Ding Key Exchange

Submit to

NIST's call for Post-quantum Cryptography Standardization

Principle Inventor, Owner & Principle Submitter

Jintai Ding
Department of Mathematical Sciences, University of Cincinnati
4314 French Hall West, 2815 Commons Way, Cincinnati, Ohio 45221, USA
jintai.ding@gmail.com
(+1) 513-556-4024

Submitters, Developers

Tsuyoshi Takagi
Graduate School of Information Science and Technology, The University of Tokyo; Institute of Mathematics for Industry, Kyushu University; CREST, Japan Science and Technology Agency
takagi@mist.i.u-tokyo.ac.jp

Xinwei Gao
Department of Information Security, School of Computer and Information Technology, Beijing Jiaotong University
xinwei.gao.7@gmail.com

Yuntao Wang
Graduate School of Mathematics, Kyushu University; Graduate School of Information Science and Technology, The University of Tokyo; JSPS Research Fellowship for Young Scientists
y-wang@math.kyushu-u.ac.jp

November 30, 2017

Table of Contents

1 Introduction

1.1 Diffie-Hellman Key Exchange

Key exchange is a very important cryptographic primitive which allows communicating parties to agree on same keys. Shared keys generated during key exchanges can be used by other cryptographic primitives, including symmetric encryption, message authentication code etc. In most cases, a key derived from a key exchange protocol is used to encrypt actual communication data between parties using symmetric encryption. Large amount of communication data is not encrypted using public key cryptosystems since performance of symmetric encryption schemes is far better than public key encryption schemes. Symmetric encryption is much more desirable for real world applications, but it needs to have shared keys between the users. Therefore a secure and efficient key exchange protocol is critical to build a secure communication channel.

In 1976, the first key exchange primitive – Diffie-Hellman key exchange protocol was proposed in [19]. This ground-breaking work is a key part of public key cryptography and it inspires cryptographers to build new public key cryptosystems and key exchange protocols. In a Diffie-Hellman key exchange, the multiplicative group of integers modulo prime p and g, which is a primitive root modulo p, are public parameters. Party i generates private key $s_i = a$, computes public key $p_i = g^a \mod p$ and sends to party j. Party j generates a private key $s_j = b$, computes a public key $p_j = g^b \mod p$ and a shared key $sk_j = p_i^b \mod p = (g^a)^b \mod p$, and sends p_j to party i. Party i computes the shared key $sk_i = p_j^a \mod p = (g^b)^a \mod p$.

With the construction above, one can see that the success of Diffie-Hellman key exchange relies of the commutativity of power maps in a cyclic group, namely the map $f(x) = x^a$ and the map $h(x) = x^b$ commute, i.e. $f \circ h = h \circ f$ where $\circ$ is composition. One critical property of such maps is nonlinearity, which is the basis of the security of this public key scheme. There were various attempts on building similar key exchange cryptosystems, including Braid group based schemes etc. However, they are all broken in practice. There is a very subtle reason behind this fact, which comes from a fundamental work of Joseph Ritt in 1923. Ritt proved that that there are actually essentially only three non-trivial commuting nonlinear maps: power polynomials, Chebyshev polynomials and elliptic curve [35]. This is the reason why we can only build and deploy Diffie-Hellman key exchange and its elliptic curve variants in real world applications. We believe philosophically this is also why all other attempts on building Diffie-Hellman-like key exchange using other structures failed. This fact was first pointed out in [22] by Ding, though he knew it since 2005.

1.2 Quantum Threats

With properly chosen parameters and implementations, Diffie-Hellman key exchange and its variants are hard to break with current computing resources. Till now, they are widely deployed in real world important security protocols and applications. However, such cryptosystems are no longer secure against sufficiently large quantum computers. In 1994, Shor proposed a quantum algorithm which can solve discrete logarithm problem (DLP) and integer factorization problem (IFP) on a quantum computer [38] in polynomial time. Therefore, if a sufficient large quantum computer is built, Shor's algorithm is expected to break cryptosystems which are constructed based on DLP, IFP and their elliptic curve variants etc. Various well-known cryptosystems, including Diffie-Hellman key exchange, RSA, DSA, ElGamal and elliptic curve variants etc. are all vulnerable to large quantum computers with Shor's algorithm.

1.3 Quantum-Resistant Construction from LWE/RLWE Problem

Since Diffie-Hellman-like key exchange protocols can be only built with commutative nonlinear maps and the three commutative maps pointed out by Ritt are all vulnerable to quantum computers due to its related group structure (multiplicative groups of numbers, multiplicative groups on the unit circle and multiplicative groups on elliptic curves). In this proposal, we focus on using a robust and truly efficient primitive – the Ring Learning With Errors (RLWE) problem [31]. The RLWE problem is the ring variant of Learning With Errors (LWE) problem [34], where the security of LWE problem is reduced to hard problems in lattices. A major advantage for RLWE compared with LWE is that it has a much reduced key size, and this is more desirable for real world applications due to smaller communication and storage cost. From our research, we believe RLWE is naturally the best choice for building Diffie-Hellman-like key exchange protocol since commutativity is inherent for multiplication in polynomial rings. Moreover, security of RLWE problem can be reduced to hard problems in ideal lattices, and there are currently no known classic and quantum algorithms can solve RLWE problem efficiently with properly chosen parameters. Moreover, RLWE-based cryptosystems can be constructed and implemented truly efficiently. With all these desirable properties, RLWE becomes an attractive candidate to build post-quantum key exchange protocol.

1.3.1 Design Rationale

Ever since LWE and RLWE problems were introduced in 2005, people have tried to seek solutions to construct key exchange over LWE/RLWE problem. Many attempts were made at the beginning but there were no major real breakthroughs on building complete LWE/RLWE-based key exchange protocol. Major technical challenge to construct Diffie-Hellman-like key exchange protocol over LWE & RLWE is the difficulty to reconcile errors efficiently, since LWE & RLWE samples are perturbed

by small error terms. Despite the fact that both parties can easily compute values that are approximately equal, they cannot derive exact same value (i.e. final shared key) directly without an algorithm to reconcile errors between approximately equal values.

The first complete solution appeared in the LWE & RLWE-based key exchange protocols proposed by Ding et al. in 2012 [22]. [22] invented a novel approach called "robust extractor" (i.e. error reconciliation mechanism) to reconcile errors between two approximately equal values. In order to assist error reconciliations, one party needs to send additional information (denoted as "signal") on the intervals of certain values to the other party. We note that the idea of sending additional values to assist error reconciliation to derive a complete key exchange protocol construction over LWE & RLWE was a fundamentally new method by the time [22] was published.

The reason for sending signal value is directly related to error reconciliation mechanism, where we try to extract same least significant bit from approximately equal values using modulo 2 computation for both parties.

In the construction, first we add a term, which is two times of error term, onto $a \cdot s$, where a a is uniformly random polynomial in the ring $R_q = \mathbb{Z}_q[x]/\Phi_m(x)$, with $\Phi_m(x) = x^n + 1$ as the m-th cyclotomic polynomial with $n = m/2$, q be a prime number and s be a private key polynomial sampled according to some error distribution in ring R_q. Afterwards, key exchange computations for both parties can get approximately equal number with same parity. However, even if the difference between two approximately equal numbers is even, they may not necessarily have the same parity. This is because each coefficient in the ring R_q is in $\mathbb{Z}_q$, where the modulo to prime q operation causes such problem.

In order to solve this problem, Ding in [22] proposed the idea of sending additional bits (i.e. signal values) to assist error reconciliation. Suppose we represent $\mathbb{Z}_q$ as $\{-\frac{q-1}{2}, \cdots, \frac{q-1}{2}\}$ and we divide $\mathbb{Z}_q$ into two regions: inner region ($[-\lfloor \frac{q}{4} \rfloor, \lfloor \frac{q}{4} \rfloor]$ or $[-\lfloor \frac{q}{4} \rfloor + 1, \lfloor \frac{q}{4} \rfloor + 1]$) and outer region (rest of $\mathbb{Z}_q$). If a number lies in the outer region and we add a small error, it is possible that the sum may jump across the boundary of $\mathbb{Z}_q$ even if the error is "small", which causes additional modulo operations even if the difference is supposed to be even. This leads to failure of generating the same key bit. To resolve this issue, the idea of sending additional information, namely the signal function was invented. Since such problem mainly lies for values in the outer region (particularly, close to boundary of outer region), if a value is in the outer region, the signal value is set to 1 and both parties add $\frac{q-1}{2}$ simultaneously to pull the value back to the inner region. With this observation, we can see that since the difference is small, the probability that one party gets additional modulo operation than the other side after adding $\frac{q-1}{2}$ on both sides is low. With the help of the signal value, we can preserve same parity for ring element coefficients of both parties without causing any additional modulo q operation, therefore error reconciliation can be made successfully with overwhelming probability. This approach gives much smaller communication

cost compared with encryption-based approaches. Also computations of signals and final shared keys are very efficient. We will elaborate on this in following sections.

As we stated above, signal functions and error reconciliation mechanisms allow both parties to agree on the same key bits. With our approach, as long as the difference between two values is even and "close", error reconciliation mechanism can reconcile errors and generate the same key bits with overwhelming probability. This clears the way towards constructing Diffie-Hellman-like key exchange protocols over LWE & RLWE problems. The RLWE-based key exchange protocol presented in [22] is the basis of this submission.

Moreover, in this proposal, we introduce a new efficient rounding technique to reduce communication cost. As a common disadvantage over traditional (Diffie-Hellman-like) key exchange protocols, LWE & RLWE-based ones have larger communication cost due to the construction of LWE & RLWE problems. [11] proposed Learning With Rounding (LWR) problem, which can reduce communication cost of LWE problem through rounding. Intuitively, rounding technique compresses public key, therefore the cryptosystem is more efficient in terms of the communication cost. Since rounding and recovering algorithms introduce deterministic errors, [11] suggests that error term in LWE problem can be eliminated with properly chosen rounding parameters, therefore both parties do not have to sample error term and computation efficiency is improved as well. [13] and [7] present cryptanalysis on LWR problem. However, since LWR and its ring variant RLWR have not gain enough attention regarding to cryptanalysis, concrete security of RLWR problem is yet to be fully analyzed. In LWR and RLWR, the secret and random error term e is no longer generated. "Error" for the term $a \cdot s$ is only generated by deterministic rounding and recovering algorithm, and this brings security concerns over LWR and RLWE problems. Moreover, LWR-based cryptosystems considered for practical applications tend to have large parameters and therefore actually higher communication cost, which is not friendly towards real world applications. Inspired by the notion of LWR and our efficient RLWE-based key exchange, we introduce a different rounding technique to reduce communication cost for our key exchange protocol. Unlike LWR-based cryptosystems where error term is discarded, we preserve the freshly generated and secret error term $2e$ in our RLWE instance $a \cdot s + 2e$, then we apply our new rounding technique. We call this a LWE+R sample. By implementing rounding and recovering techniques dedicated to our key exchange protocol, we reduce communication cost substantially, further improving the practical efficiency of our key exchange protocol. Moreover, it actually adds larger perturbation – "error" on $a \cdot s$ compared to standard RLWE instance. The added perturbation helps to improve security of our protocol even further.

1.3.2 Similar Works

We note that there are several works that follow the same idea of sending additional information – signal value other than public key to construct LWE/RLWE/MLWE-based Diffie-Hellman-like key exchange/Key Encapsulation Mechanism (KEM) protocols. Here we list a few of them: BCNS [16], NewHope [6], Frodo [14], NewHope-Simple [5], Kyber [15] etc. These schemes all follow the same method of sending additional bits first presented by Ding to indicate which specific region coefficient elements lie in, though mechanisms for signal value computation and reconciliation maybe somewhat different. With such information, both parties can then reconcile errors using different techniques and agree on same key.

It is very clear that BCNS [16], NewHope [6], Frodo [14] are variants of [22] and our proposal in the sense that all these constructions are LWE/RLWE-based Diffie-Hellman-like key exchange protocols using essentially the same basic idea of Ding. Here both parties together produce the final shared key, which is very different from encryption-based constructions. All these constructions also send signal value to assist error reconciliation. NewHope-Simple [5] and Kyber [15] are KEM constructions over RLWE and MLWE problems respectively. They reduce size of ciphertext, but they also use the idea of sending additional information on the intervals of coefficients to reconcile errors. Despite the algorithms to compute signal value and error reconciliation may not be exactly the same, the idea of sending additional bits to reconcile errors remain the same.

1.4 Summary of Our Proposal

In this submission, we propose an ephemeral-only RLWE-based key exchange protocol. Our construction is a RLWE variant of the classic Diffie-Hellman key exchange protocol, which can be regarded as a direct drop-in replacement for current widely-deployed Diffie-Hellman key exchange protocol (and its variants, e.g. elliptic curve Diffie-Hellman) without significant modifications to current security protocols and applications. We believe that our proposal is secure, efficient, simple and elegant. We present protocol specifications, parameter choices, security analysis using state-of-the-art lattice attacks, performance analysis and stress advantages, limitations and applications of our proposal.

