Practical Randomized RLWE-Based Key Exchange Against Signal Leakage Attack

Xinwei Gao, Jintai Ding, Lin Li, and Jiqiang Liu

Abstract—Ring Learning With Errors (RLWE)-based key exchange is one of the most efficient and secure primitive for post-quantum cryptography. One common approach to achieve key exchange over RLWE is error reconciliation. Recently, an efficient attack against reconciliation-based RLWE key exchange protocols with reused keys was proposed. This attack can recover a long-term private key if a key pair is reused. We also know that in the real world, key reuse is commonly adopted in applications like the Transport Layer Security (TLS) protocol to improve performance. Directly motivated by this attack, we construct a new randomized RLWE-based key exchange protocol against this attack. Our lightweight approach incorporates an additional ephemeral public error term into key exchange materials, so that this attack no longer works. With the same attack, we practically show that the signal value of our protocol is indistinguishable from uniform random, therefore, this attack no longer works. We explain how the attack fails, present 200-bit classic and 80-bit quantum secure parameter choice, efficient implementations, comparisons and discussion. Benchmark shows our protocol is truly efficient and even faster than related vulnerable protocols.

Index Terms—Post-quantum, RLWE, key exchange, leakage, attack, implementation

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. Compared with other choices to realize key exchange (e.g., RSA), Diffie-Hellman key exchange is more popular because it is simpler and more importantly, can achieve forward security. In fact, in the latest draft of TLS 1.3, RSA is removed as key exchange algorithm option because it is "static". It cannot offer forward security like the Diffie-Hellman ephemeral (DHE) key exchange. If a long-term private key of RSA is leaked, an attacker can decrypt and recover all past communications. This stresses that Diffie-Hellman-like key exchange protocols are more attactive than KEM-based protocols.

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 supersingular elliptic curve-based. Lattice-based cryptography is regarded as the most promising one because constructions 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 Learning with Errors (LWE) problem [3] and Ring-LWE problem [4]. LWE and RLWE-based key exchange protocols were introduced in 2012 (denoted as DING12) [5]. This work gives LWE and RLWE variants of Diffie-Hellman key exchange. Following the idea of this work, various LWE and RLWE-based key exchange protocols have been designed and implemented, including [6], [7], [8], [9], [10], [11], [12].

Here, we briefly recall the notion of reconciliaition-based RLWE key exchange protocol: Let R_q be ring $Z_q[x]/(x^n+1)$ for integers q and n. χ denotes a noise distribution. In RLWE-based key exchange, Alice and Bob agree on same value over error-perturbed values. They share a global parameter $a \in R_q$. The public key of Alice and Bob has the form $as + 2e$ where s is secret key and e is an error term. s and e are sampled according to χ and their coefficients are small. Alice sends her public key $pk_a = as_a + 2e_a$ to Bob. Bob returns his public key $pk_b = as_b + 2e_b$ and signal w_b. Alice computes $k_a = pk_b s_a + 2e'_a$ which approximately equals Bob's $k_b = pk_a s_b + 2e'_b$ because the error terms are small. Alice and Bob can agree on the same key using w_b with an error reconciliation mechanism that can reconcile errors between k_a and k_b.

Recently, a practical attack against reconciliation-based RLWE key exchange protocol was proposed [13]. This attack shows that an adversary can practically recover the private key of the other party with multiple queries if public

- *X. Gao, L. Li, and J. Liu are with the Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, Beijing 100044, P.R.China. E-mail: xinwei.gao.7@yandex.com, {lilin, jqliu}@bjtu.edu.cn.*
- *J. Ding is with the Department of Mathematical Sciences, University of Cincinnati, Cincinnati, OH 45219. E-mail: jintai.ding@gmail.com.*

Manuscript received 14 May 2017; revised 27 Jan. 2018; accepted 19 Feb. 2018. Date of publication 21 Feb. 2018; date of current version 16 Oct. 2018. (Corresponding author: Jintai Ding and Lin Li.)
Recommended for acceptance by Ç. K. Koç, Z. Liu, and P. Longa.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2018.2808527

and private keys are reused. It is known that key reuse can improve performance in real-world deployment because key computations are rather expensive. In TLS v1.2, the resumption mode allows key reuse, and this reduces online computations significantly. In TLS v1.3 draft version 7 [14], an optimized 0-round-trip time (RTT) resumption mode was proposed. This gives TLS the ability to initiate secure connection without any round-trip overhead. Reusing public and private key pairs saves most key exchange computations and communication overheads. In fact, the vast majority of real-world TLS connections are established with resumption mode. If RLWE-based key exchange protocols that are vulnerable to key reuse attack are adopted in TLS with reused keys, the security of communication is compromised. The main motivation of this work is to present a practical defense approach against this attack.

In project open quantum safe (OQS) [15], they present a prototype implementation that integrates several RLWE-based key exchange protocols that are vulnerable to signal leakage attack into OpenSSL. Bos et al. [7] also integrate their protocol into TLS. Gao et al. [16] integrated an RLWE-based authenticated key exchange (AKE) into TLS as well.

1.1 Related Work

In 2015, Kirkwood et al. from National Security Agency (NSA) revealed an issue in reconciliation-based key exchange protocols [17]. They suggested that if a public and private key pair is reused, current reconciliation-based LWE- and RLWE-based key exchange protocols may suffer from an attack that can reveal a private key with multiple key exchange executions. This work is very important because key reuse is very common in real-world internet protocols, including TLS etc.

In 2016, Fluhrer gave cryptanalysis on RLWE-based key exchange protocols [18]. 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 2016, Ding et al. presented a systematic attack with two extensions [13]. They also showed that attack is indeed practical with an actual attack example on the RLWE-based key exchange. This work introduces attacks with detailed analysis and attack an instantiation of DING RLWE key exchange with parameter choice $n = 1{,}024$, $q = 2^{14} + 1$, $\sigma = 3.197$. The result shows that a private key can be recovered successfully in 3.8 hours. The attack can be further applied to other reconciliation-based RLWE key exchange protocols. These works stress the urgency to improve reconciliation-based key exchange protocols.

1.2 Contributions

Contributions of this work are summarized as follows:

First, we present a new RLWE-based key exchange protocol that can practically defend against the signal leakage attack proposed in [13]. Our key exchange protocol extends DING12 RLWE key exchange so that practically signal leakage attack does not work anymore. Our 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. We also give a heuristic justification to show that the same leakage attack in [13] does not work for our protocol. We note that our protocol is a lightweight and practical defense solution, where previous reconciliation-based RLWE key exchange protocols with a reused key are vulnerable to this attack. Security proofs in DING12 still hold for our protocol.

