwabe, P.: Post-quantum key exchange - a new hope. Cryptology ePrint Archive, Report 2015/1092 (2015), http://eprint.iacr.org/2015/1092
5. Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi, S. (ed.) Advances in Cryptology – CRYPTO 2009, Lecture Notes in Computer Science, vol. 5677, pp. 595–618. Springer Berlin Heidelberg (2009)
6. Bos, J.W., Costello, C., Naehrig, M., Stebila, D.: Post-quantum key exchange for the tls protocol from the ring learning with errors problem. Cryptology ePrint Archive, Report 2014/599 (2014), http://eprint.iacr.org/2014/599
7. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. pp. 309–325. ACM (2012)
8. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-lwe and security for key dependent messages. In: Rogaway, P. (ed.) Advances in Cryptology – CRYPTO 2011, Lecture Notes in Computer Science, vol. 6841, pp. 505–524. Springer Berlin Heidelberg (2011)

9. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Trans. Inf. Theor. 22(6), 644–654 (Sep 2006), http://dx.doi.org/10.1109/TIT.1976.1055638
10. Fluhrer, S.: Cryptanalysis of ring-lwe based key exchange with key share reuse. Cryptology ePrint Archive, Report 2016/085 (2016), http://eprint.iacr.org/2016/085
11. Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Strongly secure authenticated key exchange from factoring, codes, and lattices. Cryptology ePrint Archive, Report 2012/211 (2012), http://eprint.iacr.org/2012/211
12. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the 40th annual ACM symposium on Theory of computing. pp. 197–206. STOC '08, ACM, New York, NY, USA (2008)
13. Gong, B., Zhao, Y.: Small field attack, and revisiting rlwe-based authenticated key exchange from eurocrypt'15. Cryptology ePrint Archive, Report 2016/913 (2016), http://eprint.iacr.org/2016/913
14. Jintai Ding, Xiang Xie, X.L.: A simple provably secure key exchange scheme based on the learning with errors problem. Cryptology ePrint Archive, Report 2012/688 (2012), http://eprint.iacr.org/
15. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) Advances in Cryptology – EUROCRYPT 2010, LNCS, vol. 6110, pp. 1–23. Springer Berlin / Heidelberg (2010)
16. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on gaussian measures. SIAM J. Comput. 37, 267–302 (April 2007)
17. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In: Proceedings of the 41st annual ACM symposium on Theory of computing. pp. 333–342. STOC '09, ACM, New York, NY, USA (2009)
18. Peikert, C.: Lattice cryptography for the internet. Cryptology ePrint Archive, Report 2014/070 (2014), http://eprint.iacr.org/2014/070
19. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing. pp. 84–93. STOC '05, ACM, New York, NY, USA (2005)
20. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. 26(5), 1484–1509 (Oct 1997), http://dx.doi.org/10.1137/S0097539795293172
21. Zhang, J., Zhang, Z., Ding, J., Snook, M., Dagdelen, Ö.: Authenticated key exchange from ideal lattices. In: Advances in Cryptology-EUROCRYPT 2015, pp. 719–751. Springer (2015)