Efficient KEA-Style Lattice-Based Authenticated Key Exchange

Zilong Wang and Honggang Hu

Key Laboratory of Electromagnetic Space Information,
Chinese Academy of Sciences,
School of Information Science and Technology,
University of Science and Technology of China,
Hefei, China, 230027
{wzl0830@mail.ustc.edu.cn
hghu2005@ustc.edu.cn}

Abstract. Lattice-based cryptographic primitives are believed to have the property against attacks by quantum computers. In this work, we present a KEA-style authenticated key exchange protocol based on the ring learning with errors problem whose security is proven in the BR model with weak perfect forward secrecy. With properties of KEA such as implicit key authentication and simplicity, our protocol also enjoys many properties of lattice-based cryptography, namely asymptotic efficiency, conceptual simplicity, worst-case hardness assumption, and resistance to attacks by quantum computers. Our lattice-based authenticated key exchange protocol is more efficient than the protocol of Zhang et al. (EUROCRYPT 2015) with more concise structure, smaller key size and lower bandwidth. Also, our protocol enjoys the advantage of optimal online efficiency and we improve our protocol with pre-computation.

Keywords: lattice-based cryptography, authenticated key exchange, post-quantum cryptography, ring-LWE

1 Introduction

1.1 KE and AKE

Key exchange (KE) is one of the most fundamental cryptographic primitives. In practice, a common secret key (session key) generated by their personal keys (static key) should be shared with KE before the session starts, as the network is considered to be insecure. The communication data will later be transmitted on a trusted channel established with the session key. An authenticated key exchange (AKE) is quite similar to KE while AKE provides authentication which can avoid man-in-the-middle attack.

AKE can be divided into explicit AKE and implicit AKE according to the technique that achieves authentication. Explicit AKE always needs extra cryptographic primitives such as signatures, message authentication codes, or hash functions to provide authentication, which brings additional computation and

communication overhead and makes the protocol more complicated. The IKE [22], SIGMA [24], SSL [16], TLS [13], JFK [2] are all explicit AKEs. Implicit AKE achieves authentication by ingenious design of the algebraic structure. The KEA [33], OPACITY [32], MQV [29], HMQV [23] and OAKE [34] are families of implicitly AKE.

Intuitively, an AKE is secure if no probabilistic polynomial time (PPT) adversary is able to extract any useful information from the data exchanged during the session. Formally, the widely used security models for AKE include the BR model [4, 35], the CK model [10] and the ACCE model [20]. The BR model, which is introduced by Bellare and Rogaway, is an indistinguishability-based security model. The CK model, which accounts for scenarios in which the adversary can obtain information about a party's static secret key or a session state, inherits from Krawczyk's SIGMA family of protocols. The ACCE model is a variant of the BR model which has separated properties for entity authentication and channel security.

Another property of AKE protocols is perfect forward secrecy (PFS). PFS requires that an adversary who corrupts one of the parties can not destroy the security of previous sessions. However, Krawzcyk [23] showed that no 2-pass AKE protocol can achieve PFS. Alternatively, he presented a notion of weak perfect forward secrecy (wPFS) which says that the session key of an honest session remains secure if the static keys are compromised after the session is finished.

1.2 Lattice-Based Cryptosystems and Key Reused Attack

It is important to construct protocols based on lattice problems as lattice-based cryptography is believed to resist quantum computers attacks. For instance, a post-quantum cryptography competition is held by NIST to advance the process of post-quantum cryptography standard.

The most widely used lattice problem to construct lattice-based cryptography is the learning with errors (LWE) problem which was first proposed by Regev [31] as an extension of learning parity with noise (LPN) problem [5]. Later in [28], Lyubashevsky et al. introduced the ring-LWE which is the ring -based analogue of LWE, and proved the hardness of ring-LWE. (ring-)LWE has attracted a lot of attention in theories and applications due to its good asymptotical efficiency, strong security, and exquisite construction. LWE has been used to construct public-key encryption [19, 26, 31], identity-based encryption [1, 11], key exchange [3, 7, 14, 35, 21], and fully homomorphic encryption [8, 9], etc.

1.3 Techniques and Relation to KEA

The key idea behind our protocol, which was firstly proposed by Linder et al. [26], is that the two parties share a common secret: $I(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}A\boldsymbol{y}$, where $\boldsymbol{x}$ and $\boldsymbol{y} \in \mathbb{Z}_q^n$ are the static keys of two parties, and A is randomly chosen from $\mathbb{R}_q^{n\times n}$. When it comes to ring-LWE, the form is simpler: $I(x, y) = xay$, where x and $y \in \mathbb{R}$ are the static keys of two parties, and a is randomly chosen from $\mathbb{R}$.

The definition of ring-LWE indicates that it will bring some small errors during the session. These small errors may be beneficial in security, but they make the protocol incorrect. A common method to deal with these errors is reconciliation mechanism which was first proposed by Ding et al. [14] and soon improved with a more bandwidth-efficient and unbiased one presented by Peikert [30]. In [3], Erdem et al. proposed a more efficient reconciliation mechanism based on a varying error distribution at the expense of security. In [21], Jin et al. formalized reconciliation mechanism as a black-box called key consensus (KC) and gave the upper bound on parameters for any KC. What's more, they designed a KC called OKCN which can achieve the upper bound.

Our AKE protocol is inspired by KEA which was designed by the NSA and later standardized by the NIST. However, they are very different in the underlying algebraic structures. In the original KEA protocol, the shared key is $H_K(A^y \oplus X^b)$. Later in [25], Lauter et al. improved the original KEA with a provably secure version KEA$^+$. Throughout our work, we simply refer KEA to the provably secure version KEA$^+$. Formally, let G be a cyclic group with generator $g \in G$ and $|G| = n$. Randomly choose s_i, $s_j \in \{1, ..., n\}$ as the static keys of Party i and Party j. The specification of KEA is given in Fig. 1, where H is a hash function.

Party i		Party j
PK: $P_i = g^{s_i} \in G$		PK: $P_j = g^{s_j} \in G$
SK: $s_i \in \{1, ..., n\}$		SK: $s_j \in \{1, ..., n\}$
$X = g^x$		
x randomly chosen from $\{1, ..., n\}$	$\xrightarrow{i,X}$	$Y = g^y$
	$\xleftarrow{j,Y}$	y randomly chosen from $\{1, ..., n\}$
$K_i = H(Y^{s_i}, P_j^x, i, j)$		$K_j = H(P_i^y, X^{s_j}, i, j)$

Fig. 1. Specification of KEA

As shown in [34], KEA enjoys the advantage of optimal online efficiency. The separation of two exponentiations, which allow off-line pre-computation, makes KEA much more desirable for deployments on low-power devices, such as smart cards and phones over wireless setting. Take Party i as an example, Party i pre-computes X and P_j^x before session starts, where j is one of the potential parties which Party i may communicate with.

Thanks to the simplicity of KEA, there is no complicated computation for each party to compute a closed value for reconciliation mechanism. Compared to the protocol in [35], the error of these two values is smaller. Consequently, a smaller q is sufficient for ensuring the correctness of reconciliation mechanism, which has two advantages: 1) A smaller q can reduce the bit length of public keys and the bandwidth as they are proportional to $log\, q$; 2) For any fixed error

rate α, higher security level can be achieved with a smaller q as the security of (ring-)LWE is partially dependent on the ratio of q and α.

In KEA, the $X^{s_j} = g^{xs_j}$ do not reveal any information about s_j even x is chosen by adversary. However, it is well-known that the Regev's encryption [31] is not chosen-ciphertext attack (CCA) secure. This vulnerability was utilized by Ding et al. [15] who show an attack depends on the leakage of signal function [14]. The adversary can extract the static key of target party after $2q$ queries. But this type of attack is inefficient to our protocol for two reasons: 1) Similar as PKI, the public keys of parties will be updated periodically, and there are no enough queries for adversary to extract the static keys; 2) The signal function in our protocol is probabilistic polynomial time algorithm. It is more difficult for adversary to decide the period of the signal value during the attack.