2 Protocol Specification

In this section, we introduce the specification of our RLWE-based key exchange protocol.

2.1 Preliminaries

Let $R_q = \mathbb{Z}_q[x]/f(x)$ be the quotient ring of integer polynomials with $f(x) = x^n + 1$, q a prime number, and n a number as a power of 2. A polynomial a in R_q is represented as $a = a_1 + a_2x + \cdots + a_nx^{n-1}$. Coefficients of a polynomial a can also denoted by a vector $\mathbf{a} = (a_1, ..., a_n)$.

Let Λ be a discrete subset of $\mathbb{Z}^n$. For any vector $\mathbf{c} \in \mathbb{R}^n$ and any positive parameter $\sigma > 0$, let $\rho_{\sigma,\mathbf{c}}(\mathbf{x}) = e^{-\pi\|\mathbf{x}-\mathbf{c}\|^2/\sigma^2}$ be the Gaussian function on $\mathbb{R}^n$ with the center $\mathbf{c}$ and the parameter σ. Denote $\rho_{\sigma,\mathbf{c}}(\Lambda) = \sum_{x\in\Lambda}\rho_{\sigma,\mathbf{c}(x)}$ be the discrete integral of $\rho_{\sigma,\mathbf{c}}$ over Λ, and $D_{\Lambda,\sigma,\mathbf{c}}$ be the discrete Gaussian distribution over Λ with the center $\mathbf{c}$ and the parameter σ. For all $\mathbf{y} \in \Lambda$, we have $D_{\Lambda,\sigma,\mathbf{c}}(\mathbf{y}) = \frac{\rho_{\sigma,\mathbf{c}}(\mathbf{y})}{\rho_{\sigma,\mathbf{c}}(\Lambda)}$. In this submission, we fix Λ to be $\mathbb{Z}^n$ and $\mathbf{c}$ to be zero vector. For ease of notation, we denote $D_{\mathbb{Z}^n,\sigma,0}$ as $D_{\mathbb{Z}^n,\sigma}$. Let $U[a, b]$ be the uniform distribution over discrete set $\{a, a+1, \cdots, b-1, b\}$ over integers. Let $\xleftarrow{\$} \chi$ denote a random sampling according to the distribution χ.

Here we represent $\mathbb{Z}_q$ as $\{-\frac{q-1}{2}, \cdots, \frac{q-1}{2}\}$. However, on occasion, we treat elements in $\mathbb{Z}_q$ as elements in $\{0, \cdots, q-1\}$ for convenience, but we will remark the switch clearly.

Let $||\cdot||_1$ be the l_1-norm, $||\cdot||_2$ be the l_2-norm, $||\cdot||_\infty$ be the l_∞-norm. Let $\lfloor x \rfloor$ be the floor function which outputs the greatest integer that is less than or equal to x, $\lceil x \rceil$ be ceiling function which outputs the least integer that is greater than or equal to x, $\lfloor x \rceil$ be the rounding function which rounds x to nearest integer. Let "$a\|b$" denotes the concatenation of a and b.

First we recall and introduce useful lemmas.

Lemma 1 ([40], lemma 2.5). *For $\sigma > 0$, $r \geq 1/\sqrt{2\pi}$, $\Pr[\|x\|_2 > r\sigma\sqrt{n}; x \xleftarrow{\$} D_{\mathbb{Z}^n,\sigma}] < (\sqrt{2\pi e r^2} \cdot e^{-\pi r^2})^n$.* □

Lemma 2. *For $a, b \in R_q$, $\|a \cdot b\|_\infty \leq \|a\|_2 \cdot \|b\|_2$.*

Proof. Denote the coefficient vector of polynomial $a(x) = a_1 + a_2x + a_3x^2 + \cdots + a_{n-1}x^{n-2} + a_nx^{n-1} \in R_q$ as $(a_1, a_2, a_3, \cdots, a_{n-1}, a_n)$.

For $c = a \cdot b \in R_q$, c_n equals the inner product of $(a_1, a_2, \cdots, a_{n-1}, a_n)$ and $(b_n, b_{n-1}, \cdots, b_2, b_1)$. Similar computations can be applied to coefficients $c_{n-1}, \cdots, c_2, c_1$ as well. By applying Cauchy-Schwarz inequality and property of norm (i.e. for any vector $\mathbf{x}$, $\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_1$), we have $\|c\|_\infty \leq \|a\|_2 \cdot \|b\|_2$. □

2.2 Core Functions

In this section, we define several functions which are crucial to construct our RLWE-based key exchange protocols.

Hint function. 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. A signal function Sig() is defined as:

For any $y \in \mathbb{Z}_q$, $\mathrm{Sig}(y) = \sigma_b(y)$, where $b \xleftarrow{\$} \{0,1\}$. If $\mathrm{Sig}(y) = 1$, we say y is in the outer region, otherwise y is in the inner region.

Signal function is defined for an integer $x \in \mathbb{Z}_q$. Signal function for $a \in R_q$ is computed by applying Sig() for each coefficient $a_i \in \mathbb{Z}_q$ of $a \in R_q$. In this document, we use the same notation "Sig()" for both signal functions over $\mathbb{Z}_q$ and R_q.

Reconciliation function. $\mathrm{Mod}_2()$ is a deterministic function with error tolerance δ. $\mathrm{Mod}_2()$ is defined as: for any x in $\mathbb{Z}_q$ and $w = \mathrm{Sig}(x)$, $\mathrm{Mod}_2(x,w) = (x + w \cdot \frac{q-1}{2} \mod q) \mod 2$. Here we treat elements in $\mathbb{Z}_q$ as elements in $\mathbb{Z}$ before we perform the modulo 2 operation.

We define the error tolerance δ, as the largest integer such that for any $x, y \in \mathbb{Z}_q$, if $\|x - y\|_\infty \leq \delta$, then $\mathrm{Mod}_2(x,w) = \mathrm{Mod}_2(y,w)$, where $w = \mathrm{Sig}(y)$. Error tolerance δ is $\frac{q}{4} - 2$, which is the key to ensure correctness of key exchange over RLWE with overwhelming probability.

Reconciliation function is defined for an integer $x \in \mathbb{Z}_q$. Reconciliation function for $a \in R_q$ is computed by applying $\mathrm{Mod}_2()$ for each coefficient $a_i \in \mathbb{Z}_q$ of $a \in R_q$. In this document, we use the same notation "$\mathrm{Mod}_2()$" for both reconciliation functions over $\mathbb{Z}_q$ and R_q.

Lemma 3. *Let $q > 8$ be an odd integer. Function $Mod_2()$ as defined above is a robust extractor with respect to signal function $Sig()$ with error tolerance $\delta = \frac{q}{4} - 2$.*

Proof. For any $x, y \in \mathbb{Z}_q$ such that $x - y = 2\varepsilon$ and $|2\varepsilon| \leq \frac{q}{4} - 2$. Let $w \leftarrow \mathrm{Sig}(y)$, due to definition of signal function, it is not hard to see that $|y + w \cdot \frac{q-1}{2} \mod q| \leq \frac{q}{4} + 1$ for any hint function used in Sig() to generate signal w. We have

$$\begin{aligned} x + w \cdot \frac{q-1}{2} \mod q &= y + w \cdot \tfrac{q-1}{2} + 2\varepsilon \mod q \\ &= (y + w \cdot \tfrac{q-1}{2}) \mod q + 2\varepsilon \end{aligned}$$

Since $|(y + w \cdot \frac{q-1}{2}) \mod q + 2\varepsilon| \leq \frac{q}{4} + 1 + |2\varepsilon| \leq \frac{q-1}{2}$. This implies

$$\begin{aligned} \mathrm{Mod}_2(x,w) &= (x + w \cdot \frac{q-1}{2} \mod q) \mod 2 \\ &= (y + w \cdot \frac{q-1}{2} \mod q) \mod 2 \\ &= \mathrm{Mod}_2(y,w) \end{aligned}$$

□

Lemma 4. *For any odd $q > 2$, if x is uniformly random in $\mathbb{Z}_q$, then $Mod_2(x, w)$ is uniformly random conditioned on signal $w \in \{0, 1\}$.*

Proof. For any $w, b' \in \{0, 1\}$, we have

$$\begin{aligned}\Pr_{x \xleftarrow{\$} \mathbb{Z}_q, b \xleftarrow{\$} \{0,1\}}[\mathrm{Mod}_2(x, w) = b' | \sigma_b(x) = w] &= \frac{1}{2} \Pr_{x \xleftarrow{\$} \mathbb{Z}_q}[\mathrm{Mod}_2(x, w) = b' | \sigma_0(x) = w] \\ &+ \frac{1}{2} \Pr_{x \xleftarrow{\$} \mathbb{Z}_q}[\mathrm{Mod}_2(x, w) = b' | \sigma_1(x) = w] \\ &= \frac{1}{2} \cdot \frac{\Pr_{x \xleftarrow{\$} \mathbb{Z}_q}[\mathrm{Mod}_2(x, w) = b' \wedge \sigma_0(x) = w]}{\Pr_{x \xleftarrow{\$} \mathbb{Z}_q}[\sigma_0(x) = w]} \\ &+ \frac{1}{2} \cdot \frac{\Pr_{x \xleftarrow{\$} \mathbb{Z}_q}[\mathrm{Mod}_2(x, w) = b' \wedge \sigma_1(x) = w]}{\Pr_{x \xleftarrow{\$} \mathbb{Z}_q}[\sigma_1(x) = w]}\end{aligned}$$

Denote $I = [-\lfloor \frac{q}{4} \rfloor, \lfloor \frac{q}{4} \rfloor]$ be the interval such that σ_0 equals 0, then $I + 1$ is the interval such that σ_1 equals 0. It is easy to see that $|I| = |I + 1| = 2\lfloor \frac{q}{4} \rfloor + 1$.

We separately consider two cases, when $w = 0$ and $w = 1$.

For $w = 0$, we have that

$$\Pr_{x \xleftarrow{\$} \mathbb{Z}_q}[\sigma_0(x) = 0] = \Pr_{x \xleftarrow{\$} \mathbb{Z}_q}[\sigma_1(x) = 0] = \frac{2\lfloor \frac{q}{4} \rfloor + 1}{q}.$$

Let $I_0 = \{x : x \in I \wedge x \bmod 2 = 0\}$ and $I_1 = \{x : x \in I \wedge x \bmod 2 = 1\}$ and similarly for $(I+1)_0, (I+1)_1$. Then we have $|I_0| + |(I+1)_0| = |I|$ and $|I_1| + |(I+1)_1| = |I|$.

Therefore,

$$\begin{aligned}\Pr_{x \xleftarrow{\$} \mathbb{Z}_q}[\mathrm{Mod}_2(x, 0) = b' \wedge \sigma_0(x) = 0] &= \Pr_{x \xleftarrow{\$} \mathbb{Z}_q}[x \in I_{b'}] \text{ and} \\ \Pr_{x \xleftarrow{\$} \mathbb{Z}_q}[\mathrm{Mod}_2(x, 0) = b' \wedge \sigma_1(x) = 0] &= \Pr_{x \xleftarrow{\$} \mathbb{Z}_q}[x \in (I + 1)_{b'}].\end{aligned}$$

This implies that

$$\Pr_{x \xleftarrow{\$} \mathbb{Z}_q, b \xleftarrow{\$} \{0,1\}}[\mathrm{Mod}_2(x, 0) = b' | \sigma_b(x) = 0] = \frac{1}{2} \cdot \frac{q}{2\lfloor \frac{q}{4} \rfloor + 1} \cdot \frac{|I_{b'}| + |(I + 1)_{b'}|}{q} = \frac{1}{2}.$$

For $w = 1$, we first note that the intervals $\mathbb{Z}_q \backslash I$ and $\mathbb{Z}_q \backslash (I+1)$ have even numbers, *i.e.* $q - (2\lfloor \frac{q}{4} \rfloor + 1)$. Therefore, we have:

$$\begin{aligned}&\Pr_{x \xleftarrow{\$} \mathbb{Z}_q}[\mathrm{Mod}_2(x, 1) = b' \wedge \sigma_0(x) = 1] + \Pr_{x \xleftarrow{\$} \mathbb{Z}_q}[\mathrm{Mod}_2(x, 1) = b' \wedge \sigma_1(x) = 1] \\ =&\frac{q - (2\lfloor \frac{q}{4} \rfloor + 1)}{q}.\end{aligned}$$

A routine calculation shows that $\Pr_{x \xleftarrow{\$} \mathbb{Z}_q, b \xleftarrow{\$} \{0,1\}}[\mathrm{Mod}_2(x,1) = b' | \sigma_b(x) = 1] = \frac{1}{2}$. This completes the proof. □

Rounding function. For $x \in \mathbb{Z}_q$, $q > p > 0$ be integers. x is a coefficient of polynomial in R_q, q, p are parameters of our protocol.