Second, we instantiate DING12 and our protocol with practical parameter choices and present portable C++ implementations. The classic and quantum security levels of our parameter choice are analyzed. Compared with directly related works that are vulnerable to key reuse attack ([7] and [9]), a benchmark test shows that our protocol and implementation are even more efficient. Runtime of the regular mode of our protocol outperforms several signal leakage-vulnerable RLWE key exchange protocols. When a key pair is reused, the runtime of the key reuse mode is improved even further with nearly 2x speed improvement over the regular mode. The communication cost of the key reuse mode is greatly reduced.

2 Preliminary

2.1 LWE- & RLWE-Based Key Exchange Protocols

In 2012, Ding et al. introduced the first LWE- and RLWE-based key exchange protocols [5]. This is the first work that gives practical and provable secure LWE- and RLWE-based key exchange protocols. DING12 can be regarded as LWE and RLWE variants 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 an RLWE-based AKE protocol (denoted as AKE15) [8]. AKE15 is an 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 2016, Bos et al. proposed a LWE key exchange that is a variant of the LWE version in DING12 (denoted as Frodo) [10]. 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 analogues of PAK and PPK [12]. Gao et al. presented a much optimized implementation of [12] and integration into TLS as post-quantum TLS ciphersuite [19]. Gao et al. introduced an RLWE variant of Secure Remote Password (SRP) protocol [20], which is an augmented PAKE protocol.

To sum up, there are two major approaches to realize key exchange using LWE and RLWE problems: reconciliation-based (using the signal to reconcile error) and encryption-based. DING12 [5], PKT14 [6], BCNS15 [7] and NewHope

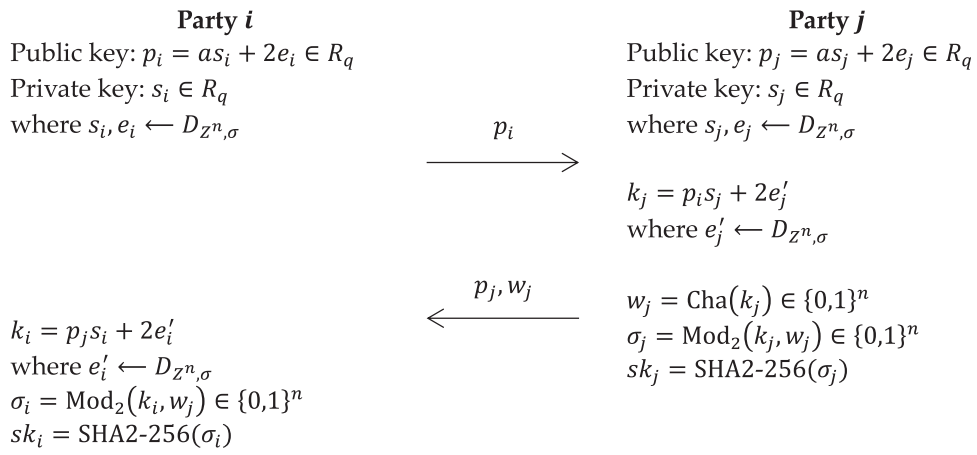


Fig. 1. DING12 RLWE-based key exchange protocol.

[10] are reconciliation-based unauthenticated key exchange protocols and signal leakage attack works for all these key exchange protocols. An important advantage of reconciliation-based protocols is reduced bandwidth. According to PKT14, they claimed that this approach could nearly halve the ciphertext size. This directly motivates our work—a practical solution for reconciliation-based RLWE key exchange protocol against leakage attack.

2.2 Revisit DING12 RLWE Key Exchange Protocol

Two key exchange protocols presented in [5] can be regarded as LWE and RLWE analogues of the classic Diffie-Hellman key exchange protocol. These protocols are quantum-resistant and proven secure. In this paper, our focus is the RLWE-based version. Because small error terms are involved, key computation is not rigorously equal to Diffie-Hellman. Two values used for key computation are approximately equal, therefore, a new error reconciliation algorithm was proposed. It can reconcile the error between two approximately equal values. First function—signal function generates a series of $0/1$ values (called signal) and denotes which region the key computation value belongs to. These $0/1$ bits are sent to the other side to reconcile errors using a reconciliation function. This is the core idea to realize Diffie-Hellman-like key exchange over LWE and RLWE. To ensure the correctness of the protocol, they also suggested a way to bound the norm of differences. By choosing prime q carefully, key exchange is successful with overwhelming probability. They also prove that their construction is secure against PPT adversaries. Publicly transmitted materials (public key and signal) cannot be distinguished from uniform random.

We first recall the state-of-the-art error reconciliation mechanism from DING12:

Signal Function. For prime $q > 2$, hint functions $\sigma_0(x),\ \sigma_1(x)$ from Z_q to $\{0,1\}$ are defined as:

$$\sigma_0(x) = \begin{cases} 1, & x \in [-\lfloor \frac{q}{4} \rfloor, \lfloor \frac{q}{4} \rfloor] \\ 1, & otherwise \end{cases};\ \sigma_1(x) = \begin{cases} 0, & x \in [-\lfloor \frac{q}{4} \rfloor + 1, \lfloor \frac{q}{4} \rfloor + 1] \\ 1, & otherwise. \end{cases}$$

Signal function $\mathrm{Cha}()$ is defined as: For any $y \in Z_q$, $\mathrm{Cha}\ (y) = \sigma_b\ (y)$, where $b \xleftarrow{\$} \{0,1\}$.

Robust Extractor. The robust extractor is defined as: $\mathrm{Mod}_2\ (x, w) = (x + w \cdot \frac{q-1}{2} \bmod q) \bmod 2$.

$\mathrm{Mod}_2()$ is a robust extractor on Z_q with error tolerance δ with respect to signal function, if the following holds:

- The deterministic algorithm $\mathrm{Mod}_2()$ takes input $x \in Z_q$ and signal w, outputs $k = \mathrm{Mod}_2(x,\ w) \in \{0,1\}$.
- $\mathrm{Cha}()$ takes an input $y \in Z_q$ and outputs signal $w \leftarrow \mathrm{Cha}(y) \in \{0,1\}$.

For any $x,\ y \in Z_q$ such that $x - y$ is even and $|x - y| \le \delta$, then it holds that $\mathrm{Mod}_2\ (x,\ w) = \mathrm{Mod}_2(y,\ w)$, where $w \leftarrow \mathrm{Cha}(y)$.

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

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

For details, please refer to [5].

Fig. 1 recalls DING12 RLWE key exchange protocol.