1.4 Related Works and Our Contributions

The raise of attention to post-quantum cryptography stimulates more constructions of lattice-based AKE protocols in the last few years. A passive-secure KE protocol based on (ring-)LWE, which was proposed by Ding et al., is translated from standard Diffie-Hellman protocol [14]. The most significant contribution of Ding' work is that they proposed the concept of reconciliation mechanism to deal with the errors. Fujioka et al. [17] proposed a generic construction of AKE from CCA secure KEMs, which can be proven secure in the CK^+ model. However, their construction was just of theoretic interest. In [18], Fujioka et al. gave a more practical AKE protocol which can be constructed from any one-way CCA secure KEM in the random oracle model. In [30], Peikert presented an efficient key encapsulation mechanism (KEM) based on ring-LWE, and then translated it into an AKE protocol using the SIGMA-style structure. After that, Bos et al. [7] utilized Peikerts KEM as a DH-like KE protocol, and integrated it into the TLS protocol. Strictly speaking, their protocol is not a lattice-based AKE protocol because classical signatures were employed to provide explicit authentication. Alkim et al. [3] then improved the performance of Peikert's KEM to make the AKE protocol more practical, and their new protocol that called NewHope was applied to the Google's browser Chrome. It is the first post-quantum AKE protocol adopted by real world. Zhang et al. [35] proposed the first lattice-based implicit AKE whose structure is similar as HMQV. In [21], Jin et al. introduced the notion of key consensus (KC) as a tool and presented generic constructions of KE based on KC.

A rough comparison of lattice-based AKEs is given in Table 1. A more detailed comparison between our protocols and the protocol in [35] is showed in Table 2 as they are very similar and are all implicit AKE protocols.

1.5 Organization

Section 2 presents some basic notations and facts. The AKE protocol based on ring-LWE problem is given in Section 3 and the analysis of the protocol from Section 3 is given in Section 4. In Section 5, a new protocol which is more efficient

Table 1. Comparison of lattice-based AKEs.

Protocols	KEM/PKE	Signature	Message-pass	Security	Num. of Rings
Fujioka [7]	CCA	-	2-pass	CK^+	$\gg 7$
Fujioka [3]	OW-CCA	-	2-pass	CK^+	7
Peikert [30]	CPA	EUF-CMA	3-pass	SK-security	$> 2^{(1)}$
Bos [7]	CPA	EUF-CMA	4-pass	ACCE	2
Alkim [3]	CPA	EUF-CMA	4-pass	-	$2 + x^{(2)}$
Zhang [35]	-	implicit	2-pass	BR with wPFS	2
Ours	-	implicit	2-pass	BR with wPFS	2

(1) 2 ring elements for KEM and more for the concrete lattice-based signatures.
(2) The actual number of ring elements depends on the signature, and it can be a traditional signature, so $x \geq 0$

Table 2. Comparison between ours and Zhang's protocol [35] with 80 bits security. mult. refers to the total number of multiplications over rings.

Protocols	n	α	$\log q$	mult.	pk	sk	init. msg	resp. msg
Zhang [35]	1024	3.397	45	4	5.625 KB	1.5 KB	5.625 KB	5.75 KB
Section 3	1024	3.192	30	3	3.75 KB	0.75 KB	3.75 KB	4 KB
Section 4	1024	3.192	30	1	3.75 KB	0.75 KB	3.75 KB	4 KB

for the Internet is considered. In the last section, we analyze the concrete choices of parameters along with the consideration of their security.

2 Preliminaries

2.1 Notation:

Let κ be the security parameter. Bold capital letters denote matrices. Bold lowercase letters denote vectors. For any integer q, let $\mathbb{Z}_q$ denote the quotient ring $\mathbb{Z}/q\mathbb{Z}$. We use $a \leftarrow_r B$ to denote that a is an element randomly chosen from B, where B is a distribution or a finite set. When we say that a function $f(x)$ is negligible, we mean that for every $c > 0$, there exists a X satisfies: $f(x) < 1/x^c$ for all $x > X$. The statistical distance between two distributions, X and Y, over some finite set S is defined as:

$$\Delta(X, Y) = \frac{1}{2} \sum_{s \in S} |Pr(X = s) - Pr(Y = s)| .$$

If $\Delta(X, Y)$ is negligible, we say that X and Y are statistically indistinguishable.

2.2 Lattice and Gaussian Distributions

A lattice always connects to a matrix $\boldsymbol{B}$, and it is finitely generated as the integer linear combinations of the column vectors of $\boldsymbol{B} = \{\boldsymbol{b}_1, ..., \boldsymbol{b}_k\}$:

$$\mathcal{L} = \mathcal{L}(\boldsymbol{B}) = \{\sum_{i=1}^{n} z_i \boldsymbol{b}_i : z_i \in \mathbb{Z}\} .$$

The integer n is called the rank of the basis, and it is an invariant of the lattice. For any positive integer n and real $s > 0$, define the Gaussian function of parameter s as:

$$\rho_s(\boldsymbol{x}) = exp(-\pi \|\boldsymbol{x}\|/s^2) .$$

We define a Gaussian distribution over lattice $\mathcal{L}$ as:

$$D_s(\boldsymbol{x}) = \rho_s(\boldsymbol{x})/\rho_s(\mathcal{L}) .$$

where $\rho_s(\mathcal{L}) = \sum_{\boldsymbol{x} \in \mathcal{L}} \rho_s(\boldsymbol{x})$.

Fact 1 *Let χ denote the Gaussian distribution with standard deviation σ and mean zero. Then, for all $C > 0$, it holds that:*

$$Pr(e \leftarrow_r \chi : |e| > C \cdot \sigma) \leq \frac{2}{C\sqrt{2\pi}} exp(-C^2/2) .$$

This fact shows that the samples from χ are around mean with very high probability. Specially, for $C = 10$, the probability is less than 2^{-70}.

Fact 2 *Let $x, y \in R$ be two polynomials whose coefficients are distributed according to a discrete Gaussian distribution with standard deviation σ and τ, respectively. The individual coefficient of the $x \cdot y$ is then normally distributed with standard deviation $\sigma\tau\sqrt{n}$, where n is the degree of the polynomial.*

Let the integer n be a power of 2. For any positive integer, we define the ring $R = \mathbb{Z}[x]/(x^n + 1)$, and $R_q = \mathbb{Z}_q[x]/(x^n + 1)$. Obviously, the discrete Gaussian distribution over the ring R can be naturally defined as the distribution of ring elements whose coefficients are distributed according to the discrete Gaussian distribution over $\mathbb{Z}^n$. Consequently, for any $x \in R_q$, we define $x \leftarrow_r \chi_\alpha$ that we sample x whose coefficients are distributed according to χ_α.

Define R_q as above. For any $s \in R_q$, the ring-LWE distribution $A_{s,\chi}$ over $R_q \times R_q$ is sampled by choosing $a \leftarrow_r R_q$ at random, choosing $e \leftarrow_r \chi$ and e is independent of a, and outputting $(a, b = s \cdot a + e \, mod \, q)$.