For the convenience of notation, we change the representation of $x \in \{-\frac{q-1}{2}, \cdots, \frac{q-1}{2}\}$ to $x \in \{0, \cdots, q-1\}$ before Round() runs. Function Round(x, p, q) is defined as follows:

Algorithm 1 Round(x, p, q)

Input: $x \in \mathbb{Z}_q, p, q$
Output: Rounded value $x' \in [0, p]$ of x
1: $x' \leftarrow \lfloor p \cdot x / q \rceil$
2: **if** ((x is odd number) AND (x' is even number)) **then**
3: $\quad x' \leftarrow x' + 1$
4: **else if** ((x is even number) AND (x' is odd number)) **then**
5: $\quad x' \leftarrow x' + 1$
6: **end if**
7: Remove_bias(x', p, q)

Rounding function is defined for an integer $x \in \mathbb{Z}_q$. Rounding function for $a \in R_q$ is computed by applying Round() for each coefficient $a_i \in \mathbb{Z}_q$ of $a \in R_q$. In this document, we use the same notation Round() for both rounding functions over $\mathbb{Z}_q$ and R_q.

A function in order to remove possible bias occurred in rounding is defined below.

Remove bias function. Remove_bias() shares similar idea as hint functions $\sigma_0(x)$ and $\sigma_1(x)$. Since rounding function Round() generates minor bias on some locations (i.e. one more number is rounded to bias locations than nearby locations), it is necessary to remove the bias.

Remove_bias(x', p, q) takes inputs x', p, q, where x' is derived after line 6 of algorithm 1. Since the location of "bias" varies regarding to specific parameters p and q, we define this function in section 2.3.3.

Recovering function. Recover() is a deterministic function. For $x' \in [0, p]$, $q > p > 0$ be integers. x' is one coefficient of rounded polynomial, q, p are parameters of our protocol. Function Recover(x', p, q) is defined as follows:

Algorithm 2 Recover(x', p, q)

Input: $x' \in [0, p], p, q$
Output: Recovered value x'' of x'
1: $x'' \leftarrow \lfloor x' \cdot q/p \rfloor$
2: **if** ((x' is odd number) AND (x'' is even number)) **then**
3: $\quad x'' \leftarrow x'' + 1$
4: **else if** ((x' is even number) AND (x'' is odd number)) **then**
5: $\quad x'' \leftarrow x'' + 1$
6: **end if**

In order to be consistent with theoretical analysis, we change representation of $x'' \in \{0, \cdots, q-1\}$ to $x'' \in \{-\frac{q-1}{2}, \cdots, \frac{q-1}{2}\}$ after Recover() runs.

Recovering function is defined for an integer $x' \in [0, p]$. Recovering function for vector $\mathbf{a}$ is computed by applying Recover() for each coefficient a_i in vector $\mathbf{a}$. In this document, we use the same notation "Recover()" for both recovering functions over integer x' and vector $\mathbf{a}$.

Lemma 5. *For parameter p and q, let $t = \lceil \log_2 q \rceil - \lceil \log_2 p \rceil$, $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ be a vector whose each coefficient is uniformly random sampled integer in $\mathbb{Z}_q$, **x'** be a vector whose each coefficient x'_i = Recover(Round(x_i, p, q),p, q). Let $\boldsymbol{d} =$ **x-x'** be a vector whose each coefficient $d_i = x_i - x'_i$ ($i \in [1, n]$). Then d_i is an even number with possible values in set $\{0, \pm 2, \cdots, \pm 2^t\}$, i.e. $0 \leq |d_i| \leq 2^t$.* □

Since probability for each possible d_i varies with respect to parameter choice p, q, we will introduce it in section 2.3.3.

a derivation function. In each key exchange execution, we use a 128-bit *seed* to generate fresh a. Set *seed* to pseudorandom number generator. Each coefficient $a_i \in \mathbb{Z}_q$ ($i \in [1, n]$) of $a \in R_q$ is derived as follows:

Algorithm 3 Derive_a()

Output: Coefficient a_i of polynomial $a \in R_q$
1: $a_i \xleftarrow{\$} U[0, q-1]$

2.3 RLWE-based Key Exchange Protocol

In this section, we present our RLWE-based key exchange protocol. Here we first describe a few basic primitives:

- Fresh a generation: Set 128-bit *seed* to pseudorandom number generator. Derive fresh $a \in R_q$ using Derive_a() function.

- Key generation: For public parameter $a \in R_q$, sample s, e from $D_{\mathbb{Z}^n,\sigma}$. Public key is $pk = a \cdot s + 2e \in R_q$ and private key is s. Round pk as pk' using function Round().
- Key exchange material computation: For a rounded public key pk' received from the other party, first recover pk' using function Recover() and denote as pk''. For private key $s \in R_q$, compute key exchange material $k = pk'' \cdot s \in R_q$.
- Signal computation: For key exchange material $k \in R_q$, compute signal $w \in \{0,1\}^n$ using function Sig().
- Error reconciliation: For key exchange material $k \in R_q$ and signal $w \in \{0,1\}^n$, reconcile errors and generate the final shared key $sk \in \{0,1\}^n$ using function $\text{Mod}_2()$ and w.

As we stated before, one particular technical challenge to construct Diffie-Hellman-like key exchange protocol over RLWE problem is how to reconcile errors. Therefore signal function and error reconciliation mechanism are invented. Moreover, in order to further reduce communication cost, we introduce the rounding technique. Apart from reducing communication cost, additional error causes larger perturbation on $a \cdot s$ than simply adding $2e$, resulting in an increased cost of attack on our protocol.

As NIST's call for proposal requested, our Diffie-Hellman-like RLWE-based key exchange protocol is formalized as KEM consisting of three major functions:

- `crypto_kem_keypair()`:
  - Generate random *seed*, party i's ephemeral public key $p'_i \in R_q$ and private key $s_i \in R_q$. Public key $pk = p'_i \| seed$.
  - Return pk and s_i.
- `crypto_kem_enc()`:
  - Generate party j's ephemeral public key $p'_j \in R_q$, private key $s_j \in R_q$ and signal w_j. Compute k_j and generate final shared key sk_j. Ciphertext $ct = p'_j \| w_j$.
  - Return ct and sk_j.
- `crypto_kem_dec()`:
  - Party i computes k_i and generates final shared key $sk_i = sk_j$.
  - Return sk_i.

2.3.1 Specification

We give the description of key exchange between party i and party j. In our protocol, users share following parameters: n, σ, q, p.

Initiate. Party i instantiates key exchange by generating 128-bit random *seed*, computes fresh $a = \text{Derive\_a}()$ and public key $p_i = a \cdot s_i + 2e_i \in R_q$, where s_i and e_i are sampled from $D_{\mathbb{Z}^n,\sigma}$. Round p_i as $p'_i = \text{Round}(p_i, p, q)$, send p'_i and *seed* to party j.

Response. Party j computes fresh $a = \text{Derive\_a}()$, public key $p_j = a \cdot s_j + 2e_j \in R_q$, where s_j and e_j are sampled from $D_{\mathbb{Z}^n,\sigma}$. Round p_j as $p'_j = \text{Round}(p_j, p, q)$. Recover public key received from party i as $p''_i = \text{Recover}(p'_i, p, q)$. Computes key exchange material $k_j = p''_i \cdot s_j \in R_q$, signal value $w_j = \text{Sig}(k_j)$ and final shared key $sk_j = \text{Mod}_2(k_j, w_j)$. Send p'_j and w_j to party i.

Finish. Party i recovers public key received from party j as $p''_j = \text{Recover}(p'_j, p, q)$. Compute key exchange material $k_i = p''_j \cdot s_i \in R_q$ and final shared key $sk_i = \text{Mod}_2(k_i, w_j)$.

The protocol is illustrated in Figure 1.

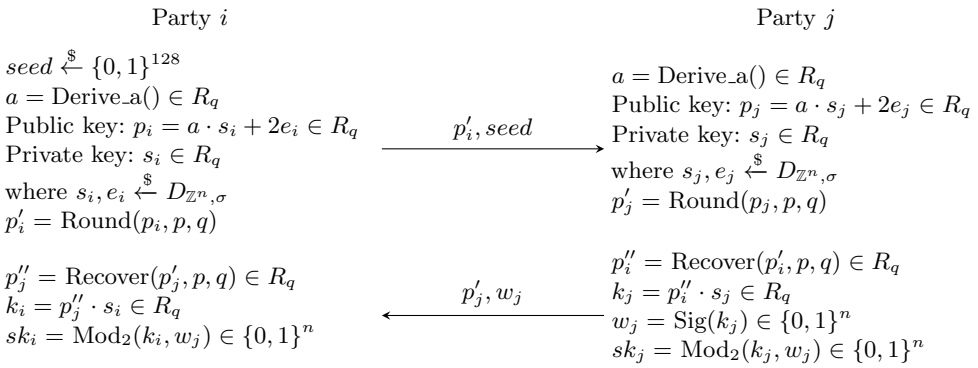

Fig. 1. The proposed RLWE key exchange protocol

2.3.2 Correctness

Note: In this subsection, we omit notation "$\cdot$" for polynomial multiplication.

With above protocol, we have

$$\begin{aligned} k_i =& p''_j s_i = (as_j + 2e_j + d_j)s_i \\ =& as_j s_i + 2e_j s_i + d_j s_i \end{aligned} \tag{1}$$

$$\begin{aligned} k_j =& p''_i s_j = (as_i + 2e_i + d_i)s_j \\ =& as_i s_j + 2e_i s_j + d_i s_j \end{aligned} \tag{2}$$

$k_i - k_j = 2(e_j s_i - e_i s_j) + (d_j s_i - d_i s_j)$. In order to achieve key exchange with overwhelming success probability, $\|k_i - k_j\|_\infty \leq$ error tolerance δ of error reconciliation mechanism, i.e. $\|k_i - k_j\|_\infty \leq \frac{q}{4} - 2$. We have

$$\begin{aligned} \|k_i - k_j\|_\infty =& \|2(e_j s_i - e_i s_j) + (d_j s_i - d_i s_j)\|_\infty \\ \leq& 4\|se\|_\infty + 2\|d's\|_\infty \end{aligned} \tag{3}$$

where $s, e \in R_q \xleftarrow{\$} D_{\mathbb{Z}^n,\sigma}$. Definition of d' is consistent with lemma 5.

With lemma 1 and 2, we have $4\|se\|_\infty \leq 4\|s\|_2 \cdot \|e\|_2 \leq 4(r\sigma\sqrt{n})^2 = 4r^2\sigma^2 n$, where $r \geq 1/\sqrt{2\pi}$ is defined in lemma 1 and n is the degree of polynomial. With lemma 5, we have $2\|d's\|_\infty \leq 2\|d'\|_2 \cdot \|s\|_2 = 2\|d'\|_2 \cdot r\sigma\sqrt{n}$. Recall that error tolerance $\delta = \frac{q}{4} - 2$. Therefore as long as $q \geq 4 \cdot [2 + (4r^2\sigma^2 n) + (2\|d'\|_2 \cdot r\sigma\sqrt{n})]$, key exchange failure probability is estimated to be $(\sqrt{2\pi e r^2} \cdot e^{-\pi r^2})^n$.

2.3.3 Parameter Choice

Key exchange protocol is instantiated with following parameters:

- Modulus q
- Degree n of R_q
- σ of distribution $D_{\mathbb{Z}^n,\sigma}$ to sample s and e
- Rounding parameter p

Parameter choices covering NIST security category I (AES-128), III (AES-192) and V (AES-256) are given in Table 1:

Table 1. Our Parameter Choice

n	σ	q	p	Claimed Security Level	NIST Security Category	Failure Probability
512	4.19	120833	7551	AES-128	I	2^{-60}
1024	2.6	120833	7551	AES-192 AES-256	III V	2^{-60}

Note that for parameter choice $(n, \sigma, q, p) = (1024, 2.6, 120833, 7551)$, it is enough to cover security of AES-192 and AES-256 (NIST security category III and V respectively). We will elaborate this in section 3.1.5.

Modulus $q = 120833$ can instantiate NTT efficiently as $q \equiv 1 \mod 2n$. p is to instantiate Round() and Recover() functions properly. A failed key exchange implies that at least one bit in sk_i and sk_j mismatches.

For lemma 5 and above parameter choices, let $t = \lceil \log_2 q \rceil - \lceil \log_2 p \rceil$. We have:

- $\Pr[d_i = 0] = \Pr[d_i = 2] = \Pr[d_i = -2] = \Pr[d_i = 4] = \Pr[d_i = -4] = \cdots = \Pr[d_i = 2^t - 2] = \Pr[d_i = -(2^t - 2)] = \frac{1}{2^t}$, $\Pr[d_i = 2^t] = \Pr[d_i = -2^t] = \frac{1}{2^{t+1}}$
- $n = 512, t = 4$, $\|d\|_2 = 32\sqrt{43}$
- $n = 1024, t = 4$, $\|d\|_2 = 32\sqrt{86}$

With concrete parameter choices, we define bias removing function Remove_bias() as we mentioned in algorithm 1. We list a set of locations *pos* where bias occurs:

– $q = 120833$, $p = 7551$, $pos = \{0, 445, 888, 1333, 1776, 2221, 2666, 3109, 3554, 3997, 4442, 4885, 5330, 5775, 6218, 6663, 7106\}$

As we explained in section 2.2, one more number is rounded to values in *pos* than nearby values in $[0, p]$, therefore if the number generated after line 6 of algorithm 1 equals to one of the values in *pos*, then bias is removed using Remove_bias().