2.3 Attacking Reconciliation-Based RLWE Key Exchange

In this section, we briefly recall the attack in [13]. This practical and effective attack can recover user's private key within multiple queries if a key pair is reused. In step 1 of the attack, an adversary chooses secret key s_{adv} to be 0 and error term $e_{adv} = 1$ in R_q, therefore public key $p_{adv} = as_{adv} + k\ e_{adv} = ke_{adv}$. By doing key exchange with party j with k looping from 0 to $q - 1$, adversary can guess the value of $s_j[i]$ (secret key of party j) based on the number of times that signal $w_j[i]$ changes. Because of the definition of signal function, it is clear to see that the signal value flips when ke_{adv} enters or exits inner region $[-q/4, q/4]$. For k loops from 0 to $q - 1$, signal value flips exactly $2s_j[i]$ times, therefore the adversary can find the absolute value of $s_j[i]$ without knowing the sign. In step 2, the adversary sends $(1 + x)p_{adv}$ to party j. By looping k from 0 to $q - 1$ once again, the adversary can get the absolute value of $s_j[0] - s_j[n-1], s_j[1] + s_j[2], s_j[2] + s_j[3], \ldots, s_j[n-2] + s_j[n-1]$ without knowing the actual signs. In steps 3 and 4, the adversary can reveal the relationship between the signs of neighboring coefficients using all information from previous steps. In step 5, the adversary verifies his guess on signs by checking the distribution of $p_j - as_j$. If it follows the distribution of e_j (normally is discrete Gaussian distribution), then the guess is correct and the secret key is successfully recovered. If not, then flip the signs and obtain the correct s_j signs.

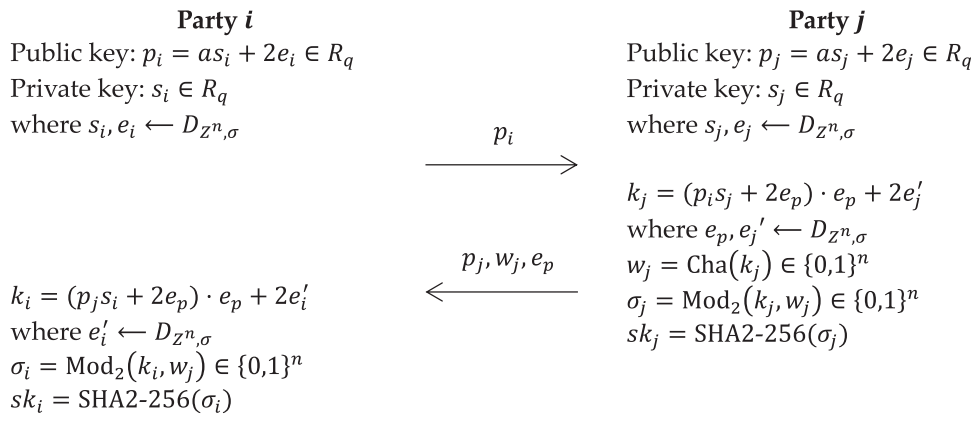


Fig. 2. Regular mode.

The above attack does not take account of additional errors $2e_i'$ and $2e_j'$ defined in Section 4.1 of the DING12 paper, but this is not an issue. When adding $2e_i'$ or $2e_j'$, 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. They suggest that one approach to defend this attack is to verify whether p_{adv} is a constant polynomial in R_q. However, the attack still works if p_{adv} is chosen to be $p_{adv} = as_{adv} + ke_{adv}$ where $e_{adv} = 1$, k starts looping with value $as_{adv}s_j[i]$. The adversary only needs to sample s_{adv} 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 party j. Note that this attack can be adapted to other reconciliation-based RLWE key exchange protocols (e.g., BCNS15 and NewHope) with modifications.

3 Practical Randomized RLWE-Based Key Exchange

In this section, we present our practical randomized RLWE-based key exchange protocol. The main objective of our protocol is to stop the signal leakage attack in [13] practically.

Our protocol has two modes: regular mode and key reuse mode. These two modes share the same structure Key reuse mode is a simplified version of regular mode, which reduces computation and communication costs with reused keys. Our protocol design closely follows the original DING12 RLWE-based key exchange protocol. We introduce protocol design of two modes, objective and preferred application scenario of each mode in Sections 3.1 and 3.2, analysis and explanations on how our protocol can stop a signal leakage attack practically in Section 3.3.

3.1 Regular Mode

Regular mode is designed for common key exchange between two parties without reused keys. In this mode, two parties exchange their public key, signal w_j and a one-time public error e_p to agree on the same session key. There are several preferable scenarios for this mode: (1) The two parties have never communicated before. Party i does not share party j'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.

Fig. 2 presents the regular mode of our randomized RLWE key exchange protocol.

Compared with DING12, we add an additional one-time public error term e_p generated by party j. This gives unpredictable and uncontrollable randomization. e_p is the core of our design. In each key exchange execution, the two parties should honestly sample secret error terms e, e' and party j should sample fresh and nonreusable public error term e_p.

Because $p_j = as_j + 2e_j$ and it cannot be distinguished from uniform random, $k_i = p_j e_p s_i + 2e_p^2 + 2e_i'$, we can take $p_j e_p$ as the uniform random element a in a regular RLWE sample because e_p is sampled from the same $D_{Z^n,\sigma}$ with small standard deviation, p_j is indistinguishable from uniform random. Note that $2e_i'$ is not a public term, therefore k_i has the same shape as the RLWE sample: a uniform random element multiplies a small secret error term, and then adds another small secret error term. $2e_p^2$ is small and public, therefore it does not matter. An adversary cannot manipulate or utilize e_p to recover s_i because k_i has the same shape as the RLWE sample. Same analysis also applies to k_j. Therefore k_i and k_j are indistinguishable from uniform random.

The correctness of our protocol can be guaranteed because $k_i \approx k_j$, $k_i = (p_j s_i + 2e_p) \cdot e_p + 2e_i' = ((as_j + 2e_j) \cdot s_i + 2e_p \cdot e_p + 2e_i' = as_i s_j e_p + 2s_i e_j e_p + 2e_p^2 + 2e_i'$, $k_j = (p_i s_j + 2e_p) \cdot e_p + 2e_j' = ((as_i + 2e_i) \cdot s_j + 2e_p) \cdot e_p + 2e_j' = as_i s_j e_p + 2s_j e_i e_p + 2e_p^2 + 2e_j'$. The difference between k_i and k_j is $k_i - k_j = 2s_i e_j e_p + 2e_i' - 2s_j e_i e_p - 2e_j' = 2e_p(s_i e_j - s_j e_i) + 2(e_i' - e_j')$. Because $s_i, s_j, e_i, e_j, e_i', e_j'$ and e_p are sampled from a discrete Gaussian distribution with certain standard deviation, the norm of $k_i - k_j$ can be bounded tightly using the same approach as DING12.