Definition 1. *Let $A_{s,\chi}$ be defined as above. Given m independent samples $(a_i, b_i) \in R_q \times R_q$ where every sample is distributed according to either: (1) $A_{s,\chi}$ for a uniformly random $s \leftarrow_r R_q$ (fixed for all samples), or (2) the uniform distribution, no PPT algorithm can distinguish, with non-negligible probability, which distribution they are chosen from.*

The ring-LWE assumption can be reduced to some hard lattice problems such as the Shortest Independent Vectors Problem (SIVP) over ideal lattices [28]:

Lemma 1. *(Hardness of the Ring-LWE Assumption) Let n be a power of 2 and α be a real number in $(0,1)$. Let q and R_q be defined as above. Then there exists a polynomial time quantum reduction from $O(\sqrt{n}/\alpha) - SIVP$ in the worst case to average-case ring-LW$E_{q,\beta}$, where $\beta = \alpha q \cdot (n\ell = log(n\ell))^{1/4}$.*

In [28], Lyubashevsky et al. showed that the ring-LWE assumption still holds even if s is chosen according to the error distribution χ_β rather than uniformly.

2.3 Reconciliation Mechanism

Reconciliation mechanism was first proposed by Ding et al. [14] and later be reconstructed by a series of works [3, 21, 30]. It enables two parties to extract identical information from two "almost" same elements σ_1 and $\sigma_2 \in \mathbb{Z}_q$. In our protocol, the reconciliation mechanism OKCN [21] is adopted, and a brief description of OKCN is given as follows.

The OKCN consists of two algorithms (Con, Rec) which have parameters q (dominating security and efficiency), m (parameterizing range of consensus key), g (parameterizing bandwidth), and d (parameterizing error rate). Define $params = (q, m, g, d, aux)$ where $aux = (q' = lcm(q, m), \alpha = q'/q, \beta = q'/m)$. The probabilistic polynomial time algorithm Con takes a security parameter $(\sigma_1, params = (q, m, g, d))$ as input and outputs (k_1, ω) where $k_1 \in \mathbb{Z}_m$ is the shared value and $\omega \in \mathbb{Z}_g$ is the signal that will be publicly delivered to the communicating peer. The deterministic algorithm Rec, on input $(\sigma_2, \omega, params)$, outputs k_2 which is identical to k_1 with overwhelming probability. The details of OKCN are presented in Algorithm 1.

Algorithm 1 Reconciliation Mechanism: OKCN

function CON($\sigma_1, params$)
  $e \leftarrow [-\lfloor(\alpha-1)/2\rfloor, \lfloor\alpha/2\rfloor]$
  $\sigma_A = (\alpha\sigma_1 + e) \bmod q'$
  $k_1 = \lfloor\sigma_A/\beta\rfloor$
  $\omega = \lfloor(\sigma_A \bmod \beta)g/\beta\rfloor \in \mathbb{Z}_g$
  return (k_1, ω)
end function
function REC($\sigma_2, \omega, params$)
  $k_2 = \lfloor\alpha\sigma_2/\beta - (\omega + 1/2)/g\rceil \bmod m$
  return k_2
end function

Lemma 2. *For OKCN: 1) k_1 and ω are independent, and k_1 is uniformly distributed over $\mathbb{Z}_m$, whenever $\sigma_1 \leftarrow \mathbb{Z}_q$; 2) If the system parameters satisfy $(2d+1)m < q(1-1/g)$ where $m \geq 2$ and $g \geq 2$, then the OKCN is correct $(k_1 = k_2)$.*

2.4 Security Model

The BR security model, which is one of the most common models for KE protocol, is usually strong enough for many practical applications. It was first proposed by Bellare and Rogaway in [4], and later in [6], the BR model was extended to adapt to the public-key setting.

A protocol is a pair of functions $P = (\Pi, \mathcal{G})$, where Π specifies how parties behave and $\mathcal{G}$ generates keys for each party. For an AKE protocol, define N to be the maximum number of parties in the AKE protocol. Each party is identified by an integer $i \in \{1, 2, 3, ..., N\}$. A single run of the protocol is called a session. A session starts with message (ID, I, i, j) or (ID, R, j, i, X_i), where ID is the identification of the protocol, and I and R stand for the party's roles. We define session identifier for the session activated by message (ID, I, i, j) as $sid = (ID, I, i, j, X_i, Y_j)$ and session identifier for the session activated by message (ID, R, j, i, X_i) as $sid = (ID, R, j, i, X_i, Y_j)$. A session is said to be completed when its owner successfully computes a session key. The matching session of $sid = (ID, I, i, j, X_i, Y_j)$ is the session with identifier $\widetilde{sid} = (ID, R, j, i, X_i, Y_j)$.

We adopt the technique in [35] to describe the adversarial capabilities: an adversary, $\mathcal{A}$, is a PPT Turing Machine taking the security parameter 1^k as input. We allow $\mathcal{A}$ to make six types of queries to simulate the capabilities of $\mathcal{A}$ in the real world.

- $\boldsymbol{Send}_0(ID, I, i, j)$: $\mathcal{A}$ activates Party i as an initiator. The oracle returns a message X_i intended for Party j.
- $\boldsymbol{Send}_1(ID, R, j, i, X_i)$: $\mathcal{A}$ activates Party j as a responder using message X_i. The oracle returns a message Y_j intended for Party i.
- $\boldsymbol{Send}_2(ID, R, i, j, X_i, Y_j)$: $\mathcal{A}$ sends Party i the message Y_j to complete a session previously activated by a $\boldsymbol{Send}_0(ID, I, i, j)$ query that returned X_i.
- $\boldsymbol{SessionKeyReveal}(sid)$: The oracle returns the session key in the session sid if it has been generated.
- $\boldsymbol{Corrupt}(i)$: The oracle returns the static secret key of Party i. A party whose key is given to $\mathcal{A}$ in this way is called dishonest; a party who does not compromise in this way is called honest.
- $\boldsymbol{Test}(sid^*)$: The oracle chooses a bit $b \leftarrow_r \{0, 1\}$. If $b = 0$, it returns a key chosen uniformly at random; if $b = 1$, it returns the session key associated with sid^*. We only allow $\mathcal{A}$ to query this oracle once, and only on a **fresh** session sid^*.

Definition 2. *(Freshness): Let* $sid^* = (ID, I, i^*, j^*, X_i, Y_j)$ *or* $(ID, R, j^*, i^*, X_i, Y_j)$ *be a completed session with initiator Party* i^* *and responder Party* j^**. We say that* sid^* *is fresh under the following conditions:*

(1) $\mathcal{A}$ *has not made a* $\boldsymbol{SessionKeyReveal}$ *query on* sid^**.*
(2) $\mathcal{A}$ *has not made a* $\boldsymbol{SessionKeyReveal}$ *query on* sid^**'s matching session.*
(3) Neither Party i^* *nor Party* j^* *is dishonest if* sid^**'s matching session does not exist.*

Security Definition: The adversary $\mathcal{A}$ can make any sequence of queries to the first five oracles above. After that $\mathcal{A}$ can make a query to ***test*** on a fresh session. Then $\mathcal{A}$ outputs a guess $b^{'}$ for b. We define the advantage of $\mathcal{A}$:

$$\boldsymbol{Adv}_{\mathcal{A}}^{ID} = |Pr(b^{'} = b) - 1/2| \ .$$

Definition 3. *(Security): An AKE protocol ID is secure under the following conditions:*

(1) Two honest parties get the same session key with overwhelming probability.
(2) For any PPT adversary $\mathcal{A}$, $\boldsymbol{Adv}_{ID,\mathcal{A}}$ is negligible.