Function Remove_bias() is defined as:

Algorithm 4 Remove_bias(x', p, q)

Input: $x' \in [0, p], p, q$
Output: x'
1: **if** $x' \in pos$ **then**
2: $rnd \xleftarrow{\$} U[0, 1]$
3: **if** $rnd = 1$ **then**
4: $x' \leftarrow x' + 2$
5: **end if**
6: **end if**

2.3.4 Communication Cost

As shown in Figure 1, party i sends the rounded public key p'_i and *seed* to party j. Party j sends the rounded public key p'_j and signal w_j to party i.

$p_i \in R_q$ is represented as $p_i = (p_{i,1}, p_{i,2}, ..., p_{i,n}) \in \mathbb{Z}_q^n$, but the rounded element $p'_i = \text{Round}(p_i, p, q)$ is represented as $p'_i = (p'_{i,1}, p'_{i,2}, ..., p'_{i,n})$. Same analysis can be applied to p_j and p'_j as well. *seed* has the size of 128 bits. Signal $w_j \in \{0, 1\}^n$ has the size of n bits.

Therefore, with parameter choices presented in Table 1, communication cost is estimated as:

Table 2. Communication Cost of Our Proposed Scheme

n	Party $i \to j$ (Byte)	Party $j \to i$ (Byte)	Total (Byte)	Claimed Security Level	NIST Security Category
512	848	896	1744	AES-128	I
1024	1680	1792	3472	AES-192 AES-256	III V

3 Known Cryptanalytic Attacks

3.1 Expected Security Strength

In this section, we will explain how to analyze the security of our protocol.

3.1.1 Prerequisites

Lattice Theory. A *lattice* L is defined as an infinite space expanded by *basis* $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$, where $\mathbf{b}_i$ $(i = 1, \ldots, n)$ are a set of linearly independent vectors in $\mathbb{R}^m$. Here n is the dimension of L. The n-dimensional *volume* of L is denoted by $\mathrm{Vol}(L)$, which is computed by the *determinant* of basis B, i.e. $\mathrm{Vol}(L) = \det(B)$. We denote $V_n(R) = R^n \cdot \frac{\pi^{n/2}}{\Gamma(n/2+1)}$ as the volume of n-dimensional Euclidean ball of radius R.

Ring LWE (RLWE) Problem. Let $m \geq 1$ be a power of 2 and $q \geq 2$ be an integer, let $R_q = \mathbb{Z}_q[x]/\Phi_m(x)$, where $\Phi_m(x) = x^n + 1$ is the m-th cyclotomic polynomial with $n = m/2$. Let χ be a β-bounded distribution. For secret polynomial $s \xleftarrow{\$} \chi$ and error polynomial $e \xleftarrow{\$} \chi$, choosing $a \in R_q$ uniformly random, output $(a, b = a \cdot s + e \in R_q)$.

Search version of RLWE problem is: for $s \xleftarrow{\$} \chi$, given $poly(n)$ number of samples of $(a, b = a \cdot s + e) \in (R_q, R_q)$, find s (and e simultaneously). Decision version of RLWE problem is: for $a \in R_q$, distinguish $b = a \cdot s + e \in R_q$ from uniform random sampled polynomial in R_q. We denote both search and decision versions of RLWE problem as RLWE problem.

Proposition. Let $z = \mathrm{Recover}(\mathrm{Round}(a \cdot s + 2e, p, q), p, q) = as + 2e + d = as + 2f \in R_q$, where $s, e \xleftarrow{\$} D_{\mathbb{Z}^n,\sigma}$ and $2f = 2e + d$. Hence we can regard f as error term e in the definition of RLWE above. The attack on our protocol is given z and a, output private key s. This problem is equivalent to:

$$\begin{aligned} & z = a \cdot s + 2f \mod q \\ \Leftrightarrow\ & 2^{-1}z = 2^{-1}a \cdot s + f \mod q \\ \Leftrightarrow\ & z'' = a'' \cdot s + f \mod q \end{aligned}$$

Standard deviation of term f is denoted as σ_f. Note that σ_f is different from σ notation in section 2.1 as f no longer follows discrete Gaussian distribution (histogram shows similar shape as Gaussian distribution), therefore σ_f is computed as the square root of variance.

Shortest Vector Problem. Given an input basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice L, *Shortest Vector Problem* (SVP) is to find a non-zero shortest vector in L. We introduce the following two variants of the SVP to be used in this section.

Short Integer Solution Problem. Given an integer q and a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, *Short Integer Solution problem* (SIS) is to compute a short vector $\mathbf{y} \in \mathcal{B}$ s.t. $\mathbf{Ay} \equiv \mathbf{0}$ mod q, where $\mathcal{B}$ is a set of short vectors with some norm bound.

Unique Shortest Vector Problem. *Unique SVP problem* (uSVP) is for a given lattice L which satisfies $\lambda_1(L) \ll \lambda_2(L)$, find the shortest vector in L. Here $\lambda_i(L)$ means the length of i-th linear independent shortest vector for $i = 1, 2$.

(Root) Hermite Factor. To evaluate the performance of lattice algorithms for solving SVP, we use the *Hermite Factor* defined in [24] as:

$$\mathrm{HF}(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \|\mathbf{b}_1\|_2/\mathrm{Vol}(L)^{1/n}.$$

Given an n-dimensional lattice, if the Hermite factor of output basis is smaller, the algorithm performs better. Also we usually use *root Hermite Factor* (rHF) in analysis, which is denoted as:

$$\delta = \mathrm{rHF}(\mathbf{b}_1, \ldots, \mathbf{b}_n) = (\|\mathbf{b}_1\|_2/\mathrm{Vol}(L)^{1/n})^{1/n}.$$

BKZ Algorithm. There are some lattice algorithms such as BKZ and sieving to solve SVP and its variants. *BKZ algorithm* was originally proposed in [36], which computes basis that are almost β-reduced, namely the projected lengths of each basis vectors are the shortest ones in the relative β-sized local blocks. Original BKZ algorithm proceeds by iterative tours consisting of $n-1$ calls to a β-dimensional SVP solver called on the projected lattice spanned by $\pi_j(\mathbf{b}_j), \ldots, \pi_j(\mathbf{b}_{\min(\mathrm{j}+\beta-1,\mathrm{n})})$ $(1 \le j \le n)$, where π_j is the orthogonal projection on $(\mathbf{b}_1, \ldots, \mathbf{b}_{j-1})$. BKZ algorithm runs in exponential time, because the classical SVP oracle enumeration algorithm (ENUM) runs in $2^{O(\beta^2)}$. There are also some efficient improvements for BKZ algorithms [18,9].

Sieving Algorithm. In 2001, Ajtai et al. proposed a sieving algorithm to solve SVP, which requires a runtime of $2^{0.52n+o(n)}$ in n dimension lattice and simultaneously requires exponential storage of $2^{0.2n+o(n)}$ [1]. According to recent research results, for a n-dimensional lattice L and fixed blocksize β in BKZ, the runtime of sieving algorithm can be estimated in $2^{0.292\beta+o(\beta)}$ clock cycles for a β-dimensional subroutine [4], and totally BKZ-β costs $8n \cdot 2^{0.292\beta+16.4}$ operations [12]. We will use this result to evaluate the security of our parameter choices.

3.1.2 Algorithms for Solving RLWE

There are several algorithms for solving RLWE. In Figure 2, we show several possible attacks on RLWE problem with only one given instance.

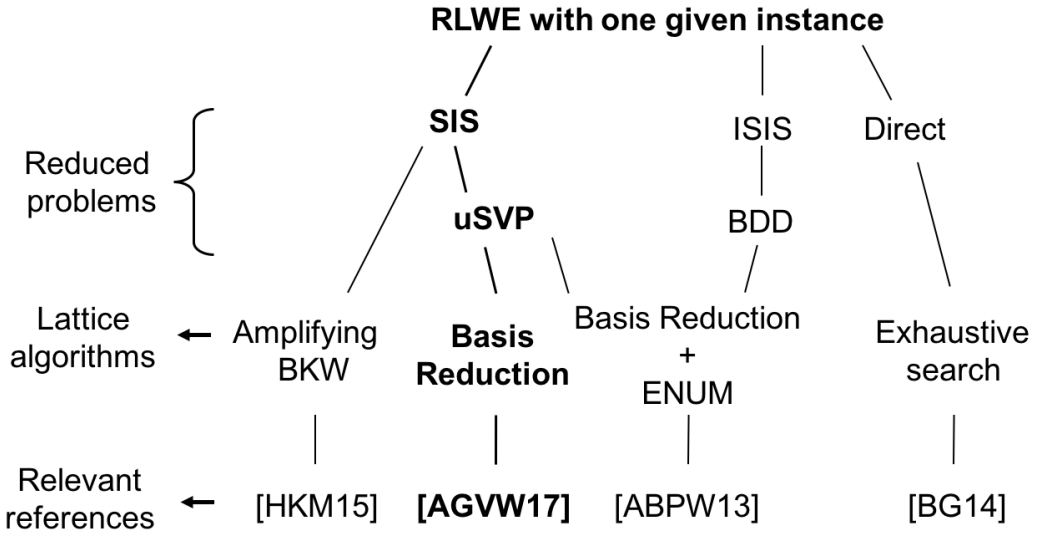

Fig. 2. Possible attacks on search RLWE problem with only one given instance. Relevant references [HKM15], [AGVW17], [ABPW13] and [BG14] are [25], [3], [8] and [10] respectively.

We explain how we choose appropriate attacks from available options:

- Firstly, exhaustive search is not efficient.
- Secondly, BKW algorithm can solve LPN problem with $2^{O(n/\log n)}$ samples and runtime. Since LWE is a descendant of LPN, BKW algorithm can also be adapted to solve LWE problem (both decision and search versions) with $2^{O(n)}$ complexity, when modulus q has polynomial size of dimension n. Amplifying technique is used in BKW algorithm to solve LPN problem and LWE problem [30,25]. However, the analysis on amplifying BKW until now are asymptotical and there is no precise analysis on RLWE problem, i.e. amplifying BKW requires $O(n\log_2(n))$ samples which will lead to much larger standard deviation of e.
- Thirdly, complexity analysis of "reduction+ENUM" method is not clear for large dimensional (> 1000) basis.
- Finally, for security analysis of our protocol, we adapt a conservative primal method: reduce RLWE problem to SIS problem, then reduce it to unique SVP problem. Then we process the basis using BKZ reduction algorithm with sieving algorithm as SVP oracle in BKZ subroutines.

We show the SIS attack on RLWE in Algorithm 5.

Algorithm 5 The SIS Attack on Ring LWE Problem

Input: m' instances from RLWE key exchange: $(a, b = a \cdot s + e) \in (R_q, R_q)$. Here $m' = poly(n)$, $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ with q as a prime, where secret polynomial s and error polynomial e are sampled from Gaussian distribution $D_{\mathbb{Z}^n,\sigma}$ with standard deviation σ.

Output: The polynomial e and s s.t. $b = a \cdot s + e \in R_q$.

Step 1. Rewrite the RLWE instance to Short Integer Solution (SIS) instance by

① Write the polynomials $a_i \in R_q$, $b_i \in R_q$, $e_i \xleftarrow{\$} D_{\mathbb{Z}^n,\sigma} \in R_q$ for $i \in [1, m']$ and the only one $s \in R_q$ as follows:

$$a_i = a_{i1} + a_{i2}x + \cdots + a_{in}x^{n-1}, b_i = b_{i1} + b_{i2}x + \cdots + b_{in}x^{n-1},$$
$$e_i = e_{i1} + e_{i2}x + \cdots + e_{in}x^{n-1} \text{ and } s = s_1 + s_2x + \cdots + s_nx^{n-1}$$

to vector form as

$$\mathbf{a}_i = (a_{i1}, a_{i2}, \cdots, a_{in}) \in \mathbb{Z}_q^{1\times n},\ \mathbf{b}' = (b_{11}, b_{12}, \cdots, b_{1n}, \cdots, b_{m'1}, b_{m'2}, \cdots, b_{m'n})^T \in \mathbb{Z}_q^{m'n\times 1},$$
$$\mathbf{e}' = (e_{11}, e_{12}, \cdots, e_{1n}, \cdots, e_{m'1}, e_{m'2}, \cdots, e_{m'n})^T \xleftarrow{\$} D_{\mathbb{Z}^{m'n},\sigma} \in \mathbb{Z}_q^{m'n\times 1},$$
$$\mathbf{s} = (s_1, s_2, \cdots, s_n)^T \xleftarrow{\$} D_{\mathbb{Z}^n,\sigma} \in \mathbb{Z}_q^{n\times 1}.$$

② Rotate polynomials $a_i = a_{i1} + a_{i2}x + \cdots + a_{in}x^{n-1} \in R_q$ to get matrices

$$\mathbf{A}_i = \begin{pmatrix} a_{i1} & a_{i2} & \dots & a_{i(n-1)} & a_{in} \\ -a_{in} & a_{i1} & a_{i2} & \dots & a_{i(n-1)} \\ -a_{i(n-1)} & -a_{in} & a_{i1} & \dots & a_{i(n-2)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -a_{i2} & -a_{i3} & -a_{i4} & \dots & a_{i1} \end{pmatrix}.$$

Then compose $\mathbf{A}' = [\mathbf{A_1}\mathbf{A}_2 \cdots \mathbf{A}_{m'}]^T \in \mathbb{Z}_q^{m'n\times n}$.

