Authenticated Key Exchange from Ideal Lattices

Jiang Zhang[1], Zhenfeng Zhang[1,*], Jintai Ding[2,*], Michael Snook[2], and Özgür Dagdelen[3]

[1] Institute of Software, Chinese Academy of Sciences, China
[2] University of Cincinnati, Cincinnati, USA
[3] Darmstadt University of Technology, Germany
jiangzhang09@gmail.com, zfzhang@tca.iscas.ac.cn, jintai.ding@gmail.com,
snookml@mail.uc.edu, oezguer.dagdelen@cased.de
* Correspoding Authors

Abstract. In this paper, we present a practical and provably secure two-pass AKE protocol from ideal lattices, which is conceptually simple and has similarities to the Diffie-Hellman based protocols such as HMQV (CRYPTO 2005) and OAKE (CCS 2013). Our protocol does not rely on other cryptographic primitives—in particular, it does not use signatures—simplifying the protocol and resting the security solely on the hardness of the ring learning with errors problem. The security is proven in the Bellare-Rogaway model with weak perfect forward secrecy. We also give a one-pass variant of our two-pass protocol, which might be appealing in specific applications. Several concrete choices of parameters are provided, and a proof-of-concept implementation shows that our protocols are indeed practical.

1 Introduction

Key Exchange (KE) is a fundamental cryptographic primitive, allowing two parties to securely generate a common secret key over an insecure network. Because symmetric cryptographic tools (*e.g.* AES) are reliant on both parties having a shared key in order to securely transmit data, KE is one of the most used cryptographic tools in building secure communication protocols (*e.g.* SSL/TLS, IPSec, SSH). Following the introduction of the Diffie-Hellman (DH) protocol [26], cryptographers have devised a wide selection of KE protocols with various use-cases. One such class is Authenticated Key Exchange (AKE), a class of KE protocols where each party is able to verify the other's identity, so that an adversary cannot impersonate one party in the conversation.

For an AKE protocol, each party has a pair of *static keys*: a *static secret key* and a corresponding *static public key*. The static public key is certified to belong to its owner using a public-key or ID-based infrastructure. For each run of the protocol, the parties involved generate *ephemeral secret keys* and use these to generate *ephemeral public keys* that they exchange. Then all the keys are used along with the transcripts of the session to create a shared *session state*, which is then passed to a *key derivation function* to obtain the final session key. Intuitively, such a protocol is secure if no efficient adversary is able to extract any information about the session key from the publicly exchanged messages. More formally, Bellare and Rogaway [8] introduced an indistinguishability-based security model for AKE, the BR model, which captures key authentication such as *implicit mutual key authentication* and *confidentiality of agreed session keys*. The most prominent alternatives stem from Canetti and Krawczyk [15] and LaMacchia *et al.*[48], that also accounts for scenarios in which the adversary is able to obtain information about a static secret key or a session state other than the state of the target session. In practice, AKE protocols are usually required to have a property, Perfect Forward Secrecy (PFS), that an adversary cannot compromise session keys after a completed session, even if it obtains the parties' static secret keys (*e.g.*, via a heartbleed attack[4]). As shown in [46], no two-pass AKE protocol based on public-key authentication can achieve PFS. Thus, the notion of weak PFS

[4] http://heartbleed.com/

(wPFS) is usually considered for two-pass AKE protocols, which states that the session key of an honestly run session remains private if the static keys are compromised after the session is finished [46].

One approach for achieving authentication in KE protocols is to explicitly authenticate the exchanged messages between the involved parties by using some cryptographic primitives (*e.g.*, signatures, or MAC), which usually incurs additional computation and communication overheads with respect to the basic KE protocol, and complicates the understanding of the KE protocol. This includes several well-known protocols such as IKE [39,44], SIGMA [45], SSL [31], TLS [25,47,56,36,12], as well as the standard in German electronic identity cards, namely EAC [14,22], and the standardized protocols OPACITY [23] and PLAID [24]. Another line of designing AKEs follows the idea of MQV [57,41,46,68] (which has been standardized by ISO/IEC and IEEE, and recommended by NIST and NSA Suite B) by making good use of the algebraic structure of DH problems to achieve implicit authentication. All the above AKEs are based on classic hard problems, such as factoring, the RSA problem, or the computational/decision DH problem. Since these hard problems are vulnerable to quantum computers [64] and as we are moving into the era of quantum computing, it is very appealing to find other counterparts based on problems believed to be resistant to quantum attacks. For instance, post-quantum AKE is considered of high priority by NIST [17]. Due to the potential benefits of lattice-based constructions such as asymptotic efficiency, conceptual simplicity, and worst-case hardness assumptions, it makes perfect sense to build lattice-based AKEs.

1.1 Our Contribution

In this paper, we propose an efficient AKE protocol based on the Ring Learning With Errors (Ring-LWE), which in turn is as hard as some lattice problems (*e.g.*, SIVP) in the worst case on ideal lattices [54,29]. Our method avoids introducing extra cryptographic primitives, thus simplifying the design and reducing overhead. In particular, the communicating parties are not required to either encrypt any messages with the other's public key, nor sign any of their own messages during key exchange. Furthermore, by having the key exchange as a self-contained system, we reduce the security assumptions needed, and are able to rely directly and solely on the hardness of Ring-LWE.