3 The KEA-style Authenticated Key Exchange

In this section, we describe our protocol in details. Let n be a power of 2. Define ring $R_q = Z_q[x]/(x^n + 1)$. Let $H : \{0,1\}^* \rightarrow \{0,1\}^\kappa$ be a hash function to derive session keys, where κ is the length of the session key. In our protocol, this hash function is simulated by a random oracle. Let χ_α be a discrete Gaussian distribution with parameter $\alpha \in R^+$. Let $a \in R_q$ be the public parameter uniformly chosen from R_q. Suppose Party i is the initiator, and Party j is the responder. Let $s_i \leftarrow_r \chi_\alpha$ be the static secret key of Party i, and $p_i = as_i + e_i$ is the public key of Party i, where $e_i \leftarrow_r \chi_\alpha$. Similarly, $s_j \leftarrow_r \chi_\alpha$ is the static secret key of Party j, and Party j's public key is $p_j = as_j + e_j$, where $e_j \leftarrow_r \chi_\alpha$.

Initiation: Initiator i proceeds as follows to activate the session:

a. Sample $r_i, f_i \leftarrow_r \chi_\alpha$, and compute $x_i = ar_i + f_i$;
b. Send x_i to Party j.

Response: After receiving x_i from Party i, Party j proceeds as follows:

1. Sample $r_j, f_j \leftarrow_r \chi_\alpha$, and compute $y_j = ar_j + f_j$;
2. Sample $g_{j1}, g_{j2} \leftarrow_r \chi_\alpha$, and compute $k_{j1} = p_i \cdot r_j + g_{j1}$ and $k_{j2} = x_i \cdot s_j + g_{j2}$;
3. Compute $(\sigma_{j1}, \omega_{j1}) \leftarrow Con(k_{j1}, params)$ and $(\sigma_{j2}, \omega_{j2}) \leftarrow Con(k_{j2}, params)$;
4. Party j derives his session key $sk_j = H(\sigma_{j1}, \sigma_{j2}, i, j, x_i, y_j, \omega_{j1}, \omega_{j2})$;
5. Send $y_j, \omega_{j1}, \omega_{j2}$ to Party i.

Finish: After receiving $y_j, \omega_{j1}, \omega_{j2}$ from Party j, Party i proceeds as follows:

c. Sample $g_{i1}, g_{i2} \leftarrow_r \chi_\alpha$, and compute $k_{i1} = p_j \cdot r_i + g_{i1}$, $k_{i2} = y_j \cdot s_i + g_{i2}$;
d. Compute $\sigma_{i1} = Rec(k_{i1}, \omega_{j1}, params)$ and $\sigma_{i2} = Rec(k_{i2}, \omega_{j2}, params)$, then Party i derives his session key $sk_i = H(\sigma_{i2}, \sigma_{i1}, i, j, x_i, y_j, \omega_{j1}, \omega_{j2})$;

Party i		Party j
PK: $p_i = as_i + e_i \in R_q$ SK: $s_i \leftarrow_r \chi_\alpha$		PK: $p_j = as_j + e_j \in R_q$ SK: $s_j \leftarrow_r \chi_\alpha$
$x_i = ar_i + f_i \in R_q$ where $r_i, f_i \leftarrow_r \chi_\alpha$	$\xrightarrow{x_i}$	$y_j = ar_j + f_j \in R_q$ where $r_j, f_j \leftarrow_r \chi_\alpha$ $k_{j1} = p_i \cdot r_j + g_{j1}$ $k_{j2} = x_i \cdot s_j + g_{j2}$ where $g_{j1}, g_{j2} \leftarrow_r \chi_\alpha$ $(\sigma_{j1}, \omega_{j1}) \leftarrow Con(k_{j1}, params)$
$k_{i1} = p_j \cdot r_i + g_{i1}$ $k_{i2} = y_j \cdot s_i + g_{i2}$ where $g_{i1}, g_{i2} \leftarrow_r \chi_\alpha$ $\sigma_{i1} = Rec(k_{i1}, \omega_{j1}, params)$ $\sigma_{i2} = Rec(k_{i2}, \omega_{j2}, params)$	$\xleftarrow{y_j, \omega_{j1}, \omega_{j2}}$	$(\sigma_{j2}, \omega_{j2}) \leftarrow Con(k_{j2}, params)$
$sk_i = H(\sigma_{i2}, \sigma_{i1}, i, j, x_i, y_j, \omega_{j1}, \omega_{j2})$		$sk_j = H(\sigma_{j1}, \sigma_{j2}, i, j, x_i, y_j, \omega_{j1}, \omega_{j2})$

Fig. 2. Our AKE protocol from ring-LWE

3.1 Analysis of the Protocol

Theorem 1. *(Correctness) For appropriately chosen parameters, both parties compute the same session key with overwhelming probability, which means* $sk_i = sk_j$.

Proof. To show $sk_i = sk_j$, it is sufficient to show that $\sigma_{i1} = \sigma_{j2}$ and $\sigma_{i2} = \sigma_{j1}$ according to the way the session keys are computed. We just need to show that k_{i1} is closed to k_{j2} and k_{i2} is closed to k_{j1}. Due to the symmetry, we only estimate the size of $\|k_{i2} - k_{j1}\|$.

$$\begin{aligned} k_{i2} - k_{j1} &= ((ar_j + f_j)s_i + g_{i2}) - ((as_i + e_i)r_j + g_{j1}) \\ &= (f_j s_i - e_i r_j) + (g_{i2} - g_{j1}) \,. \end{aligned} \tag{1}$$

According to Lemma 2, if $\|k_{i2} - k_{j1}\| < \frac{(g-1)q - gm}{2gm}$, we have $\sigma_{i2} = \sigma_{j1}$. Similarly, we have $\sigma_{i1} = \sigma_{j2}$. Here we just need to know if q is big enough, then the inequality can be satisfied. The concrete parameters will be considered in Section 5. ■

Theorem 2. *(Security) Let* n, q, α *be defined as above. Let* H *be a random oracle. If ring-LWE*$_{q,\alpha}$ *is hard, then the proposed AKE is secure.*

The proof of Theorem 2 appears in Appendix, and a proof sketch is given as follows.

Proof. (sketch) The proof proceeds by a sequence of games. In each game, a simulator $\mathcal{S}$ answers the queries of $\mathcal{A}$. We show that the output of $\mathcal{S}$ in the first

game is computationally indistinguishable with the output in the last game, and $\boldsymbol{Adv}_{\mathcal{A}}^{ID}$ of last game is negligible. Here are the basic ideas.

a. (P_i, a) (with secret s_i) and (P_j, a) (with secret s_j) are ring-LWE pairs. Given P_i and P_j, $\mathcal{A}$ cannot get any information about s_i and s_j.
b. (x_i, a) (with secret r_i) and (y_j, a) (with secret r_j) are ring-LWE pairs. Given x_i and y_j, $\mathcal{A}$ cannot get any information about r_i and r_j.
c. Due to the nice properties of ring-LWE, k_{j1}, k_{j2}, k_{i1}, k_{i2} are randomly distributed in R_q.
d. The distribution of ω_{j1} and ω_{j2} reveals no information about k_{j1} and k_{j2}.

These together indicate that the shared session key is secure. That is to say, the session key is uniformly random and independent of the messages exchanged during the session. ■

4 Efficient AKE with Pre-Computing

In this section, we consider the pre-computation to make our AKE protocol more efficient for the Internet. As we see, the most inefficient operation in our protocol is the multiplication over a ring. Inspired by KEA, we show that it is possible to pre-compute something off-line in our protocol, which can reduce the number of multiplication over a ring on-line. Define N is the maximum number of parties, and $[N] := \{0, ..., N-1\}$. Let $s_i \leftarrow_r \chi_\alpha$ be the static secret key of Party i, and $p_i = as_i + e_i$ is the public key of Party i, where $e_i \leftarrow_r \chi_\alpha$. Similarly, $s_j \leftarrow_r \chi_\alpha$ is the static secret key of Party j, and Party j's public key is $p_j = as_j + e_j$, where $e_j \leftarrow_r \chi_\alpha$. We give the description of our pre-computing version AKE protocol:

Off-line: Take Party i as an example. Party i chooses $r_i, f_i \leftarrow_r \chi_\alpha$, and computes $x_i = ar_i + f_i$ and $k_{i1}^j = p_j \cdot r_i + g_{i1}$ for every $j \in [N]/i$. Party i holds the Table T_i which stores the N values (x_i and k_{i1}^j for $j \in [N]/i$) computed above. Similarly, for $j \in N/i$, Party j executes the same as Party i, and holds its Table T_j.