③ Derive $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$ by sampling m-row vectors from $\mathbf{A}' \in \mathbb{Z}_q^{m'n\times n}$. Similarly, derive $\mathbf{b} \in \mathbb{Z}_q^m$ from $\mathbf{b}' \in \mathbb{Z}_q^{m'n\times 1}$ and $\mathbf{e} \in \mathbb{Z}_q^m$ from $\mathbf{e}' \in \mathbb{Z}_q^{m'n\times 1}$.

Then we get a randomly sampled normal LWE case as $(\mathbf{A}, \mathbf{b} \equiv \mathbf{As} + \mathbf{e} \pmod q) \in (\mathbb{Z}_q^{m\times n}, \mathbb{Z}_q^m)$.

④ Transform the randomly sampled LWE instance $(\mathbf{A}, \mathbf{b} \equiv \mathbf{As} + \mathbf{e} \mod q) \in (\mathbb{Z}_q^{m\times n}, \mathbb{Z}_q^m)$ from ③ to a SIS instance:

Given $(\mathbf{A}, \mathbf{b}) \in (\mathbb{Z}_q^{m\times n}, \mathbb{Z}_q^m)$, find a short vector $(\mathbf{s} \mid \mathbf{e} \mid 1) \in \mathbb{Z}_q^{m+n+1}$ s.t.

$$(\mathbf{A}\ |I_m|{-}\mathbf{b})(\mathbf{s} \mid \mathbf{e} \mid 1)^T = \mathbf{0} \mod q.$$

Step 2. Set $\mathbf{A}'' = (\mathbf{A}\ |I_m|{-}\mathbf{b}) \in \mathbb{Z}_q^{m\times(m+n+1)}$. Compute basis $\mathbf{B}$ of q-ary lattice

$$L^{\perp}_{(\mathbf{A}'',q)} = \{\mathbf{x} \in \mathbb{Z}^{m+n+1} \mid \mathbf{A}''\mathbf{x} \equiv \mathbf{0} \pmod q\}.$$

Compute kernel $Ker(\mathbf{A}'')$ of $\mathbf{A}''$ over $\mathbb{Z}^{(m+n+1)\times(n+1)}$.

Step 3. For some matrix $\mathbf{A}^*$, construct basis $\mathbf{B} = \begin{pmatrix} \frac{Ker(\mathbf{A}'')^T}{q\mathbf{I}_{m+n+1}} \end{pmatrix} \in \mathbb{Z}_q^{(m+2n+2)\times(m+n+1)}$

and compute the HNF of $\mathbf{B}$ as $\mathbf{B}_{\text{HNF}} = \begin{pmatrix} q\mathbf{I}_m & \mathbf{0} \\ \mathbf{A}^*_{(n+1)\times m} & \mathbf{I}_{n+1} \end{pmatrix} \in \mathbb{Z}^{(m+n+1)\times(m+n+1)}$.

Step 4. Apply lattice reduction algorithm (LLL or BKZ) on basis $\mathbf{B}_{\text{HNF}}$ and get reduced basis $Red(B_{\text{HNF}})$. Inside of $Red(B_{\text{HNF}})$, a short vector $\mathbf{v} = (\mathbf{s} \mid \mathbf{e} \mid 1)$ if it succeeded.

This algorithm is an adaptation of the primal attack mentioned in [10].

3.1.3 Cost of Known BKZ Estimators

We explain cost of two estimators for BKZ algorithm.

3.1.3.1 Progressive BKZ Runtime Simulator [9]

In 2016, Aono et al. proposed a precise simulator for estimating runtime of progressive BKZ algorithm (pBKZ), which processes given basis by increasing block size with

some strategy [9]. They optimized four critical parameters in BKZ: blocksize β, GSA constant r, ENUM search radius coefficient α and ENUM search success probability p.

Thus, for fixed β, one can compute the other three parameters with equation 4, 5 and 6.

$$p = \frac{2}{\alpha^{\beta}}. \tag{4}$$

$$r = \left(\frac{\beta+1}{\alpha\cdot\beta}\right)^{\frac{4}{\beta-1}} \cdot V_{\beta}(1)^{\frac{4}{\beta(\beta-1)}}. \tag{5}$$

$$\log_2(r) = \begin{cases} -18.2139/(\beta+318.978) & (\beta \in [40,100]) \\ (-1.06889/(\beta-31.0345)) \cdot \log(0.417419\beta - 25.4889) & (\beta > 100) \end{cases} \tag{6}$$

When dimension n is large ($n \geq 100$), runtime of pBKZ (second) is estimated as

$$\begin{aligned} &\mathrm{TimeBKZ}(n, \beta_t) \\ &= \sum_{\beta^{alg}=10}^{\beta_t} \sum_{t=1}^{\sharp tours} \left[2.5\cdot 10^{-4} \cdot \frac{n-\beta^{alg}}{250-\beta^{alg}} \cdot n^2 + 1.5\cdot 10^{-8} \cdot \beta^{alg} \cdot \sum_{i=1}^{n-1} \mathrm{ENUMCost}(B_i; \alpha, p)\right]. \end{aligned} \tag{7}$$

$\mathrm{ENUMCost}(B_i; \alpha, p)$ is the number of ENUM search nodes in pBKZ simulator:

$$\mathrm{ENUMCost}(B_i; \alpha, p) = p \cdot \frac{V_{\beta/2}(\alpha \cdot \mathrm{GH}(B_r))}{\prod_{i=\beta/2+1}^{\beta} \|\mathbf{b}_i^*\|_2} = 2\alpha^{-\beta/2} \cdot \frac{V_{\beta/2}(1) \cdot V_{\beta}(1)^{-1/2}}{r^{\beta^2/16}}.$$

Refer to [9] for details.

3.1.3.2 **Martin Albrecht et al.'s Method [3]**

In the experiments of [3], authors use BKZ 2.0 implemented in fplll (`https://github.com/fplll/fplll`) and fpylll (`https://github.com/fplll/fpylll`). They set the success probability of SVP oracle ENUM close to 1. Then they test the performance of BKZ-β under inequality 8 by fixing several smaller BKZ blocksize β and limiting number of local reduction tours. They replace SVP oracle in BKZ from ENUM with extreme pruning ($2^{o(n^2)-0.5n}$) to sieving ($2^{0.292n+o(n)}$) [28] for theoretical parameter estimation. They use the assumption in [12,2] that costs $8d \cdot 2^{0.292\beta+16.4}$ operations running BKZ-β of d-dimensional lattice.

3.1.4 Significance of Number of Samples in Practical Attack

At first we claim that because of the special case of our protocol: only one RLWE instance $(a, b = a \cdot s + e \bmod q) \in (R_q, R_q)$ is given, Kannan's embedding technique [27] and Liu-Nguyen's decoding attack [29] cannot be adopted since solving basis of lattice $L_{(\mathbf{A},q)} = \{\mathbf{v} \in \mathbb{Z}_q^m \mid \mathbf{v} \equiv \mathbf{Ax} \pmod q, \mathbf{x} \in D_\sigma^n\}$ is trivial when $m = n$. Therefore our estimator should be different from some other key exchange schemes as NewHope [6], BCNS [16] and Albrecht's estimator [3] etc. which regard RLWE and normal LWE problem as having the same difficulty. In practical attack, we can get only one n-dimensional RLWE instance, which can be amplified to $2 \cdot n + 1$ without changing the distribution of error vectors, see Algorithm 5. Therefore the lattice dimension in our case of solving RLWE is $d = 2n + 1$.

3.1.5 Our Simulator

For security analysis of our parameter choices, we refer to the approach in bold text in Figure 2. In Asiacrypt2017 [3], Albrecht et al. re-estimated the hardness of LWE problem using Kannan's embedding and Bai-Gal's embedding respectively under estimation in NewHope [6] (denoted as "2016 estimation"). 2016 estimation states that if the Gaussian Heuristic and the Geometric Series Assumption (GSA) [37] hold for BKZ-β reduced basis and

$$\sqrt{\beta/d} \cdot \|(\mathbf{e}|1)\|_2 \approx \sqrt{\beta}\sigma \leq \delta_0^{2\beta-d} \cdot \mathrm{Vol}(L_{(\mathbf{A},q)})^{1/d}. \tag{8}$$

then error e can be found by BKZ-β with root Hermite Factor δ_0. Equation 8 originates from NewHope [6] and was corrected in [3].

Using SIS $\rightarrow$ uSVP approach method to attack our protocol, we can get n samples by iterating only one given instance $z'' = a''s + f \in R_q$, therefore we need to evaluate the complexity of processing a $d = 2n + 1$ dimension basis. For BKZ reduction runtime estimation, we will give the result of progressive BKZ and Albrecht's BKZ with sieving estimator.

Step 1. We compute the complexity of BKZ-β with sieving SVP oracle estimated as $8d \cdot 2^{0.292\beta+16.4}$ double precision floating point operations [12,2]. We got the maximal record of 400×10^9 Floating Point Operations per second (FLOPS) per thread from the LINPACK benchmark test on 24-thread Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz (overclocked to 3.499GHz). We translate this to time (second) unit by

$$T_{BKZ} = 8d \cdot 2^{0.292\beta+16.4}(\text{FLOPs})/(400 \times 10^9/24(\text{FLOPS per thread})). \tag{9}$$

Simultaneously, T_{BKZ} can also be replaced by progressive BKZ simulator explained in section 3.1.3.1.

Then we compute security level of RLWE(n, q, σ_f) by:

$$\log_2(T_{BKZ}) + \log_2(2.7 \times 10^8). \tag{10}$$

Here 2.7×10^8 comes from the result of `RC5-72` benchmark test published on www.distributed.net, which means that above CPU can check 2.7×10^8 keys in second per thread. This method of converting runtime to security level was used by Aono et al. in [8].

Step 2. A short vector $\|\mathbf{b}_1\|_2 = \delta_0^d \cdot \det(\mathbf{B})$ is assumed to be inside of the BKZ-β reduced basis $\mathbf{B}$ of dimension d [17], where the root Hermite Factor is

$$\delta_0 = (((\pi\beta)^{1/\beta}\beta)/(2\pi e))^{1/(2(\beta-1))}. \tag{11}$$

We pre-compute the expected δ_0 for $\beta = 10, \cdots, n$ and rewrite equation 8 as

$$\sqrt{\beta}\sigma_f \leq \delta_0^{2\beta-d} \cdot \mathrm{Vol}(L_{(\mathbf{A},q)})^{1/d}. \tag{12}$$

In our case, $d = 2n+1$ and $\mathrm{Vol}(L_{(\mathbf{A},q)}) = q^n$. Therefore we can adapt inequality 12 to

$$\sqrt{\beta}\sigma_f \leq \delta_0^{2\beta-2n-1} \cdot q^{n/(2n+1)}. \tag{13}$$

We compute lower bound of σ_f in RLWE(n, q, σ_f) which covers security of AES-128/192/256 using equations 10, 11 and 13. Note that f no longer follows discrete Gaussian distribution (histogram shows similar shape as Gaussian distribution). Therefore we take a heuristic approach to estimate σ_f. We generate large amount of $as + 2e$ samples, then apply Round() and Recover() functions, giving us

$$z = \mathrm{Recover}(\mathrm{Round}(a \cdot s + 2e, p, q), p, q) = a \cdot s + 2f.$$

With $\frac{z-as}{2} = f$, we compute standard deviation σ_f. Results are given in Table 3.

Table 3. Lower Bound of σ_f in RLWE(n, q, σ_f) Covering Security of AES-128/192/256

Security level (n, q)	AES-128 (512,120833)		AES-192 (1024,120833)		AES-256 (1024,120833)	
Method	pBKZ	2016 estimation	pBKZ	2016 estimation	pBKZ	2016 estimation
Lower bound of σ_f	1.0061	4.9013	0.0068	0.9032	0.0186	3.9972
σ (for s and e) of our parameter choice	4.19		2.6			
σ_f	4.92		4.72			

Given a n-dimensional basis, in order to use pBKZ simulator, we need target β_t for our parameter choice. In section 3.1.5, with equations 11, 13 and parameter sets (n, σ, q), we can compute the root Hermite factor δ_0. At this stage, we can get the final target GSA constant $r_t = \delta_0^{-4d/(d-1)} = \delta_0^{-4-2/n}$, where $d = 2n+1$ is the dimension of lattice at step 2 in Algorithm 5. Hence we can compute the terminating block size

β_t corresponding to r_t, and set parameters α, r, p. Due to the uncertainty simulation for runtime with large dimension and large β (> 1000 and > 200 respectively), we are now sure about the simulation results for our key exchange protocol. However, our parameter choices can cover results from pBKZ simulator. Therefore we show results from pBKZ simulator in Table 3 as well. We will leave it as future work.