For RLWE key exchange, major computation cost are sampling, Number Theory Transform (NTT), Inverse-NTT and polynomial multiplication operations. The count of these operations of the regular mode is given in Table 1.

TABLE 1
Major Time-Consuming Operations for Regular Mode

	Party i	Party j
Sampling	3	4
Number theoretic transform (NTT)	2	3
Inv-NTT (Inverse-NTT)	1	1
Multiplication	3	3

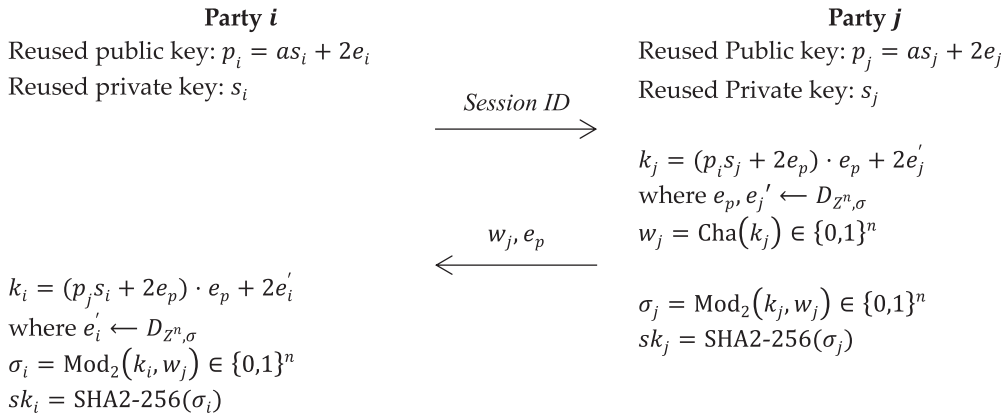


Fig. 3. Key reuse mode.

Compared with DING12 RLWE key exchange, party j gains additional one sampling, one NTT and one multiplication computation due to additional public error term e_p. Party i does not gain additional sampling and NTT operations because e_p is received from party j. This will lead to slower performance for party j.

3.2 Key Reuse Mode

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.

Fig. 3 presents the key reuse mode of our randomized RLWE key exchange protocol.

The key reuse mode shares the same protocol structure as the regular mode, except both parties reuse a key pair. Party i needs to send a 256-bit session id to party j, notifying party j that he wants to reuse the key from a previous session. Party j retrieves keys of party i from the database, generates a fresh public error e_p, return signal w_j and e_p to party i to generate a fresh session key. The count of major time-consuming operations is given in Table 2.

Compared with the regular mode, party i reduces two sampling, two NTT and one polynomial multiplication operations. Party j reduces one sampling, two NTT and one polynomial multiplication operations. This improves performance and reduces communication cost.

3.3 How Does Attack and Our Defense Work?

We first analyze the attack in [13] and observe what leads to the failure of current reconciliation-based key exchange protocols. In this attack, an adversary can capture all publicly transmitted messages, including p_j, w_j. The adversary also can manipulate the message he sends, i.e., p_{adv}. Except for carefully chosen p_{adv}, s_{adv} and e_{adv}, the adversary executes the protocol faithfully. Party j executes the protocol faithfully with the reused key pair.

As pointed out in [13], the adversary can choose $e_{adv} = 1$ and challenge party j with $p_{adv} = as_{adv} + ke_{adv}$, $k = 0, 1, \ldots, q-1$. The attack works mainly due to the construction of $k_j = as_{adv}s_j + ke_{adv}s_j + 2e'_{adv}$. When the key is reused, s_{adv} and s_j are fixed. By looping k from 0 to $q-1$, only $ke_{adv}s_j$ changes. Notice that $e_{adv} = 1$, $s_{adv} = 0$, this leads to a linear increase of k_j. If the signal value flips from 0 to 1 or 1 to 0 at some point, this implies that the value of k_j exits or enters the inner region $[-q/4, q/4]$. There are exactly $2s_j[i]$ times of signal changes for the i-th coefficient of $s_j[i]$. With this observation, the adversary can figure out the exact value of s_j within $2q$ queries with party j. Despite BCNS15 and NewHope adopting somewhat different reconciliation mechanisms, they also use the signal to imply which region the value of k_j belongs to, therefore the idea of this attack works for them as well.

From the above analysis, we know that fixed values a, s_i and s_j, combined with a smart attack forces the signal to reveal more information than it should be and lead to the failure of these protocols against this attack. Our approach to defend this attack is straightforward: adding additional fresh randomization to k_i and k_j using one-time public error e_p. Now, $k_j = (p_i s_j + 2e_p) \cdot e_p + 2e'_j = ((as_i + 2e_i)s_j + 2e_p) \cdot e_p + 2e'_j = as_i s_j e_p + 2s_j e_i e_p + 2e_p^2 + 2e'_j$, $k_i = (p_j s_i + 2e_p) \cdot e_p + 2e'_i = ((as_j + 2e_j)s_i + 2e_p) \cdot e_p + 2e'_i = as_i s_j e_p + 2s_i e_j e_p + 2e_p^2 + 2e'_i$. In each execution of our protocol, party j must generate a fresh e_p. Now, the signal value of k_j shows which region $as_i s_j e_p + 2s_j e_i e_p + 2e_p^2 + 2e'_j$ lies in, instead of $as_i s_j + 2e_i s_j + 2e_j'$. Note that e_p is out of the adversary's control and it randomizes and masks the property of s_j with rather low cost.

Let us observe how our approach defends this attack more intuitively. Assume we are attacking DING12 with reused keys. An adversary will receive the following pairs from the first round of attack:

$$
\begin{aligned}
k &= 0, \mathrm{Cha}(as_{adv}s_j);\\
k &= 1, \mathrm{Cha}(as_{adv}s_j + s_j);\\
k &= 2, \mathrm{Cha}(as_{adv}s_j + 2s_j);\\
&\cdots\\
k &= q-2, \mathrm{Cha}(as_{adv}s_j + (\boldsymbol{q-2})s_j);\\
k &= q-1, \mathrm{Cha}(as_{adv}s_j + (\boldsymbol{q-1})s_j).
\end{aligned}
$$

TABLE 2
Major Time-Consuming Operations for Key Reuse Mode

	Party i	Party j
Sampling	1	2
NTT	0	1
Inverse-NTT	1	1
Polynomial multiplication	2	2

Only bold items are different for each execution. It is clear that with k looping from 0 to $q-1$, s_j and $as_{adv}s_j$ are fixed, the value of $as_{adv}s_j + ks_j$ is increased linearly, therefore this leads to flips of signal value with a clear pattern (00...0011...1100...) so that the attack is successful. One practical example is shown in Section 5 of [13].