On-line: Suppose Party i is a initiator and Party j is a responder.
Initiation: Party i proceeds as follows to activate the session:

a. Look up the Table T_i for x_i.
b. Send x_i to Party j.

Response: After receiving x_i from Party i, Party j proceeds as follows:

1. Look up the table S_j for y_j;
2. Sample $g_{j1}, g_{j2} \leftarrow_r \chi_\alpha$, and compute $k_{j2} = x_i \cdot s_j + g_{j2}$. Look up the Table T_j for $k_{j1} = k_{j1}^i$;
3. Compute $(\sigma_{j1}, \omega_{j1}) \leftarrow Con(k_{j1}, params)$ and $(\sigma_{j2}, \omega_{j2}) \leftarrow Con(k_{j2}, params)$;
4. Party j derives his session key $sk_j = H(\sigma_{j1}, \sigma_{j2}, i, j, x_i, y_j, \omega_{j1}, \omega_{j2})$;
5. Send $y_j, \omega_{j1}, \omega_{j2}$ to Party i.

Finish: After receiving $y_j, \omega_{j1}, \omega_{j2}$ from Party j, Party i proceeds as follows:

c. Sample $g_{i1}, g_{i2} \leftarrow_r \chi_\alpha$, and compute $k_{i2} = y_j \cdot s_i + g_{i2}$. Look up the Table T_i for $k_{i1} = k_{i1}^j$;
d. Compute $\sigma_{i1} = Rec(k_{i1}, \omega_{j1}, params)$ and $\sigma_{i2} = Rec(k_{i2}, \omega_{j2}, params)$, then Party i derives his session key $sk_i = H(\sigma_{i2}, \sigma_{i1}, i, j, x_i, y_j, \omega_{j1}, \omega_{j2})$;

In practice, the number N is not very big. Therefore, the size of Table T_i is N ring elements, which is small. Table lookups have advantages in efficiency over multiplication over rings. There is only 1 multiplication in our protocol compared with 3 multiplications in the protocol from Section 3 and 4 multiplications in the protocol from [35].

5 Parameters and Conclusions

To maintain the property of correctness, according to the conclusion of Theorem 1, $\|k_{i2} - k_{j1}\| < \frac{(g-1)q-gm}{2gm}$ must be satisfied, that is to say:

$$(f_j s_i - e_i r_j) + (g_{i1} - g_{j1}) < \frac{(g-1)q - gm}{2gm} \ . \tag{2}$$

Combine the Fact 1 and Fact 2, with high probability, we have:

$$\begin{aligned}(f_j s_i - e_i r_j) + (g_{i1} - g_{j1}) &\leq (\|f_j s_i\| + \|e_i r_j\| + \|g_{i1}\| + \|g_{j1}\|) \\ &\leq (6\alpha^2\sqrt{n} + 6\alpha^2\sqrt{n} + 6\alpha + 6\alpha) \\ &= 12(\alpha^2\sqrt{n} + \alpha) \ .\end{aligned}$$

So we have the inequality:

$$q > \frac{24gm(\alpha^2\sqrt{n} + \alpha)}{g-1} \ . \tag{3}$$

As recommended in [26, 31], it is necessary to set the Gaussian parameter α as:

$$\alpha \geq 8/\sqrt{2\pi} \ . \tag{4}$$

To estimate the concrete security of our protocol, we consider the approach of [12], which investigates the two most efficient ways to solve the underlying (ring-)LWE problem, namely the embedding attack and the decoding attack. The embedding attack is more efficient than the decoding attack when the adversary only has access to a few samples. In our protocol, the decoding attack is more efficient as m is close to the optimal dimension $\sqrt{nlg(q)/lg(\delta))}$. Thus we concentrate on the decoding attack.
The decoding attack was introduced by Lindner et al. [26], which is inherently from nearest-plane algorithm. It is further improved by Liu et al. [27] with pruned enumeration approach. For a instance of (ring-)LWE, the decoding attack first uses a lattice reduction algorithm, and then applies a decoding algorithm

from [26] or [27]. The output is a set of vectors closed to the target vector. There is a continuous correspondence between the success probability of returning the actual closest vector and the Hermite factor. In our analysis, we follow the approach proposed by Lindner et al. [26] to predict this success probability, and the runtime of lattice reduction algorithm is predicted by $T(\delta) = 1.8/lg(\delta) - 110$.

Above all, the candidates of our parameters are given in Table 3. The size of sk is the value of expectation computed using Fact 1 ($sk \in (-10\alpha, 10\alpha)$ with high probability).

Table 3. Candidates of our parameters.

n	m	g	α	$log\, q$	Security	pk	sk	init. msg	resp. msg	$\|K\|$
1024	2^4	2^1	3.192	32	80	4 KB	0.75 KB	4 KB	4.25KB	2KB
1024	2^2	2^1	3.192	24	132	3 KB	0.75 KB	3 KB	3.25 KB	0.5KB
1024	2^1	2^1	3.192	18	190	2.25 KB	0.75 KB	2.25 KB	2.5 KB	0.125KB

References

1. Agrawal, S., Boneh, D., Boyen X.: Efficient Lattice (H)IBE in the Standard Model. In: Gilbert H. EUROCRYPT 2010. LNCS, vol. 6110, pp. 553-572. Springer, Berlin, Heidelberg (2010)
2. Aiello, W., Bellovin, S., M., Blaze, M., Canetti, R., Ioannidis, J., Keromytis, A. D., Reingold, O.: Just fast keying: Key agreement in a hostile internet. ACM Trans. Inf. Syst. Secur. 7,2, pp. 242-273 (2004)
3. Alkim, E., Ducas, L., Poppelmann, T., Schwabe, P.: Post-quantum key exchange-a new hope. In: USENIX Security Symposium. pp. 327-343 (2016)
4. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: Stinson D.R. (eds) CRYPTO 1993. LNCS, vol 773, pp. 232-249. Springer, Berlin, Heidelberg (1993)
5. Berlekamp, E., R., McEliece, R., J., Van Tilborg, H., C.: On the inherent intractability of certain coding problems. IEEE Transactions on Information Theory, 24.3, pp. 384-386 (1978)
6. Blake-Wilson, S., Johnson, D., Menezes, A.: Key Agreement Protocols and Their Security Analysis. In: Darnell M. (eds). Cryptography and Coding 1997. LNCS, vol 1355, pp. 30-45. Springer, Berlin, Heidelberg (1997)
7. Bos, J., Costello, C., Naehrig, M., Stebila, D.: Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In: IEEE Symposium on Security and Privacy, pp. 553-570 (2015)
8. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. SIAM Journal on Computing, 43.2, pp. 831-871 (2014)
9. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Rogaway P. (eds). CRYPTO 2011. LNCS, vol 6841, pp. 505-524. Springer, Berlin, Heidelberg (2011)

10. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann B. (eds). EUROCRYPT 2001. LNCS, vol 2045, pp. 453-474. Springer, Berlin, Heidelberg (2001)
11. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. Journal of cryptology, 25.4. pp. 601-639 (2012)
12. Dagdelen, O., Bansarkhani, R.E., Gopfert, F., Guneysu, T., Oder, T., Poppelmann, T., Sanchez, A.H., Schwabe, P.: High-speed signatures from standard lattices. In: Aranha D., Menezes A. (eds). LATINCRYPT 2014. LNCS, vol 8895, pp. 84-103. Springer, Cham (2014)
13. Dierks, T., Allen, C.: The TLS protocol version 1.0 (1999)
14. Ding, J., Xie, X., Lin, X.: A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem. `http://eprint.iacr.org/2012/688`
15. Ding, J., Alsayigh, S., Saraswathy, R. V., Fluhrer, S., Lin, X.: Leakage of signal function with reused keys in RLWE key exchange. In Communications (ICC), 2017 IEEE International Conference. IEEE. (2017)
16. Freier, A., Karlton, P., Kocher, P.: The secure sockets layer (SSL) protocol version 3.0 (2011)
17. Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Strongly Secure Authenticated Key Exchange from Factoring, Codes, and Lattices. In: Fischlin M., Buchmann J., Manulis M. (eds) Public Key Cryptography C PKC 2012. Lecture Notes in Computer Science, vol 7293. Springer, Berlin, Heidelberg (2012)
18. Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Practical and post-quantum authenticated key exchange from one-way secure key encapsulation mechanism. In: ASIACCS 2013, pages 83C94 (2013)
19. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In Proceedings of the fortieth annual ACM symposium on Theory of computing (STOC '08). ACM, New York, NY, USA. pp. 197-206 (2008)
20. Jager, T., Kohlar, F., Schage, S., Schwenk, J.: On the security of TLS-DHE in the standard model. In: Safavi-Naini R., Canetti R. (eds). CRYPTO 2012. LNCS, vol 7417, pp. 273-293. Springer, Berlin, Heidelberg (2012)
21. Jin, Z., Zhao, Y.: Optimal Key Consensus in Presence of Noise. `http://eprint.iacr.org/2017/1058`
22. Kaufman, C.: Internet key exchange (IKEv2) protocol (2005)
23. Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In: Shoup V. (eds). CRYPTO 2005. LNCS, vol 3621, pp. 546-566. Springer, Berlin, Heidelberg (2005)
24. Krawczyk, H.: SIGMA: The 'SIGn-and-MAc' approach to authenticated Diffie-Hellman and its use in the IKE protocols. In: Boneh D. (eds). CRYPTO 2003. LNCS, vol 2729, pp. 400-425. Springer, Berlin, Heidelberg (2003)
25. Lauter, K., E., Mityagin, A.: Security analysis of KEA authenticated key exchange protocol. In: Yung M., Dodis Y., Kiayias A., Malkin T. (eds). PKC 2006. LNCS, vol 3958, pp. 378-394. Springer, Berlin, Heidelberg (2006)
26. Lindner, R., Peikert, C.: Better key sizes (and attacks) for LWE-based encryption. In: Kiayias A. (eds). CT-RSA 2011. LNCS, vol 6558, pp. 319-339. Springer, Berlin, Heidelberg (2011)
27. Liu M., Nguyen P.Q.: Solving BDD by Enumeration: An Update. In: Dawson E. (eds) Topics in Cryptology C CT-RSA 2013. CT-RSA 2013. Lecture Notes in Computer Science, vol 7779. Springer, Berlin, Heidelberg (2013)
28. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. Journal of the ACM, 60.6 (2013)

29. Menezes, A., Qu, M., Vanstone, S.: Some new key agreement protocols providing mutual implicit authentication. In: SecondWorkshop on Selected Areas in Cryptography (1995)
30. Peikert, C.: Lattice cryptography for the internet. In: Mosca M. (eds). PQCrypto 2014. LNCS, vol 8772, 197-219. Springer, Cham (2014)
31. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM, 56.6, pp. 34 (2009)
32. Saint, E., L., Fedronic, D., Liu, S.: Open Protocol for Access Control Identification and Ticketing with Privacy. OPACITY protocol specification 2011
33. Skipjack, N.: KEA algorithm specifications (1998)
34. Yao, A., C., C., Zhao, Y.: OAKE: A new family of implicitly authenticated Diffie-Hellman protocols. In Proceedings of the 2013 ACM SIGSAC conference on Computer and communications security (CCS '13). ACM, New York, NY, USA, pp. 1113-1128 (2013)
35. Zhang, J., Zhang, Z., Ding, J., Snook, M., Dagdelen, Ö.: Authenticated Key Exchange from Ideal Lattices. In: Oswald E., Fischlin M. (eds). EUROCRYPT 2015. LNCS, vol 9057, pp. 719-751. Springer, Berlin, Heidelberg (2015)

Appendix: Proof of Theorem 2

The strategy of our proof is from [35]. First of all, we classify all the adversaries into five types, which give a complete description of all situations. Let $M = ploy(n)$ be the maximum number of sessions for each party. We distinguish the five types of adversaries as:

- **TYPE I:** $sid^* = (ID, I, i^*, j^*, x_{i^*}, (y_{j^*}, \omega_{j^*1}, \omega_{j^*2}))$ is the test session, and y_{j^*} is output by a session activated at Party j^* by a $Send_1(ID, R, j^*, i^*, x_{i^*})$ query.
- **TYPE II:** $sid^* = (ID, I, i^*, j^*, x_{i^*}, (y_{j^*}, \omega_{j^*1}, \omega_{j^*2}))$ is the test session, and y_{j^*} is not output by a session activated at Party j^* by a $Send_1(ID, R, j^*, i^*, x_{i^*})$ query.
- **TYPE III:** $sid^* = (ID, I, j^*, i^*, x_{i^*}, (y_{j^*}, \omega_{j^*1}, \omega_{j^*2}))$ is the test session, and x_{i^*} is not output by a session activated at Party i^* by a $Send_0(ID, I, i^*, j^*,)$ query.
- **TYPE IV:** $sid^* = (ID, I, j^*, i^*, x_{i^*}, (y_{j^*}, \omega_{j^*1}, \omega_{j^*2}))$ is the test session, and x_{i^*} is output by a session activated at Party i^* by a $Send_0(ID, I, i^*, j^*,)$ query, but i^* either never completes the session, or i^* completes it with exact y_{j^*}.
- **TYPE V:** $sid^* = (ID, I, j^*, i^*, x_{i^*}, (y_{j^*}, \omega_{j^*1}, \omega_{j^*2}))$ is the test session, and x_{i^*} is output by a session activated at Party i^* by a $Send_0(ID, I, i^*, j^*,)$ query, but i^* completes the session with another $y_j^{'} \neq y_{j^*}$.

To prove Theorem 2, it is sufficient to show that the protocol is security under each type of adversary. The proof of different types of adversaries are similar, so we only give the detailed security proof for $TypeI$ adversary in next lemma.

Lemma 3. *Let n be a power of 2, q is a prime satisfying $q = 1 \, mod \, 2n$, real $\beta = \omega(\alpha\gamma n\sqrt{nlog\, n})$. Then, if ring-$LWE_{q,\alpha}$ is hard, the proposed AKE is secure against any PPT TypeI adversary $\mathcal{A}$ in the random oracle model.*

Proof of Lemma 3: A simulator $\mathcal{S}$ will be constructed to answer the queries from $\mathcal{A}$. After a sequence of games, *Game m* for $0 \leq m \leq 4$, we show that the advantage of $\mathcal{A}$ is negligible. In *Game* 0, $\mathcal{S}$ executes as the original protocol, while the last game *Game* 4 is a fake one with randomly and independently chosen session key for the test session. The security is established by showing that any two consecutive games are computationally indistinguishable. As any two consecutive games are very similar, we only show the difference between two games by framing them.