With lower bound of σ_f given in Table 3, we claim that parameter choices given in Table 1 cover security of AES-128/192/256 respectively, which satisfies NIST's security category I/III/V respectively.

3.2 Key Reuse Attack

In this section, we describe an efficient attack on reconciliation-based RLWE key exchange protocol. This attack only works for reconciliation-based RLWE key exchange protocols with reused keys. If the protocol does not allow key reuse, then this attack does not work. The number of queries required for the attack to find the exact value of the secret is roughly $2q$, which is extremely efficient. This is why our protocol is an ephemeral-only RLWE key exchange protocol with no key reuse. In addition, we show that a new construction can defeat such an attack that allows key reuse.

Fluhrer described an attack framework on error reconciliation-based Diffie-Hellman-like RLWE key exchange protocols in [23]. In this attack, an important setting is that key pair is reused, i.e. public and private keys of one party remain unchanged during the attack. Adversary can recover private key within polynomial time and queries. Ding et al. elaborated the attack on reconciliation-based RLWE key exchange protocol in [20]. This attack is an extension of [23]. [20] introduced a step-by-step attack with two extensions and a proof-of-concept attack implementation on [22] with reused keys and practical parameters. They show that this attack can successfully recover user's private key if key pair is reused. They also present a toy example and practical implementation of the attack. General idea of this attack works for other similar RLWE key exchange protocols since they share the same notion of error reconciliation and usage of signal function, despite exact approach to reconcile error is not the same.

3.2.1 Outline

Attacks in [23] and [20] use properties of signal function. Since signal value indicates which region a coefficient lies in, a smart approach to force signal value to reveal more information is introduced. The attack takes advantage of the number of times of a signal value changes to deduct the value of private key. We fix reused public key and private key of party j to be $p_j = a \cdot s_j + 2e_j$, public key of adversary as $p_{\mathcal{A}}$ and corresponding private key and error term are $s_{\mathcal{A}}$ and $e_{\mathcal{A}}$ respectively. Here we briefly recalls simplified version of the attack, where error terms e_j and $e_{\mathcal{A}}$ are not added to computation of $k_{\mathcal{A}}$ and k_j.

In this section we use the multiplication $k \cdot e$ for $k \in \mathbb{Z}$ and $e \in R_q$. We denote $s_j \in R_q$ as $s_j = s_j[0] + s_j[1]x + \cdots + s_j[n-1]x^{n-1}$.

In step 1, adversary $\mathcal{A}$ chooses private key $s_{\mathcal{A}}$ to be 0 and $e_{\mathcal{A}}$ to be the identity element 1 in R_q. $p_{\mathcal{A}} = ke_{\mathcal{A}}$, where $k \in [0, q-1]$. This gives $k_j = ks_j$. Adversary loops k from 0 to $q-1$ and executes the key exchange protocol with party j. We can see that with a looping k, adversary can make a correct guess on value of $s_j[i]$ based on the number of times the signal value $w_j[i]$ for each coefficient i of s_j. As k takes values from 0 to $q-1$, value of $k_j[i]$ changes in k multiples of $s_j[i]$ and there are changes in the signal value when $ks_j[i]$ is near the boundary values of signal regions defined in section 2.2. Therefore, there will be exactly $2s_j[i]$ number of changes in signal value for i-th coefficient of s_j. After this round, adversary can guess the value up to the $\pm$ sign, therefore in second round, more information is collected in order to deduct sign for each coefficient of s_j.

In step 2, adversary $\mathcal{A}$ chooses the same private key as step 1, but the public key is $(1+x) \cdot p_{\mathcal{A}}$. With this construction, adversary receives signal value of $p_{\mathcal{A}} \cdot ((1+x) \cdot s_j)$. Similar to the first round, adversary checks number of signal changes. Adversary can find out values of coefficients of $(1+x)s_j$, which are $s_j[0] - s_j[n-1]$, $s_j[1] + s_j[2]$, $\cdots$, $s_j[n-2] + s_j[n-1]$ up to the $\pm$ sign.

In step 3, consider pair of coefficients $s_j[0]$, $s_j[n-1]$, then by recovering the value of $s_j[0] - s_j[n-1]$ up to the $\pm$ sign in step 2 of the attack, and already knowing $s_j[0], s_j[n-1]$ values up to the sign from step 1 of the attack, adversary determines if $s_j[0]$ and $s_j[n-1]$ have same or opposite sign.

In step 4, adversary can repeat step 3 for every pair of coefficients $s_j[x], s_j[y]$, x from 0 to $n-2$, y from 1 to $n-1$ with the value of $s_j[x] + s_j[y]$ up to the $\pm$ sign from step 2 of the attack to determine if they have equal or opposite signs.

Lastly, adversary only need to guess the sign of $s_j[0]$. The rest of the coefficients follow since adversary has determined if every pair of coefficients $s_j[x], s_j[y]$ have equal or opposite signs. Adversary computes $p_j - a \cdot s_j$ and verifies the distribution of the result. If the guess on sign of $s_j[0]$ and other coefficients is correct, $p_j - a \cdot s_j$ should follows the distribution of e_j, which is discrete Gaussian distribution; otherwise, adversary simply flips the sign to obtain correct signs. This completes the attack.

In two extensions of the above attack, adversary shares same idea but slightly adjust the construction of the public key accordingly in order to recover the private key successfully. Here we omit details. An illustrated example in section 5 of [20] demonstrates the idea and effect of the attack intuitively. The number of queries required for the attack to find the exact value of the secret is estimated to be $2q$, where q is the modulus for R_q. The above attack can recover private key of [22] instantiated with practical parameters within 3.8 hours on a common PC.

3.2.2 Discussion

We note that this is exactly the reason why we stress that our Diffie-Hellman-like RLWE-based key exchange protocol does not allow key reuse. With the above attack, adversary can recover the private key very efficiently within only $2q$ queries. Our key exchange protocol is secure if public and private keys are used only once in each session. After key exchange is completed, both parties should delete the private key in case of potential attacks. For a new session, public key and private key should be newly generated. This is the exact same as ephemeral Diffie-Hellman key exchange protocol.

We also note that the attack does not directly work for our proposal since we apply our new rounding technique. However, the notion of attack is very important and we believe that a variant of this attack would work. In fact, this attack implies that for reconciliation-based key exchange protocols (including [22], [33], [16], [6] etc.), public and private keys should not be reused for such protocols.

3.2.3 Our Countermeasure

In this section, we described a work-in-progress RLWE-based key exchange protocol which can achieve secure key reuse [21]. The idea for this construction is using zero knowledge proof, where the secret term can be proven to be "small". The construction of key exchange describe in this section is based on the authentication protocol proposed in [21], where they first design 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. It is zero knowledge-based with negligible soundness and completeness errors. For concrete security proof of authentication mechanism and protocol description, please refer to [21].

Here we briefly introduce how to construct a reconciliation-based key reusable RLWE key exchange protocol. We note that the following protocol does not utilize our new rounding technique to reduce communication cost. The protocol demonstrates the idea of constructing key reusable reconciliation-based RLWE key exchange.

Let $H : \{0,1\}^* \to \{0,1\}^\tau$ be a hash function, where τ is the length of output of H. H is used for the commitment between the prover and verifier. This helps to make sure that the verifier commits to a message before receiving the value from the prover. Samp() is a function which generates polynomial in R_q using output of H according to distribution $D_{\mathbb{Z}^n,\sigma}$.

In order to prove zero knowledge, it is required that the commitment scheme is computationally hiding and unconditionally binding which ensures the integrity of the commitments.

A description of key exchange between party i and party j with reusable keys is given as follows:

Initiate Party i instantiates key exchange by computing public key $p_i = a \cdot s_i + 2e_i$, where s_i and e_i are sampled from $D_{\mathbb{Z}^n,\sigma}$. Send p_i to party j.

Response Party j computes public key $p_j = a \cdot s_j + 2e_j$, where s_j and e_j are sampled from $D_{\mathbb{Z}^n,\sigma}$. Sample e'_j and e''_j from $D_{\mathbb{Z}^n,\sigma}$. Compute $\text{Samp}(H(p_i)) \leftarrow D_{\mathbb{Z}^n,\sigma}$. Compute $\overline{p_i} = a \cdot \text{Samp}(H(p_i)) + 2e'_j + p_i$, key exchange material $k_j = \overline{p_i} \cdot s_j$, $w_j = \text{Sig}(k_j)$ and final shared key $sk_j = \text{Mod}_2(k_j, w_j)$. Send p_j and w_j to party i.

Finish Party i samples $e'_i \xleftarrow{\$} D_{\mathbb{Z}^n,\sigma}$, computes $H(p_i) \leftarrow D_{\mathbb{Z}^n,\sigma}$, key exchange material $k_i = p_j \cdot (s_i + \text{Samp}(H(p_i)))$ and final shared key $sk_i = \text{Mod}_2(k_i, w_j)$.

The protocol is illustrated in Figure 3:

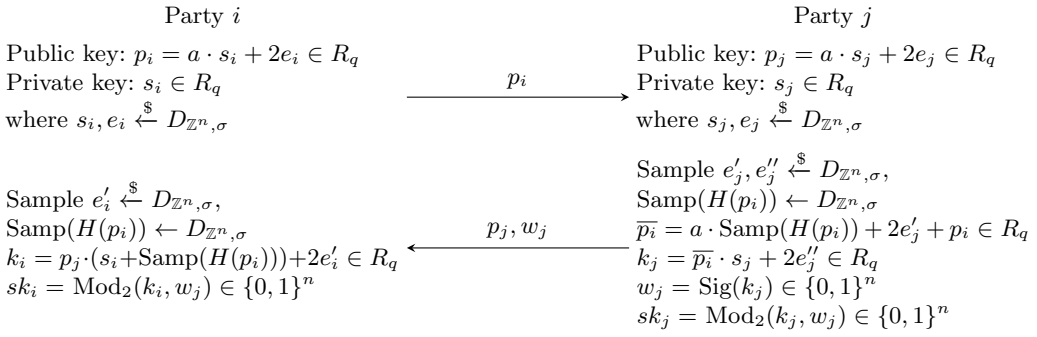

Fig. 3. RLWE key exchange protocol with reusable keys

With above construction and zero knowledge proof-based authentication mechanism, key reuse issue can be solved. Above construction is mainly for the purpose of presenting such a solution. Therefore we do not apply our new rounding technique and fresh a generation here. Correctness of the protocol can be analyzed using same approach as we showed in section 2.3. Compared with our proposal, the above protocol should have slightly different parameters and performance due to the additional $H(p_i)$ term.

4 Passive Security

4.1 Outline of IND-CPA for Our Protocol

For a probabilistic asymmetric encryption algorithm with key pairs: public key (PK) and private key (SK), indistinguishability under chosen plaintext attack (IND-CPA) is defined by the game between an adversary and a challenger. Let us use the notation $E(PK, M)$ to represent the encryption of a message M under the public key PK.

For schemes based on computational security, the adversary is modeled by a probabilistic polynomial time Turing machine that completes the game to output a result within a polynomial number of time steps:

1. *The challenger generates a key pair: public key (PK) and private key (SK) according to certain security parameter λ, and publishes PK to the adversary. The challenger keeps SK private.*
2. *The adversary may perform a polynomial number of encryptions or other operations.*
3. *Eventually, the adversary submits two distinct chosen plaintexts M_0, M_1 to the challenger.*
4. *The challenger selects a bit $b \in \{0, 1\}$ uniformly at random, and sends the challenge ciphertext $C = E(PK, M_b)$ back to the adversary.*
5. *The adversary is free to perform any number of additional computations or encryptions. Finally, it outputs a guess for the value of b.*

An asymmetric cryptosystem is **indistinguishable** under chosen plaintext attack if every probabilistic polynomial time adversary has only a negligible "advantage" over random guessing.

An adversary is deemed to have a negligible "advantage" if it wins the above game with probability

$$\left(\frac{1}{2}\right) + \epsilon(\lambda),$$

where $\epsilon(\lambda)$ is a negligible function in the security parameter λ, namely, for every (nonzero) polynomial function $poly()$, there exists λ_0 such that

$$|\epsilon(\lambda)| < \left|\frac{1}{poly(\lambda)}\right|$$

for all $\lambda > \lambda_0$.

Although the adversary knows M_0, M_1 and PK, the probabilistic nature of E means that the encryption of M_b will be only one of many valid ciphertexts, and therefore encrypting M_0, M_1 and comparing the resulting ciphertexts with the challenge ciphertext does not afford any negligible advantage to the adversary.

The above definition is specific to an asymmetric key cryptosystem, it can be adapted to the KEM case. Surely we can also impose it on a key exchange scheme,

though we think it is very **unnatural** due to the fact that in the key exchange scheme, we actually make the ciphertext first and then the plaintext.

4.2 Security Proof

We now define the passive security of our Diffie-Hellman-like ephemeral-only RLWE-based key exchange protocol defined in section 2.3. Notations are consistent with section 2.3. We start with the security of our key exchange protocol without rounding and recovering public key. We then discuss the hardness of our protocol defined in section 2.3.