Here, we observe our defense approach: suppose that the key pair of party j is reused, our protocol is honestly executed and the adversary applies same the attack, the adversary will receive the following pairs from the first round of attack:

$$
\begin{aligned}
k &= 0, \mathrm{Cha}\Big(\boldsymbol{as_{adv}s_je_{p_0}} + 2e^2_{p_0} + 2e'_{j_0}\Big)\boldsymbol{e_{p_0}}\\
k &= 1, \mathrm{Cha}\Big(\boldsymbol{as_{adv}s_je_{p_1}} + \boldsymbol{s_je_{p_1}} + 2e^2_{p_1} + 2e'_{j_1}\Big), \boldsymbol{e_{p_1}};\\
k &= 2, \mathrm{Cha}\Big(\boldsymbol{as_{adv}s_je_{p_2}} + 2\boldsymbol{s_je_{p_2}} + 2e^2_{p_2} + 2e'_{j_2}\Big), \boldsymbol{e_{p_2}};\\
&\cdots\\
k &= q-2, \mathrm{Cha}\Big(\boldsymbol{as_{adv}s_je_{p_{q-2}}} + (\boldsymbol{q}-2)\boldsymbol{s_je_{p_{q-2}}} + 2e^2_{p_{q-2}} + 2e'_{j_{q-2}}\Big),\\
&\quad \boldsymbol{e_{p_{q-2}}};\\
k &= q-1, \mathrm{Cha}\Big(\boldsymbol{as_{adv}s_je_{p_{q-1}}} + (\boldsymbol{q}-1)\boldsymbol{s_je_{p_{q-1}}} + 2e^2_{p_{q-1}} + 2e'_{j_{q-1}}\Big),\\
&\quad \boldsymbol{e_{p_{q-1}}}.
\end{aligned}
$$

Bold items are randomized and different for each execution. Because e_p is sampled from $D_{Z^n,\sigma}$ and it is fresh for each k, e_p masks and randomizes the value of $as_{adv}s_j + ks_j$, therefore the adversary cannot manipulate the value of k_j and derive nonuniform random signal values with linear increased k. Now, $\mathrm{Cha}()$ reveals which region $as_{adv}s_je_p + ks_je_p + 2e^2_p + 2e'_j$ belongs to. The adversary cannot discover the property of s_j even if looping k from 0 to $q-1$ and observing how many times the value of $\mathrm{Cha}()$ changes due to randomization from e_p. This make the adversary's attempt to recover s_j with the same attack ineffective. Moreover, e_p is chosen by the same side that computes signal value, the adversary cannot manipulate e_p with the value he desires to initiate attack because e_p and signal value are computed by party j. Finally, $as_is_je_p + 2s_je_ie_p + 2e^2_p + 2e'_j$ causes even more trouble for the adversary because $2s_je_ie_p + 2e^2_p + 2e'_j$ invokes much larger fluctuations than $2s_je_i + 2e'_j$ in the original protocol. This is not as important as randomization caused by e_p on as_is_j because an adversary can no longer recover the property of s_j, not to mention larger fluctuations.

3.4 Randomness Test on Signal Value

From the above analysis, we know that an attack causes the distribution of signal value to be nonuniformly random. A typical example of a signal value of DING12 protocol with reused key combined with this attack may look like 00...0011...1100...0011.... Apparently, this is not uniformly random. However, it is required that the signal be indistinguishable from uniformly random bits to ensure the security of the protocol.

To demonstrate that the signal value of our protocol is indeed indistinguishable from uniform random, we ran a randomness test on the signal value from our protocol underthe same leakage attack. Randomness is measured by NIST's statistical test suite for random numbers (NIST SP 800-22, Revision 1a) [21]. This standard provides a set of statistical tests for evaluating deterministic and non-deterministic random number generators. NIST's suite contains 15 tests that target arbitrarily long binary sequences to evaluate randomness.

TABLE 3
Results of NIST's Randomness Test Suite

Test Title	Mean	Output of confidence function
Frequency (Monobit) Test	0.499908	0.002323163
Frequency Test within a Block	0.499445	0.002318156
Runs Test	0.499179	0.002322441
Binary Matrix Rank Test	0.490403	0.002008774
Discrete Fourier Transform (Spectral) Test	0.488831	0.002313658
Nonoverlapping Template Matching Test	0.604813	0.002598904
Linear Complexity Test	0.677002	0.002855694
Approximate Entropy Test	0.997078	5.00E-05
Cumulative Sums (Cusums) Test	0.509738	0.002342064
Cumulative Sums Reverse Test	0.509736	0.002342551

The experiment is designed and implemented as follows: We first randomly generate a key pair and fix it for each execution of our test. An adversary applies the same attack to our new protocol as before. We run our protocol and execute the attack for 1,000,000 rounds. In each round, k is uniformly generated between 0 and $q-1$. We record all 1,000,000 signal value bit strings from each round. We test the randomness of each 1024-bit signal bit string from 1,000,000 rounds and compute the mean value of results of all randomness tests. We also compute a confidence function with significance level to be $1/2^{50}$ (i.e., confidence level $= 1 - 1/2^{50}$) to show our result is very stable and reliable. NIST's randomness test suite we use is available at https://gerhardt.ch/random.php.

We present evaluation results in Table 3.

The result of "Serial Test": Mean: [0.988727494, 0.972356834], output of confidence function: [0.000302446, 0.000848657]. Result of "Random Excursions Test": Mean: [0.525727259, 0.512163243, 0.505589151, 0.530324303, 0.530807662, 0.505328364, 0.512085556, 0.526170653], output of confidence function: [0.002753355, 0.002580277, 0.002415705, 0.002307506, 0.002309005, 0.002418534, 0.002579932, 0.002751813]. Result of "Random Excursions Variant Test": Mean: [0.557833269, 0.552352021, 0.546790105, 0.541084283, 0.535514883, 0.529585122, 0.523319032, 0.516735328, 0.510506562, 0.510242218, 0.516783563, 0.523111905, 0.529327059, 0.535071314, 0.540538897, 0.546291842, 0.55189773, 0.557576803], output of confidence function: [0.002031106, 0.002058113, 0.002084945, 0.002111785, 0.002138426, 0.002167919, 0.002199825, 0.002235446, 0.002303776, 0.00230122, 0.002235963, 0.002198592, 0.002169171, 0.002139191, 0.002110441, 0.002083984, 0.002057557, 0.002031193].