***Game* 0**: $\mathcal{S}$ uses the original protocol to simulates the security game for $\mathcal{A}$. $\mathcal{S}$ chooses $i^*, j^* \leftarrow_r \{1, ..., N\}$, $m_{i^*}, m_{j^*} \leftarrow_r \{1, ..., M\}$, and supposes the sid for the test session is $sid^* = (ID, I, i^*, j^*, x_{i^*}, (y_{j^*}, \omega_{j^*1}, \omega_{j^*2}))$, where x_{i^*} is output by the m_{i^*}-th session of Party i^*, and y_{j^*} is output by the m_{j^*}-th session of Party j^* activated by a $Send_1(ID, R, j^*, i^*, x_{i^*})$ query. Then, $\mathcal{S}$ chooses $a \leftarrow_r R_q$, generates static public keys for all parties. Besides, $\mathcal{S}$ maintains a table L for the hash function H to serve as a random oracle. After that, $\mathcal{S}$ answers $\mathcal{A}$'s queries as follows:

- $\boldsymbol{H}(in)$: If there does not exist a tuple (in, out) in L, choose an invertible element $out \in \chi_\gamma$ at random, and add (in, out) into L. Then, return out to $\mathcal{A}$.
- $\boldsymbol{Send_0(ID, I, i, j)}$ **:** $\mathcal{A}$ intends to activate a new session with Party j, and $\mathcal{S}$ proceeds as follows:
  a. Sample $r_i, f_i \leftarrow_r \chi_\alpha$, and compute $x_i = ar_i + f_i$;
  b. Send x_i to Party j.
- $\boldsymbol{Send}_1(ID, R, j, i, x_i)$: $\mathcal{S}$ computes sk_j as follows:
  1. Sample $r_j, f_j \leftarrow_r \chi_\alpha$, and compute $y_j = ar_j + f_j$;
  2. Sample $g_{j1}, g_{j2} \leftarrow_r \chi_\alpha$, and compute $k_{j1} = p_i \cdot r_j + g_{j1}$ and $k_{j2} = x_i \cdot s_j + g_{j2}$;
  3. Compute $(\sigma_{j1}, \omega_{j1}) \leftarrow Con(k_{j1}, params)$ and $(\sigma_{j2}, \omega_{j2}) \leftarrow Con(k_{j2}, params)$;
  4. Party j derives his session key $sk_j = H(\sigma_{j1}, \sigma_{j2}, i, j, x_i, y_j, \omega_{j1}, \omega_{j2})$;
  5. Send $y_j, \omega_{j1}, \omega_{j2}$ to Party i.
- $\boldsymbol{Send}_2(ID, I, i, j, x_i, (y_j, \omega_{j1}, \omega_{j2}))$: $\mathcal{S}$ proceeds as follows:
  c. Sample $g_{i1}, g_{i2} \leftarrow_r \chi_\alpha$, and compute $k_{i1} = p_j \cdot r_i + g_{i1}$, $k_{i2} = y_j \cdot s_i + g_{i2}$;
  d. Compute $\sigma_{i1} = Rec(k_{i1}, \omega_{j1}, params)$ and $\sigma_{i2} = Rec(k_{i2}, \omega_{j2}, params)$, then Party i derives his session key $sk_i = H(\sigma_{i2}, \sigma_{i1}, i, j, x_i, y_j, \omega_{j1}, \omega_{j2})$;
- $\boldsymbol{SessionKeyReveal}(sid)$: After receiving sid from $\mathcal{A}$, $\mathcal{S}$ returns sk_i if the session key of sid has been generated.
- $\boldsymbol{Corrupt}(i)$: Return the static secret key s_i of Party i to $\mathcal{A}$.
- $\boldsymbol{Test}(sid)$: The input of this query is $sid = (ID, I, i, j, x_i, (y_j, \omega_{j1}, \omega_{j2})$. After receiving sid from $\mathcal{A}$, firstly, $\mathcal{S}$ should make sure that $(i, j) = (i^*, j^*)$; x_i and y_j are output by the m_{i^*}-th session of Party i^* and the m^*-th session of Party j^* respectively. If not, $\mathcal{S}$ aborts. Else $\mathcal{S}$ chooses a random $b \in \{0, 1\}$. $\mathcal{S}$ returns sk_i of sid if $b = 1$, else returns $sk_i^{'} \leftarrow_r \{0, 1\}^\kappa$.

We define $EQUAL_i$ as the event that $\mathcal{A}$ outputs b^* and $b^* = b$ in *Game* i. Then we consider the probability that $\mathcal{S}$ will not abort in *Game* 0.

Claim 1. $Pr(\mathcal{S}\ will\ not\ abort) = 1/M^2N^2$.

Proof. i^* and j^* are randomly chosen from $\{1, ..., N\}$, and m_{i^*} and m_{j^*} are random chosen from $\{1, ..., M\}$ by $\mathcal{S}$, which are independently from the view of $\mathcal{A}$. So the probability that $(i, j) = (i^*, j^*)$, and x_i and y_j are output by the m_{i^*}-th session of Party i^* and the m^*-th session of Party j^* respectively is $1/M^2N^2$. □

***Game* 1**: In *Game* 1, $\mathcal{S}$ behaves almost the same as in *Game* 0, except the computation of sk_i

- $\boldsymbol{Send}_2(ID, I, i, j, x_i, (y_j, \omega_{j1}, \omega_{j2}))$: If $(i, j) \neq (i^*, j^*)$, or x_i is not the s_i^*-th session of i^*, $\mathcal{S}$ behaves the same as in *Game* 2. Else if $(y_j, \omega_{j1}, \omega_{j2})$ is output by the s_j^*-th session of Party j^*, $\boxed{\mathcal{S}\text{ directly sets } sk_i = sk_j \text{ and return it to } \mathcal{A}}$, where sk_j is the session key of $sid = (ID, R, j, i, x_i, (y_j, \omega_{j1}, \omega_{j2}))$. Otherwise, $\mathcal{S}$ samples $g_{i1}, g_{i2} \leftarrow_r \chi_\alpha$, and computes $k_{i1} = p_j \cdot r_i + g_{i1}$, $k_{i2} = y_j \cdot s_i + g_{i2}$. Finally, it computes $\sigma_{i1} = Rec(k_{i1}, \omega_{j1}, params)$ and $\sigma_{i2} = Rec(k_{i2}, \omega_{j2}, params)$, and derives $sk_i = H(\sigma_{i2}, \sigma_{i1}, i, j, x_i, y_j, \omega_{j1}, \omega_{j2})$.

Claim 2. $Pr(EQUAL_1) = Pr(EQUAL_0) - negl(\kappa)$

Proof. In the second case, it is just a conceptual change of *Game* 0 due to the correctness of the protocol. □

We know that y_j is output by the s_j^*-th session of Party j^*. But we never know whether $(\omega_{j1}, \omega_{j2}) = (\omega'_{j1}, \omega'_{j2})$. But for convenience, we suppose $(\omega_{j1}, \omega_{j2}) = (\omega'_{j1}, \omega'_{j2})$, and we consider the situation $(\omega_{j1}, \omega_{j2}) \neq (\omega'_{j1}, \omega'_{j2})$ later. We define Q_l as the event in Game ***Game* $\boldsymbol{l}$** that: 1) $(y_j, \omega'_j, \omega'_{j2})$ is output by the s_j^*-th session of party j^* but $(\omega_j, \omega_{j2}) \neq (\omega'_j, \omega'_{j2})$, and 2) $\mathcal{A}$ makes a query to H that is exactly used to generate the session key sk_i for the s_i^*-th session of party i^*.