Intuitively, any PPT adversary should not distinguish a real shared key ($sk \in \{0,1\}^n$) from a random one ($rand \xleftarrow{\$} \{0,1\}^n$) even if he gets the transcripts (public key and signal value) of the protocol, which in fact satisfies IND-CPA notion. More specifically, we define the advantage of an adversary $\mathcal{A}$:

$$|\Pr(\mathcal{A}(a, p_i, p_j, w_j, sk) = 1) - \Pr(\mathcal{A}(a, p_i, p_j, w_j, rand) = 1)|.$$

Definition 1. *We say a key exchange protocol is secure under passive adversary, if for any PPT adversary the advantage is negligible.*

We want the adversary to distinguish the final shared key $sk \in \{0,1\}^n$ from uniformly random one ($rand \xleftarrow{\$} \{0,1\}^n$), *i.e.* we prove that

$$|\Pr(\mathcal{A}(a, p_i, p_j, sk) = 1) - \Pr(\mathcal{A}(a, p_i, p_j, rand) = 1)|.$$

is negligible. Lemma 4 guarantees that this definition is sufficient.

Theorem 1. *The construction above is secure against passive PPT adversaries, if the RLWE assumption holds.*

Proof. We prove the security by a series of games. The first game $\mathbf{Game}_0$ is the real game which the adversary gets the real k_j, while the last game $\mathbf{Game}_4$ the adversary gets a uniformly random k_j. We show that the views of $\mathbf{Game}_0$ and $\mathbf{Game}_4$ are computational indistinguishable for any PPT adversaries, under the RLWE assumption.

$\mathbf{Game}_0$. This is the real game between the protocol challenger and the passive adversary $\mathcal{A}$. That is, the adversary obtains a, p_i, p_j, w_j, k_j, where $p_i = a \cdot s_i + 2e_i$, $p_j = a \cdot s_j + 2e_j$ and $k_j = p_i \cdot s_j$. Then $\mathcal{A}$ outputs a guess b'.

$\mathbf{Game}_1$. This game is identical to $\mathbf{Game}_0$ except that instead of setting $p_i = a \cdot s_i + 2e_i$ and $k_j = p_i \cdot s_j$, the challenger sets $p_i = b_i$ and $k_j = b_i \cdot s_j$, where $b_i \xleftarrow{\$} R_q$.

In lemma 6, we show that under the RLWE assumption, the views in $\mathbf{Game}_0$ and $\mathbf{Game}_1$ are computationally indistinguishable for any PPT passive adversaries.

$\mathbf{Game}_2$. This game is identical to $\mathbf{Game}_1$ except that instead of setting $p_j = a \cdot s_j + 2e_j$ and $k_j = b_i \cdot s_j$, the challenger sets $p_j = b_j$ and $k_j = u$, where $b_j \xleftarrow{\$} R_q$ and $u \xleftarrow{\$} R_q$.

We show the views for any PPT passive adversaries in $\mathbf{Game}_1$ and $\mathbf{Game}_2$ are computationally indistinguishable, if the RLWE assumption holds. The proof is given in lemma 7.

$\mathbf{Game}_3$. This game is identical to $\mathbf{Game}_2$ except that instead of setting $p_i = b_i$, the challenger sets $p_i = a \cdot s_i + 2e_i$.

In lemma 8, we prove the views in $\mathbf{Game}_2$ and $\mathbf{Game}_3$ are computationally indistinguishable, if the RLWE assumption holds.

$\mathbf{Game}_4$. This game is identical to $\mathbf{Game}_3$ except that instead of setting $p_j = b_j$, the challenger sets $p_j = a \cdot s_j + 2e_j$.

In lemma 9, we prove that the views in $\mathbf{Game}_3$ and $\mathbf{Game}_4$ are indistinguishable, if the RLWE assumption holds. The conclusion follows from lemma 6, 7, 8, 9 directly. □

Lemma 6. *Any PPT passive adversary cannot distinguish* $\boldsymbol{Game}_0$ *and* $\boldsymbol{Game}_1$*, if the RLWE assumption holds.*

Proof. We prove the lemma by showing that if there exists an adversary $\mathcal{A}$ who can distinguish $\mathbf{Game}_0$ and $\mathbf{Game}_1$, then we can construct another adversary $\mathcal{B}$ to distinguish the RLWE samples from uniform random. $\mathcal{B}$ works as follows. Once obtaining challenges $(a, b_i) \in R_q \times R_q$ from the RLWE oracle, where b_i is either $a \cdot s + 2e$ or uniformly random in R_q, $\mathcal{B}$ samples $s_j \xleftarrow{\$} D_{Z^n,\sigma}$ and sets $k_j = b_i \cdot s_j$ and computes $p_j = a \cdot s_j + 2e_j$. Finally $\mathcal{B}$ sends $(a, p_i = b_i, p_j, w_j, k_j)$ to $\mathcal{A}$. $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs. We note that $\mathcal{B}$ can compute w_j by himself.

If b_i is an RLWE sample, then what $\mathcal{A}$ obtains are exactly the same as in $\mathbf{Game}_0$, if b_i is uniformly random in R_q, then what $\mathcal{A}$ obtains are exactly the same as in $\mathbf{Game}_1$. This implies that if $\mathcal{A}$ can distinguish $\mathbf{Game}_0$ and $\mathbf{Game}_1$ with noticeable advantage, then $\mathcal{B}$ can distinguish RLWE samples from uniformly random with the same advantage. This finishes the proof. □

Lemma 7. *Any PPT passive adversary cannot distinguish* $\boldsymbol{Game}_1$ *and* $\boldsymbol{Game}_2$*, if the RLWE assumption holds.*

Proof. We prove this lemma by showing that if there exists an adversary $\mathcal{A}$ distinguishes $\mathbf{Game}_1$ and $\mathbf{Game}_2$, then we can construct a PPT adversary $\mathcal{B}$ to distinguish the RLWE samples from uniform. $\mathcal{B}$ works as follows. Once obtaining challenges $(a, b_j) \in R_q \times R_q$ and $(b_i, u) \in R_q \times R_q$, where u and b_i are either $a \cdot s + 2e, p_i \cdot s$ or uniformly random in R_q, $\mathcal{B}$ sets $p_i = b_i$, $p_j = b_j$ and $k_j = u$, and computes w_j. $\mathcal{B}$ sends (a, p_i, p_j, w_j, k_j) to $\mathcal{A}$, and outputs whatever $\mathcal{A}$ outputs. It is easy to see that if b_j, u are RLWE samples, then what $\mathcal{A}$ gets are exactly the same as in $\mathbf{Game}_1$; if b_j, u are uniformly random, then what $\mathcal{A}$ gets are exactly the same as in $\mathbf{Game}_2$. Therefore, if $\mathcal{A}$ can distinguish the two games with noticeable advantage, then $\mathcal{B}$ can break the RLWE problem with noticeable advantage. This complete the proof. □

Lemma 8. *Any PPT passive adversary cannot distinguish* $\boldsymbol{Game}_2$ *and* $\boldsymbol{Game}_3$*, if the RLWE assumption holds.*

Proof. The proof is similar to lemma 6, except we still choose k_j uniformly from R_q. □

Lemma 9. *Any PPT passive adversary cannot distinguish* $\boldsymbol{Game}_3$ *and* $\boldsymbol{Game}_4$*, if the RLWE assumption holds.*

Proof. The proof is similar to lemma 7, except we still choose k_j uniformly from R_q. □

Now we deal with the security regarding to rounding and recovering $a \cdot s + 2e$ in the next lemma.

Lemma 10. *For following two key exchange protocols:*

1. $p_i = a \cdot s_i + 2e_i, p_j = a \cdot s_j + 2e_j, k_i = p_j \cdot s_i, k_j = p_i \cdot s_j, w_j = Sig(k_j), sk_i = Mod_2(k_i, w_j), sk_j = Mod_2(k_j, w_j)$
2. $p_i = a \cdot s_i + 2e_i, p_j = a \cdot s_j + 2e_j, p'_i = Round(p_i, p, q), p''_i = Recover(p'_i, p, q), p'_j = Round(p_j, p, q), p''_j = Recover(p'_j, p, q), k_i = p''_j \cdot s_i, k_j = p''_i \cdot s_j, w_j = Sig(k_j), sk_i = Mod_2(k_i, w_j), sk_j = Mod_2(k_j, w_j)$

The hardness of computing final shared key of second protocol is at least as hard as computing final shared key of first protocol.

Proof. With publicly known algorithm Round() and Recover(), publicly known parameters and public terms p'_i, p'_j, any adversary can compute $p''_i \approx p_i$ and $p''_j \approx p_j$. However, $p''_i \neq p_i$, $p''_j \neq p_j$, Round() and Recover() function generate additional errors, which makes recovering private key s_i or s_j using transcripts from our key exchange at least no easier than using p_i, p_j or k_i, k_j to solve RLWE problem. □

Theorem 2. *Our security proof implies that our key exchange protocol is IND-CPA secure.*

Proof. As the adapted version of IND-CPA notion for our key exchange protocol in section 4.1 and our security proofs, we show that passive adversary cannot distinguish transcripts of our protocol from uniform random, giving passive adversary no additional advantage to break the protocol.

In addition, since we proved that final shared key sk generated from our protocol is uniformly random, if we replace either M_0 or M_1 defined in IND-CPA notion with a random string of n bits, adversary gains no additional advantage to break the protocol. □

5 Performance Analysis

In this section, we introduce our implementation and performance for the formalized KEM-API of our key exchange scheme in section 2.3. Note that in our implementation, a number in $\mathbb{Z}_q$ is represented as $[0, q-1]$. One can convert the regions defined for hint and signal functions from $\{-\frac{q-1}{2}, \cdots, \frac{q-1}{2}\}$ to corresponding regions in $[0, q-1]$.

5.1 On Number Theoretic Transform and Gaussian Sampler

As a number theoretic version of Fast Fourier Transform (FFT), the Number Theoretic Transformation (NTT) technique is a discrete Fourier transformation over $\mathbb{Z}_q$. Given the coefficients of two polynomials, $A(x)$ and $B(x)$ of degree no larger than n, the polynomial multiplication $A(x) \cdot B(x)$ costs $\Theta(n \log n)$ ring operations in $\mathbb{Z}_q$ using NTT, which is $O(n^2)$ by a naive multiplication method. The fast NTT with adapted butterfly operation is applied in Victor Shoup's NTL library [39], for doing polynomial operations as multiplication, division, GCD, factoring and so on. Hence we use NTL library in our implementation.

We use the Discrete Gaussian Sampler (DGS) based on Cumulative Distribution Table (CDT) in [32]. Here we briefly explain the general idea of CDT sampler. At first, it pre-computes a table of the discrete Gaussian Cumulative Distribution Function (CDF) values $F[i]$ for $i \in \{0, 1, \cdots, N-3\}$, as $0 = F[0] < F[1] < \cdots < F[N-3] = 1$, where $N = \tau \times \sigma$ with σ be the standard deviation and τ be the sampling precision parameter. Then it samples a parameter $r \in [0, 1)$ and find s s.t. r is located in the interval $F[s] \leq r < F[s+1]$ with probability $f[s] = F[s+1] - F[s]$. Here s is the index of the expected discrete Gaussian sample. Please find the details in [32]. DGS with CDT is an efficient Gaussian sampler at this moment according to the experimental analysis in [26].

5.2 Experimental Results

Our implementation uses C/C++ language and NTL library. We run 100,000 times experiments for NIST security category I, III and V, on a computer with Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz, running CentOS Linux release 7.4.1708, g++ version 6.3.0. Then we evaluate the average runtime for discrete Gaussian sampling (TimeDGS), polynomial multiplication (TimePM), key generation (TimeKeyPair), encapsulation (TimeEnc) and decapsulation (TimeDec) respectively. We show the experimental results in Table 4 with two decimal precision. We also note that large amount of time is spent on conversion between NTL library type variables and unsigned char variables in order to abide by NIST's API requirement since signal computation and error reconciliation are extremely efficient.

Table 4. Runtime (millisecond) of our NIST API Implementation

Security level	TimeDGS	TimePM	TimeKeyPair	TimeEnc	TimeDec
I	0.03	0.46	1.31	1.71	1.19
III/V	0.05	0.95	2.54	3.48	2.36

6 Advantages, Limitations and Applications

6.1 Advantages

Hardness from RLWE Problem. As our protocol is constructed based on the RLWE problem and properly instantiated RLWE instance is very hard to solve for both classic and quantum computers, this is clearly an important security advantage over classic key exchange protocols.

One RLWE Sample and Parameter Choices. As explained in our design, we would like to point out the fact that our parameter choices are based on the fact that the attackers have only **one** RLWE sample since our design is an ephemeral key exchange. It turns out in most literature, the practical attacks on RLWE are based on the access to multiple samples. Our observation allow us to study deeply into practical attacks and derive much smaller parameters. We show that our protocol and parameter choices remain strong against state-of-the-art cryptanalysis techniques. Our parameter choices can match security of AES-128/192/256 (NIST security category I/III/V) respectively.