Judging from criteria of each randomness test in [21], all randomness tests on 1,000,000 signal bit strings of our protocol have passed. The value of confidence function proves that the result is very stable. This practically proves that the signal value of our protocol is truly uniform random with reused keys and the same attack, therefore the attack does not work for our protocol.

4 Instantiation and Implementation

In this section, we first choose a practical parameter for DING12 and our protocol, then we analyze classic and quantum security levels of our parameter choice. We also present efficient portable C++ implementation and report performance comparison on DING12, two modes of our protocol, BCNS15 and NewHope. The communication cost of these protocols is also discussed. The results show that our implementation is indeed practical.

4.1 Parameter Choice

Here, we choose the same parameter for DING12 and our protocol. We choose similar parameters for BCNS15 to present a relatively fair comparison: discrete Gaussian distribution for sampling with standard deviation $\sigma = 8/\sqrt{2\pi} \approx 3.192$, degree $n = 1024$ and prime $q = 1073479681$ (30 bits). We safely set statistical distance between the sampled distribution and discrete Gaussian distribution to be 2^{-128} to preserve high statistical quality and security. We remark that $\sigma \approx 3.192$ and $n = 1024$ are identical to BCNS15, prime q is 2-bits smaller.

We also analyze classic and quantum security levels of our parameter choice. For classic security, we use the same approach as BCNS15, which adopt LWE estimator in [22] to evaluate. LWE estimator gives a thorough security estimation for both LWE and RLWE-based cryptosystems. It evaluates the security level of cryptosystems by computing attack complexity of exhaustive search, BKW, lattice reduction, decoding, reducing BDD to unique-SVP and meet-in-the-middle attacks. Given any parameters, the LWE estimator outputs computation and space complexity of these attacks, therefore it gives a nice security estimation for LWE and RLWE-based cryptosystems.

Instructions for using the LWE estimator to estimate classic security level are presented as follows:

```
> load("https://bitbucket.org/malb/lwe-estimator/raw/HEAD/estimator.py")
> n, alpha, q = 1024, alphaf(8, 1073479681), 1073479681
> set_verbose(1)
> _ = estimate_lwe(n, alpha, q, skip = ["arora-gb"])
```

Our parameters offer at least 200-bit classic security.

For quantum security estimation, NewHope presented an analysis that estimates the complexity of primal and dual attack for solving underlying lattice problem on a quantum computer. They claimed that this estimation is pessimistic because they only considered core SVP hardness, therefore actual complexity estimation result should be higher than the result from estimation in NewHope. The result shows that our parameter choice offers >80-bit quantum security.

4.2 Implementation and Performance

We present portable C++ implementations of DING12 and our improved protocol. We first implement DING12 and modify it to adapt to our protocol. This might be the first work that instantiates DING12 practically and provided an efficient C++ implementation.

We implement major RLWE-related computations with NFLlib library [23]. NFLlib is an efficient NTT-based library for ideal lattice cryptography. This library includes various algorithms and programming optimizations to achieve high performance. It also provides portable and AVX2-optimized implementations. Routines including NTT, inverse-NTT and sampling from discrete Gaussian distribution are very efficient. Note that NTT and inverse NTT implementation benefit from optimizations using SSE instruction sets and NFLlib is not a const-time implementation. For discrete Gaussian sampling, we safely set statistical distance from sampled distribution to discrete Gaussian distribution as 2^{-128} to preserve high statistical quality and security. Each coefficient of public and private key, error terms, key exchange material k_i and k_j are represented in a "uint32_t" type variable. For one-time public error term e_p, coefficients are represented in a "uint8_t" type variable.

TABLE 4
Performance of DING12, This Work, BCNS15 and NewHope

	Party i (ms)	Party j (ms)
DING12	0.0703	0.0705
This work - regular mode	0.0728 $(1.04x)$	0.0904 $(1.28x)$
This work - key reuse mode	0.0328 $(0.47x)$	0.0495 $(0.70x)$
BCNS15	0.702 $(9.99x)$	0.867 $(12.30x)$
NewHope	0.0838 $(1.19x)$	0.0957 $(1.36x)$

We remark that our portable C++ implementation does not take advantage of new instruction sets (e.g., AVX2) to achieve high performance. In fact, the CPU we use to test the performance of all implementations does not support AVX2, but the efficiency of NTT and inverse NTT enjoys optimization techniques in NFLlib using SSE instruction sets.

We test implementations including DING12, two modes of our protocol and compare with BCNS15 and NewHope on the same server equipped with 3.4 GHz Intel Xeon E5-2687W v2 processor and 64 GB memory. The server runs 64-bit Ubuntu 14.04. Implementations of DING12 and our protocol are all compiled by g++ version 4.8.4 with '-O3 -fomit-frame-pointer -march = native -m64' flags to get maximum performance out of the code.

For the implementation of BCNS15 and NewHope, we use liboqs, which is an open-source C library for quantum-resistant cryptographic algorithms [15]. For BCNS15, we tested nonconst time version implementation because NFLlib is not const time implementation (-DCONSTANT_-TIME flag in Makefile is disabled). For NewHope, we chose nonconst time portable C implementation. We use the latest version of liboqs to measure performance. Note that BCNS15 and NewHope implementations do not have handcrafted SSE instruction sets optimizations like NFLlib, therefore, the performance of our implementation benefits from optimizations in NFLlib. We compile liboqs using same optimization flags as our implementation using gcc compiler version 4.8.4 and measure the performance with the "./test_kex –bench" command.

We report the average runtime of DING12, two modes of our protocol with 50,000 executions, benchmark of BCNS15 and NewHope from liboqs library command line in Table 4. All implementations only run on a single core and do not utilize parallel computing techniques.

The number in parentheses is the number of times of runtime with performance of DING12 as the baseline. Regular mode of our protocol has very similar performance to DING12 for party i, but party j is slower due to additional

TABLE 5
CPU Cycles and Runtime of Major Computation Operations

	Runtime (ms)	CPU cycles
Sampling	0.0074	27415
NTT	0.0104	37045
Inverse-NTT	0.0113	39109
Error reconciliation	0.0072	26806
Key Generation	0.0472	161696

TABLE 6
Communication Costs of DING12, This Work, BCNS15 and NewHope

	To party j (KB)	To party i (KB)
DING12	3.75	3.875
This work -regular mode	3.75	4.875
This work -key reuse mode	0.03125	1.125
BCNS15	4	4.125
NewHope	1.78125	2

sampling and NTT computation on e_p. Compared with regular mode, key reuse mode saves nearly half of total runtime. Performance of our protocol is better than BCNS15 and NewHope, which proves that our implementation is very efficient. Performance on error sampling and polynomial multiplication of NFLlib is 18.11x and 19.27x faster than BCNS15 due to more efficient implementation and optimization from SSE instruction sets. Because our parameter choice is very close to BCNS15, therefore the benchmark is relatively comparable. NewHope's benchmark is provided as a reference. We believe that if these protocols are implemented using the same library, we can expect very similar performance. A more detailed comparison analysis between DING12 and BCNS15 is given in [24].