***Game* 2**: $\mathcal{S}$ behaves almost the same as in ***Game* 1** except in the follow case:

- $\boldsymbol{Send_0(ID, I, i, j)}$ **:** If $(i, j) \neq (i^*, j^*)$, or it is not the s_i^*-th session of i^*, $\mathcal{S}$ behaves the same as in *Game* 1. Otherwise, $\mathcal{S}$ proceeds as follows:
  a. Sample $r_i, f_i \leftarrow_r \chi_\alpha$;
  b. Randomly chooses $\boxed{x_i \leftarrow_r R_q}$;
  c. Send x_i to $\mathcal{A}$.
- $\boldsymbol{Send}_1(ID, R, j, i, x_i)$: If $(i, j) \neq (i^*, j^*)$, or it is not the s_j^*-th session of j^*, $\mathcal{S}$ behaves the same as in *Game* 2. Otherwise, $\mathcal{S}$ computes sk_j as follows:
  1. Randomly chooses $\boxed{y_j \leftarrow_r R_q}$;
  2. Sample $g_{j1}, g_{j2} \leftarrow_r \chi_\alpha$, and compute $k_{j1} = p_i \cdot r_j + g_{j1}$ and $k_{j2} = x_i \cdot s_j + g_{j2}$;

3. Compute $(\sigma_{j1}, \omega_{j1}) \leftarrow Con(k_{j1}, params)$ and $(\sigma_{j2}, \omega_{j2}) \leftarrow Con(k_{j2}, params)$;
4. Party j derives his session key $sk_j = H(\sigma_{j1}, \sigma_{j2}, i, j, x_i, y_j, \omega_{j1}, \omega_{j2})$;
5. Send $y_j, \omega_{j1}, \omega_{j2}$ to Party i.

– $\boldsymbol{Send}_2(ID, I, i, j, x_i, (y_j, \omega_{j1}, \omega_{j2}))$: If $(i, j) \neq (i^*, j^*)$, or x_i is not the s_i^*-th session of i^*, or $(y_j, \omega_{j1}, \omega_{j2})$ is output by the s_j^*-th session of Party j^*, $\mathcal{S}$ behaves the same as in *Game* 1. Otherwise, $\mathcal{S}$ proceeds as folllows

c. $\boxed{\text{Randomly chooses } k_{i1}, k_{i2} \leftarrow_r R_q}$;
d. Compute $\sigma_{i1} = Rec(k_{i1}, \omega_{j1}, params)$ and $\sigma_{i2} = Rec(k_{i2}, \omega_{j2}, params)$, then Party i derives his session key $sk_i = H(\sigma_{i2}, \sigma_{i1}, i, j, x_i, y_j, \omega_{j1}, \omega_{j2})$;

Claim 3. : If ring-$LWE_{q,\alpha}$ is hard, we have $Pr(Q_2) = Pr(Q_1) - negl(\kappa)$, and $Pr(EQUAL_2|\neg Q_2) = Pr(EQUAL_1|\neg Q_1) - negl(\kappa)$.

Proof. Since H is a random oracle, the event Q_i $(i = 1, 2)$ is independent from the distribution of the corresponding sk_i. In addition, under the ring-$LWE_{q,\alpha}$ assumption, we have $x_i = ar_i + f_i$ and $y_j = ar_j + f_j$ in *Game* 1 are computationally indistinguishable from uniform distribution over $\mathbb{R}_q$, so $Pr(Q_2) = Pr(Q_1)$. Suppose Q_i $(i = 1, 2)$ does not happen, $\mathcal{A}$ gains no information about sk_i which means $Pr(EQUAL_2|\neg Q_2) = Pr(EQUAL_1|\neg Q_1) - negl(\kappa)$. □

***Game* 3**: $\mathcal{S}$ behaves almost the same as in ***Game* 2** except in the follow case:

– $\boldsymbol{Send}_1(ID, R, j, i, x_i)$: If $(i, j) \neq (i^*, j^*)$, or it is not the s_j^*-th session of j^*, $\mathcal{S}$ behaves the same as in *Game* 2. Otherwise, $\mathcal{S}$ computes sk_j as follows:

1. Randomly chooses $y_j \leftarrow_r R_q$;
2. $\boxed{\text{Randomly chooses } k_{j1}, k_{j2} \leftarrow_r R_q}$;
3. Compute $(\sigma_{j1}, \omega_{j1}) \leftarrow Con(k_{j1}, params)$ and $(\sigma_{j2}, \omega_{j2}) \leftarrow Con(k_{j2}, params)$;
4. Party j derives his session key $sk_j = H(\sigma_{j1}, \sigma_{j2}, i, j, x_i, y_j, \omega_{j1}, \omega_{j2})$;
5. Send $y_j, \omega_{j1}, \omega_{j2}$ to Party i.

Claim 4. : If ring-$LWE_{q,\alpha}$ is hard, we have ***Game* 2** and ***Game* 3** are computationally indistinguishable. In particular, we have $Pr(Q_3) = Pr(Q_2) - negl(\kappa)$ and $Pr(EQUAL_3|\neg Q_3) = Pr(EQUAL_2|\neg Q_2) - negl(\kappa)$

Proof. It is easy to identify that k_{j1} and k_{j2} are ring-$LWE_{q,\alpha}$ samples. Under the ring-$LWE_{q,\alpha}$ assumption, we have k_{j1} and k_{j2} in *Game* 2 are computationally indistinguishable from uniform distribution over $\mathbb{R}_q$. □

Now it is enough to show that the event Q_3 is happen with negligible probability:

Claim 5. We have $Pr[Q_3] = 2^{-n} + negl(\kappa)$.

Proof. Let $k_{i1,l}$ and $k_{i2,l}$ be the element computed by $\mathcal{S}$ for the s_i^*-th session at party i^* in *Game* l, and $k_{j1,l}$ and $k_{j2,l}$ be the element computed by S for the s_j^*-th session at party j^*. By the correctness of the protocol, we have that $k_{i1,1} = k_{j1,1} + \hat{g_1}$ and $k_{i2,1} = k_{j2,1} + \hat{g_2}$ for some $\hat{g_1}, \hat{g_2}$ with small coefficients. As we have proven that the view of the adversary before Q_l happens is computationally indistinguishable, the equation $k_{i1,3} = k_{j1,3} + \hat{g_1}'$ and $k_{i2,3} = k_{j2,3} + \hat{g_2}'$ should

still holds for some $\hat{g_1}', \hat{g_2}'$ with small coefficients. In *Game* 3, $k_{j1,3}$ and $k_{j2,3}$ are randomly chosen from $\mathbb{R}_q$, and under the property of OKCN in Lemma 2, we have that the adversary makes a query to H that is exactly used to generate the session key sk_i for the s_i^*-th session of party i^* is $2^{-n} + negl(\kappa)$. □

Claim 6. $Pr[EQUAL_3|\neg Q_3] = 1/2 + negl()$

Proof. Let $(y_j, \omega_{j1}, \omega_{j2})$ be output by the s_j^*-th session of Party j, and $j = j^*$. $(y_j, \omega_{j1}^{'}, \omega_{j2}^{'})$ be the message that is used to complete the test session. We conclude the proof by the following two cases:

- $\omega_j = \omega_j^{'}$: Since k_j1 and k_j2 is randomly chosen from R_q, so $\sigma_{j1}, \sigma_{j2} \in \{0,1\}^n$ is random duo to the property of reconciliation mechanism. Thus, the probability that $\mathcal{A}$ has made a query H with the exact $(\sigma_{j1}, \sigma_{j2})$ is $2^{-n} + negl(\kappa)$.
- $\omega_j \neq \omega_j^{'}$: According to Claim 5, $\mathcal{A}$ will never make a query H with the exact $(\sigma_{j1}, \sigma_{j2})$. □

Combining the Claim 1 to Claim 6, we have that Lemma 3 follows.

Similarly, it's simple to show that our protocol is security against adversary of **TYPE II**, **TYPE III**, **TYPE IV** and **TYPE V**. As we know, These five types of adversaries give a complete partition of all the adversaries, it's sufficient to complete the proof of Theorem 2. ■