Simple Design. Our protocol enjoys elegant and simple design, which makes the protocol practically efficient. As an error reconciliation-based key exchange protocol, an important advantage of our protocol over KEM-based ones is much simpler design and final shared key generation. Communication cost of ciphertext is much smaller as well. Computations of signal value and error reconciliation are more efficient than decapsulation (i.e. decryption) in KEM-based ones. Moreover, our parameter choices allow us to use NTT for polynomial multiplication, which can be implemented very efficiently in practice. Size of signal value is only 64 or 128 bytes for parameter choice $n = 512$ or $n = 1024$ respectively. The computation of signal value and error reconciliation is truly efficient.

Reduced Communication Cost. Our rounding technique gives smaller communication cost compared with similar RLWE/MLWE-based works including NewHope [6], NewHope-Simple [5] and Kyber [15] etc. at the same security level. As one of the main issues for RLWE-based constructions is larger communication cost compared with classic public key key exchange protocols (e.g. Diffie-Hellman and elliptic curve variants) and since communication cost is more "expensive" than computation cost in general, our protocol with smaller communication cost is more desirable.

Larger Final Shared Key. Our protocol generates a key of size n-bit, where n is a parameter of ring R_q. Our parameter choices suggest $n = 1024$ or $n = 512$. Therefore a 1024-bit or 512-bit key is generated during protocol execution. From the perspective of information theory, since key bits are uniformly random in $\{0, 1\}^n$, longer keys give much larger entropy than shorter keys (e.g. 256 bits). Therefore we believe that our key exchange protocol with longer final shared key is more desirable. Moreover,

from Grover's quantum algorithm's perspective, a 256-bit key gives roughly 128-bit security, whereas various works (e.g. Frodo [14], NewHope [6], NewHope-Simple [5] and Kyber [15] etc.) generate only 256-bit final shared key and claim much higher security level than 128-bit security. In this case, an adversary may choose Grover's algorithm to do an exhaustive key search on final shared key directly, instead of trying to solve lattice and other hard problems to break the protocol.

Forward Secure. Our protocol generates a more secure key than a usual encryption scheme since the shared key of previous sessions cannot be recovered simply by knowing the user's private key in the case of a usual public key encryption scheme. This is known as forward secrecy. If a private key of an encryption scheme is revealed, then adversary can recover all past communications simply by decrypting all past session keys. This does not work for our protocol and other ephemeral Diffie-Hellman-like ones, since each public and private pair is used only once. Adversary has to compromise every single session in order to decrypt all past sessions.

Key Exchange vs KEM. As a Diffie-Hellman-like key exchange protocol, an important advantage of our protocol over KEM-based ones is both parties negotiate and decide the final shared key together. For a KEM, since it is an encryption-based approach, one party decides the key, namely, party i encrypts a session key using public key of party j and party j decrypts the ciphertext using his private key. For our key exchange protocol, no single party can fully decide the final shared key. It can be only generated using one's private key and the other party's public key together. Therefore it is better than KEM-based approaches as a true key exchange. In addition, in a KEM, one must perform additional work to generate the shared keys.

Moreover, our protocol can be implemented efficiently and extended to various platforms, including but not limited to desktop processors, ARM, lower-end processors (microcontrollers etc.), Internet-of-Things (IoT) devices, hardware-based ones etc. There are various works that have demonstrate efficient error sampling and NTT implementation on such platforms, where these two parts take up most running time for our protocol. Since computation of signal and error reconciliation is very efficient, most expensive parts are sampling and polynomial multiplication.

6.2 Limitations

Compared with current classic key exchange protocols (e.g. ephemeral Diffie-Hellman and elliptic curve variants), one disadvantage is larger key size. Since public key and private key in RLWE-based constructions are degree n polynomials in R_q, common choice for degree n is 1024 or 512. Therefore RLWE-based constructions have larger communication cost. For Diffie-Hellman key exchange protocol, both parties exchange a l-bit public key, where l is recommended to be 2048 currently. Elliptic curve Diffie-Hellman has even smaller key size. We choose $n = 512$ and $n = 1024$, each coefficient

in public key is larger than 10 bits. This gives larger communication cost than Diffie-Hellman key exchange and its elliptic curve variant. Even so, our protocol has smaller communication cost than several similar works.

In addition, public and private keys cannot be reused for multiple sessions. As we stated in section 3.2, adversary may recover private key of our protocol with reused keys in roughly $2q$ queries. As we define our proposal as an "ephemeral-only Diffie-Hellman-like" RLWE-based key exchange protocol, therefore key reuse should be prohibited towards real world deployment of our protocol. Technique to defeat such attack is introduced in section 3.2.3. In latest version of Transport Layer Security protocol (TLS 1.3), RSA is removed from key exchange choices due to concerns regarding to forward secrecy, where in current TLS 1.2, static RSA is used as KEM for session key encapsulation. This move implies that ephemeral-only key exchange (ephemeral Diffie-Hellman and elliptic curve variant) is more preferred than KEM (encryption-based) ones. We believe this idea also can be applied to post-quantum variants, where our key exchange protocol is ephemeral-only and Diffie-Hellman-like. Therefore, for real world deployment of our protocol, key reuse should be forbidden.

6.3 Applications

Our proposal is a RLWE variant of Diffie-Hellman key exchange and its elliptic curve variants. Therefore it is a quantum-resistant drop-in candidate for protocols and real world applications. Currently, there are various protocols and applications have taken advantage Diffie-Hellman key exchange protocol and its variants. We believe our proposal is a drop-in replacement of quantum-vulnerable key exchange protocols. A few important real world applications can adopt our post-quantum key exchange protocol, including but not limited to:

- Transport Layer Security (TLS) and related protocols (QUIC, HTTPS, IMAPS, SMTPS etc.)
- Secure Shell (SSH), SSH File Transfer Protocol (SFTP)
- Internet Key Exchange (IKE) in Internet Protocol Security (IPsec)
- Virtual Private Networks (VPN)
- Other applications where key exchange is integrated (e.g. Secure messaging/video calling applications, end-to-end secure applications, client-server applications etc.)

Above protocols and applications are widely deployed in real world currently, including secure communication, online banking and e-commerce, secure remote access, point-to-point secure applications etc. Note that for ephemeral Diffie-Hellman key exchange, each party needs to only transmit public key to the other party. Similarly for our proposal, both parties exchange public key alongside an additional signal value, which has very low additional communication cost. We believe that our proposal is a secure, efficient and lightweight drop-in replacement for current Diffie-Hellman key exchange protocol.

References

1. Ajtai, M., Kumar, R., Sivakumar, D.: A sieve algorithm for the shortest lattice vector problem. In: Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing. pp. 601–610. STOC '01 (2001)
2. Albrecht, M.R.: On dual lattice attacks against small-secret LWE and parameter choices in helib and SEAL. In: Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II. pp. 103–129 (2017)
3. Albrecht, M.R., Göpfert, F., Virdia, F., Wunderer, T.: Revisiting the expected cost of solving usvp and applications to lwe. ASIACRYPT (2017)
4. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Journal of Mathematical Cryptology 9(3), 169–203 (2015)
5. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Newhope without reconciliation. IACR Cryptology ePrint Archive 2016, 1157 (2016)
6. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key exchange-a new hope. In: USENIX Security Symposium. pp. 327–343 (2016)
7. Alperin-Sheriff, J., Apon, D.: Dimension-preserving reductions from lwe to lwr. IACR Cryptology ePrint Archive 2016, 589 (2016)
8. Aono, Y., Boyen, X., Phong, L.T., Wang, L.: Key-private proxy re-encryption under LWE. In: Progress in Cryptology - INDOCRYPT 2013 - 14th International Conference on Cryptology in India, Mumbai, India, December 7-10, 2013. Proceedings. pp. 1–18 (2013)
9. Aono, Y., Wang, Y., Hayashi, T., Takagi, T.: Improved Progressive BKZ Algorithms and Their Precise Cost Estimation by Sharp Simulator, pp. 789–819. Springer Berlin Heidelberg (2016)
10. Bai, S., Galbraith, S.D.: Lattice decoding attacks on binary LWE. In: Information Security and Privacy - 19th Australasian Conference, ACISP July, 2014. Proceedings. pp. 322–337 (2014)
11. Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom functions and lattices. Advances in Cryptology–EUROCRYPT 2012 pp. 719–737 (2012)
12. Becker, A., Ducas, L., Gama, N., Laarhoven, T.: New directions in nearest neighbor searching with applications to lattice sieving. In: Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016. pp. 10–24 (2016)
13. Bogdanov, A., Guo, S., Masny, D., Richelson, S., Rosen, A.: On the hardness of learning with rounding over small modulus. In: Theory of Cryptography Conference. pp. 209–224. Springer (2016)
14. Bos, J., Costello, C., Ducas, L., Mironov, I., Naehrig, M., Nikolaenko, V., Raghunathan, A., Stebila, D.: Frodo: Take off the ring! practical, quantum-secure key exchange from lwe. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 1006–1018. ACM (2016)
15. Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Stehlé, D.: Crystals–kyber: a cca-secure module-lattice-based kem. Tech. rep., Cryptology ePrint Archive, Report 2017/634, 2017. http://eprint. iacr. org/2017/634 (2017)
16. Bos, J.W., Costello, C., Naehrig, M., Stebila, D.: Post-quantum key exchange for the tls protocol from the ring learning with errors problem. In: Security and Privacy (SP), 2015 IEEE Symposium on. pp. 553–570. IEEE (2015)
17. Chen, Y.: Lattice reduction and concrete security of fully homomorphic encryption. Dept. Informatique, ENS, Paris, France, PhD thesis (2013)
18. Chen, Y., Nguyen, P.Q.: Bkz 2.0: Better lattice security estimates. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 1–20. Springer (2011)
19. Diffie, W., Hellman, M.: New directions in cryptography. IEEE transactions on Information Theory 22(6), 644–654 (1976)

20. Ding, J., Alsayigh, S., Saraswathy, R., Fluhrer, S., Lin, X.: Leakage of signal function with reused keys in rlwe key exchange. In: Communications (ICC), 2017 IEEE International Conference on. pp. 1–6. IEEE (2017)
21. Ding, J., Saraswathy, R.: Rlwe-based authentication using rounding signals. University of Cincinnati preprint (2017)
22. Ding, J., Xie, X., Lin, X.: A simple provably secure key exchange scheme based on the learning with errors problem. IACR Cryptology EPrint Archive 2012, 688 (2012)
23. Fluhrer, S.R.: Cryptanalysis of ring-lwe based key exchange with key share reuse. IACR Cryptology ePrint Archive 2016, 85 (2016)
24. Gama, N., Nguyen, P.Q.: Predicting lattice reduction. In: Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings. pp. 31–51 (2008)
25. Herold, G., Kirshanova, E., May, A.: On the asymptotic complexity of solving LWE. IACR Cryptology ePrint Archive 2015, 1222 (2015)
26. Howe, J., Khalid, A., Rafferty, C., Regazonni, F., O'Neill, M.: On practical discrete gaussian samplers for lattice-based cryptography. IEEE Transactions on Computers (2016)
27. Kannan, R.: Minkowski's convex body theorem and integer programming. Mathematics of operations research 12(3), 415–440 (1987)
28. Laarhoven, T., de Weger, B.: Faster sieving for shortest lattice vectors using spherical locality-sensitive hashing. In: Progress in Cryptology - LATINCRYPT 2015 - 4th International Conference on Cryptology and Information Security in Latin America, Guadalajara, Mexico, August 23-26, 2015, Proceedings. pp. 101–118 (2015)
29. Liu, M., Nguyen, P.Q.: Solving BDD by enumeration: An update. In: Topics in Cryptology - CT-RSA 2013 - The Cryptographers' Track at the RSA Conference 2013, February, 2013. Proceedings. pp. 293–309 (2013)
30. Lyubashevsky, V.: The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In: Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th InternationalWorkshop on Randomization and Computation, RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings. pp. 378–389 (2005)
31. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 1–23. Springer (2010)
32. Peikert, C.: An efficient and parallel gaussian sampler for lattices. In: Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings. pp. 80–97 (2010)
33. Peikert, C.: Lattice cryptography for the internet. In: International Workshop on Post-Quantum Cryptography. pp. 197–219. Springer (2014)
34. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM (JACM) 56(6), 34 (2009)
35. Ritt, J.F.: Permutable rational functions. Transactions of the American Mathematical Society 25(3), 399–448 (1923)
36. Schnorr, C.P., Euchner, M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Mathematical Programming 66(1), 181–199 (Aug 1994)
37. Schnorr, C.P.: Lattice reduction by random sampling and birthday methods. In: STACS. vol. 2607, pp. 145–156. Springer (2003)
38. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM review 41(2), 303–332 (1999)
39. Shoup, V.: NTL, a library for doing number theory (2017), available at `http://www.shoup.net/ntl/`

40. Stephens-Davidowitz, N.: Discrete gaussian sampling reduces to cvp and svp. In: Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 1748–1764. Society for Industrial and Applied Mathematics (2016)