NewHope also provides an AVX2-optimized implementation, which gives a significant performance boost over their portable implementation. The overall performance of key exchange for AVX2-optimized implementation is nearly 4x faster than their portable implementation and 21x faster than BCNS15. Note that BCNS15 does not provide an AVX2-optimized implementation. Detailed performance summary chart please refer to Table 2 of [9].

We also report CPU cycles and runtime of major computation operations, including discrete Gaussian sampling ($n = 1024, \sigma = 3.192$), NTT, Inverse NTT, error reconciliation and key generation. We tested the performance of these operations based on DING12 implementation and report results over 50,000 executions in Table 5.

4.3 Communication Cost

The communication costs of DING12, two modes of our protocol, BCNS15 and NewHope are reported in Table 6.

Because of the public error e_p, the communication cost of regular mode of our protocol is larger than DING12. e_p increases 1 KB communication cost. When the key is reused, the client only need to send a 256-bit session ID to party j and party j responds with a 1 KB fresh public error e_p and 0.125 KB signal. Compared with the regular mode, the key reuse mode decreases communication cost and total runtime significantly.

Our parameter choices for DING12 and our protocol lead to higher communication cost than NewHope because we use a larger prime. In the NewHope implementation, they choose $q = 12289$ which is approximately 14 bits. Our prime is 30 bits with the same $n = 1024$ as NewHope. Larger prime directly leads to a larger key size. Our implementation of DING12 and the new protocol work are very efficient, but the communication cost is larger than NewHope. Compared with BCNS15, they also choose $n = 1024$ but they choose a slightly larger modulus $p = 2^{32} - 1$. Similarly, BCNS15's larger modulus leads to larger communication cost than DING12.

5 Discussion

We remark that our protocol is a practical and lightweight approach to a signal leakage attack. Kirkwood et al. suggested that a Fujisaki-Okamoto transformation [25] might be extended to a variant for key exchange to defeat this attack [17]. Although this approach can be theoretically proven secure, it is considered to be more expensive regarding computation.

For the attack and our defense approach, we can see that the fundamental issue that caused this attack is the inherent property of the signal value. The success of this attack is a typical example of utilizing the property of signal function cleverly. Beause the signal value reveals that the value is in the inner or outer region, it always more or less reveals some information of k_j. This is the reason why we remark that this work is a practical solution to make such attack no longer effective. We cannot theoretically prove that the signal in our protocol does not leak any information about the secret key and we do not claim formal active security for our protocol, but with our defense approach, analysis and heuristic justification, we practically show that the attack in [13] fails in practice. A zero knowledge-based construction might be able to claim provably secure key reuse. Our 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 etc.

We remark that passive security proofs in DING12 still stand. We use the same definition, structure, error reconciliation mechanism and approaches as DING12, where they proved that their protocol is secure against a passive PPT adversary. An adversary cannot distinguish the final shared key with uniformly random values even if the adversary gets transcripts of the protocol. They also proved that for the error reconciliation mechanism, output of signal function $\mathrm{Cha}()$ and $\mathrm{Mod}_2()$ are uniformly random. For our protocol, major improvement comes from one-time public error. Construction of key material (k_i and k_j) follows the original design of DING12. Coefficients of k_j in our protocol and DING12 are uniformly random, all other premises for security proof in DING12 are satisfied. Experiments in Section 3.4 practically proves that even with signal leakage attack, the signal value of our protocol is uniformly random.

Our protocol is an attempt toward defending against this attack. Because the attack is effective against all

reconciliation-based key exchange protocols, it is vital and urgent to design new protocols that can defeat this attack with rigorous security proofs.

We remark that there are provably secure and efficient KEM-based RLWE key exchange candidates. The CCA-secure KEM "Kyber" in [26] can achieve secure key reuse and enjoys various very nice security properties. However, one disadvantage of such a construction that it is much slower than a reconciliation-based approach. Decryption in Kyber is 6.5x slower than NewHope for AVX2 optimized implementation. Because users are more likely to have devices with limited computing power, this will cause more latency in real-world applications. Another concern for the KEM-based approach is forward security. In the latest draft of TLS 1.3, RSA is removed from key exchange candidates for this reason. Similarly, if RLWE-based KEM is integrated and a secret key of the long-term public key is leaked, an adversary can recover and decrypt all past communication data. For reconciliation-based key exchange, it can work very similarly as current DHE/ECDHE key exchange in TLS. Our work is only a small step toward key reuse, and provably secure reconciliation-based construction is an important future work.

6 Conclusion

In this paper, we present a practical and lightweight RLWE-based key exchange protocol that can defend against the signal leakage attack in [13]. Compared with DING12, BCNS15 and NewHope, our protocol is a practical solution to the key reuse issue for reconciliation-based RLWE key exchange protocols with little additional computational overhead, where these three protocols are all vulnerable to attack in [13]. Regular mode is designed for regular key exchange and key reuse mode allows both parties to reuse a key pair. Communication and computation overheads are reduced in key reuse mode. We explain in detail how we use an additional fresh public error term to make this attack no longer work. By testing the randomness of signal values of our protocol, we practically show that our protocol remains strong against signal leakage attack. We also choose practical parameters that offer at least 200-bit classic and 80-bit quantum security for DING12 and our protocol. Finally, we present an efficient and portable C++ implementation of DING12 and two modes of our protocol. A benchmark shows that our protocol and implementation are indeed practical, providing similar or even better performance than BCNS15 and NewHope.

Acknowledgments

This work is funded by the China Scholarship Council and National Natural Science Foundation of China (Grant No. 61402035). We also thank reviewers for reading early versions of this paper and their valuable comments.

References

[1] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory*, vol. 22, no. 6, pp. 644–654, Nov. 1976.

[2] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annu. Symp. Found. Comput. Sci.*, 1994, pp. 124–134.

[3] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *J. ACM*, vol. 56, no. 6, 2009, Art. no. 34.

[4] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2010, pp. 1–23.

[5] J. Ding, X. Xie, and X. Lin, "A simple provably secure key exchange scheme based on the learning with errors problem," Cryptology ePrint Archive, Rep. 2012/688, http://eprint.iacr.org/2012/688, 2012.

[6] C. Peikert, "Lattice cryptography for the internet," in *Proc. Int. Workshop Post-Quantum Cryptography*, 2014, pp. 197–219.

[7] J. W. Bos, C. Costello, M. Naehrig, and D. Stebila, "Post-quantum key exchange for the TLS protocol from the ring learning with errors problem," in *Proc. IEEE Symp. Security Privacy*, May 2015, pp. 553–570.

[8] J. Zhang, Z. Zhang, J. Ding, M. Snook, and Ö. Dagdelen, "Authenticated key exchange from ideal lattices," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Tech.*, 2015, pp. 719–751.

[9] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange-a new hope," Cryptology ePrint Archive, Rep. 2015/1092, 2015. [Online]. Available: http://eprint.iacr.org/2015/1092

[10] J. Bos, et al., "Frodo: Take off the ring! practical, quantum-secure key exchange from LWE," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2016, pp. 1006–1018.

[11] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "New hope without reconciliation," Cryptology ePrint Archive, Rep. 2016/1157, 2016. [Online]. Available: http://eprint.iacr.org/2016/1157

[12] J. Ding, S. Alsayigh, J. Lancrenon, R. V. Saraswathy, and M. Snook, "Provably secure password authenticated key exchange based on RLWE for the post-quantum world," in *Proc. Cryptographers' Track RSA Conf.*, 2017, pp. 183–204.

[13] J. Ding, S. Alsayigh, R. V. Saraswathy, S. Fluhrer, and X. Lin, "Leakage of signal function with reused keys in RLWE key exchange," in *Proc. IEEE Int. Conf. Commun.*, May 2017, pp. 1–6.

[14] E. Rescorla, "The transport layer security (TLS) protocol version 1.3," [Online]. Available: https://tools.ietf.org/html/draft-ietf-tls-tls13-07, Accessed on: Feb. 10, 2017.

[15] D. Stebila and M. Mosca, "Post-quantum key exchange for the internet and the open quantum safe project," Cryptology ePrint Archive, Rep. 2016/1017, 2016. [Online]. Available: http://eprint.iacr.org/2016/1017

[16] X. Gao, L. Li, J. Ding, J. Liu, R. V. Saraswathy, and Z. Liu, "Fast discretized Gaussian sampling and post-quantum TLS ciphersuite," in *Proc. Int. Conf. Inf. Security Practice Experience*, 2017, pp. 551–565.

[17] D. Kirkwood, B. C. Lackey, J. McVey, M. Motley, J. A. Solinas, and D. Tuller, "Failure is not an option: Standardization issues for post-quantum key agreement," in *Proc. Talk NIST Workshop Cybersecurity Post-Quantum World*, 2015. [Online]. Available: http://www.nist.gov/itl/csd/ct/post-quantum-crypto-workshop-2015.cfm

[18] S. R. Fluhrer, "Cryptanalysis of ring-LWE based key exchange with key share reuse," Cryptology ePrint Archive, Rep. 2016/85, 2016. [Online]. Available: http://eprint.iacr.org/2016/85

[19] X. Gao, J. Ding, L. Li, S. RV, and J. Liu, "Efficient implementation of password-based authenticated key exchange from RLWE and post-quantum TLS," Cryptology ePrint Archive, Rep. 2017/1192, 2017. [Online]. Available: http://eprint.iacr.org/2017/1192

[20] X. Gao, J. Ding, J. Liu, and L. Li, "Post-quantum secure remote password protocol from RLWE problem." Cryptology ePrint Archive, Rep. 2017/1196, 2017. [Online]. Available: http://eprint.iacr.org/2017/1196

[21] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and E. Barker, "SP 800-22 Rev. 1a, A statistical test suite for random and pseudorandom number generators for cryptographic applications," [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-22/rev-1a/final, Accessed: Jan. 2018.

[22] M. R. Albrecht, R. Player, and S. Scott, "On the concrete hardness of learning with errors," *J. Math. Cryptology*, vol. 9, no. 3, pp. 169–203, 2015.

[23] C. Aguilar-Melchor, J. Barrier, S. Guelton, A. Guinet, M.-O. Killijian, and T. Lepoint, "NFLlib: NTT-based fast lattice library," in *Proc. Cryptographers' Track RSA Conf.*, 2016, pp. 341–356.

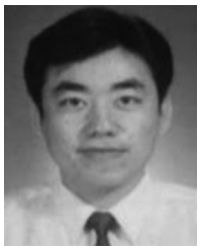
[24] X. Gao, J. Ding, R. V. Saraswathy, L. Li, and J. Liu, "Comparison analysis and efficient implementation of reconciliation-based RLWE key exchange protocol," Cryptology ePrint Archive, Rep. 2017/1178, 2017. [Online]. Available: http://eprint.iacr.org/2017/1178

[25] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," in *Proc. Annu. Int. Cryptology Conf.*, 1999, pp. 537–554.
[26] J. Bos, et al., "CRYSTALS–Kyber: A CCA-secure module-lattice-based KEM," Cryptology ePrint Archive, Rep. 2017/634, 2017. [Online]. Available: http://eprint.iacr.org/2017/634

Xinwei Gao received the BS degree from Beijing Jiaotong University, in 2014. He is working towards the PhD degree with the Beijing Key Laboratory of Security and Privacy in intelligent transportation at Beijing Jiaotong University. He is currently a visiting student with the University of Cincinnati with sponsorship from the China Scholarship Council. His research interests include post-quantum cryptography and RLWE-based key exchange.

Jintai Ding received the PhD degree from Yale, in 1995. He is currently a professor of mathematics with the University of Cincinnati. He was Humboldt fellow and visiting professor with the TU Darmstadt, in 2006-2007. He received the Zhong Jia Qing Prize from the Chinese Math Society, in 1990. His research interest include the post-quantum cryptography (PQC). He was a co-chair of the second international workshop on PQC. He and his colleagues invented LWE- and RLWE-based key exchange protocols, Rainbow signature, "GUI" HFEV- signature and Simple Matrix encryption.

Lin Li received the PhD degree from Shandong University, in 2007. She is currently an assistant professor at the Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University. Her current research interests include cryptography and privacy preserving.



Jiqiang Liu received the BS and PhD degrees from Beijing Normal University, in 1994 and 1999 respectively. He is currently a professor with the Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University. He is also the vice dean of the graduate school of Beijing Jiaotong University. His research interests include security protocols, trusted computing and privacy preserving.