By utilizing many useful properties of Ring-LWE problems and discrete Gaussian distributions, we establish an approach to combine both the static and ephemeral public/secret keys, in a manner similar to HMQV [46]. Thus, our protocol not only enjoys many nice properties of HMQV such as two-pass messages, implicit key authentication, high efficiency, and without using any explicit entity authentication techniques (*e.g.*, signatures), but also has many properties of lattice-based cryptography, such as asymptotic efficiency, conceptual simplicity, worst-case hardness assumption, as well as resistance to quantum computer attacks. However, there are also several shortcomings inherited from lattice-based cryptography, such as "handling of noises" and large public/secret keys. Besides, unlike HMQV which works on "nice-behaving" cyclic groups, the security of our protocol cannot be proven in the CK model [15] due to the underlying noise-based algebraic structures. Fortunately, we prove the security in the BR model, which is the most common model considered as it is usually strong enough for many practical applications and it comes with composability [13]. In addition, our protocol achieves weak PFS property, which is known as the best PFS notion achievable by two-pass protocols [46].

As MQV [57] and HMQV [46], we present a one-pass variant of our basic protocol (*i.e.*, the two parties can only exchange a single message in order to derive a shared session key), which might be useful in client-server based applications. Finally, we select concrete choices of parameters and construct a proof-of-concept implementation to examine the efficiency of our protocols. Through the implementation has not undergone any real optimization, the performance results already indicate that our protocols are practical.

We note that none of the techniques we use prevent us from instantiating our AKE protocol based on standard lattices. One just has to keep in mind that key sizes and performance eventually become worse.

1.2 Techniques, and Relation to HMQV

Our AKE protocol is inspired by HMQV [46], which makes our protocol share some similarities to HMQV. However, there are also many differences between our protocol and HMQV due to the different underlying algebraic structures. To better illustrate the commons and differences between our AKE protocol and HMQV, we first briefly recall the HMQV protocol [46]. Let $\mathbb{G}$ be a cyclic group with generator $g \in \mathbb{G}$. Let $(P_i = g^{s_i}, s_i)$ and $(P_j = g^{s_j}, s_j)$ be the static public/secret key pairs of party i and party j, respectively. During the protocol, both parties exchange ephemeral public keys, *e.g.*, party i sends $X_i = g^{r_i}$ to party j, and party j sends $Y_j = g^{r_j}$ to party i. Then, both parties compute the same key material $k_i = (P_j^d Y_j)^{s_i c + r_i} = g^{(s_i c + r_i)(s_j d + r_j)} = (P_i^c X_i)^{s_j d + r_j} = k_j$ where $c = H_1(j, X)$ and $d = H_1(i, Y)$ are computed by using a function H_1, and use it as input of a key derivation function H_2 to generate a common session key, *i.e.*, $\mathrm{sk}_i = H_2(k_i) = H_2(k_j) = \mathrm{sk}_j$.

As mentioned above, HMQV has many nice properties such as only two-pass messages, implicit key authentication, high efficiency, and without using any explicit entity authentication techniques (*e.g.*, signatures). Our main goal is to construct a lattice-based counterpart such that it not only enjoys all those nice properties of HMQV, but also belongs to post-quantum cryptography, *i.e.*, the underlying hardness assumption is believed to hold even against quantum computer. However, such a task is highly non-trivial since the success of HMQV extremely relies on the nice property of cyclic groups such as commutativity (*i.e.,* $(g^a)^b = (g^b)^a$) and perfect (and public) randomization (*i.e.* g^a can be perfectly randomized by computing $g^a g^r$ with a uniformly chosen r at random).

Fortunately, as noticed in [27,61,9], the Ring-LWE problem actually supports some kind of "approximate" commutativity, and can be used to build passive-secure key exchange protocol. Specifically, let R_q be a ring, and χ be a Gaussian distribution over R_q. Then, given two Ring-LWE tuples with both secret and errors choosing from χ, *e.g.,* $(a, b_1 = as_1 + e_1)$ and $(a, b_2 = as_2 + e_2)$ for randomly chosen $a \leftarrow_r R_q, s_1, s_2, e_1, e_2 \leftarrow_r \chi$, the approximate equation $s_1 b_2 \approx s_1 a s_2 \approx s_2 b_1$ holds with overwhelming probability for proper parameters. By the same observation, we construct an AKE protocol (as illustrated in Fig. 1), where both the static and ephemeral public keys are actually Ring-LWE elements corresponding to a globally public element $a \in R_q$. In order to overcome the inability of "approximate" commutativity, our protocol has to send a signal information w_j computed by using a function Cha. Combining this with another useful function Mod_2, both parties are able to compute the same key material $\sigma_i = \sigma_j$ (from the approximately equal values k_i and k_j) with a guarantee that $\sigma_j = \mathsf{Mod}_2(k_j, w_j)$ has high min-entropy even conditioned on the partial information $w_j = \mathsf{Cha}(k_j)$ of k_j (thus it can be used to derive a uniform session key sk_j).

However, the strategy of sending out the information $w_j = \mathsf{Cha}(k_j)$ inherently brings an undesired byproduct. Specifically, unlike HMQV, the security of our AKE protocol cannot be proven in the CK model which allows the adversaries to obtain the session state k_j via *session state reveal queries*. This is because in a traditional definition of session identifier that consists of all the exchanged messages, the two "different" sessions $\mathrm{sid} = (i, j, x_i, y_j, w_j)$ and $\mathrm{sid}' = (i, j, x_i, y_j, w'_j)$ in our protocol have the same session state, *i.e.,* k_i at party i.[5] This also means that we cannot directly use $\sigma_i = \sigma_j$ as the session key, because the binding between the value of σ_i and the session identifier is too loose (especially for the signal part, w_j's). Since

[5] We remark that this problem might not exist if we consider a different definition of session identifier, *e.g.*, the one that was uniquely determined at the beginning of each execution of the protocol.

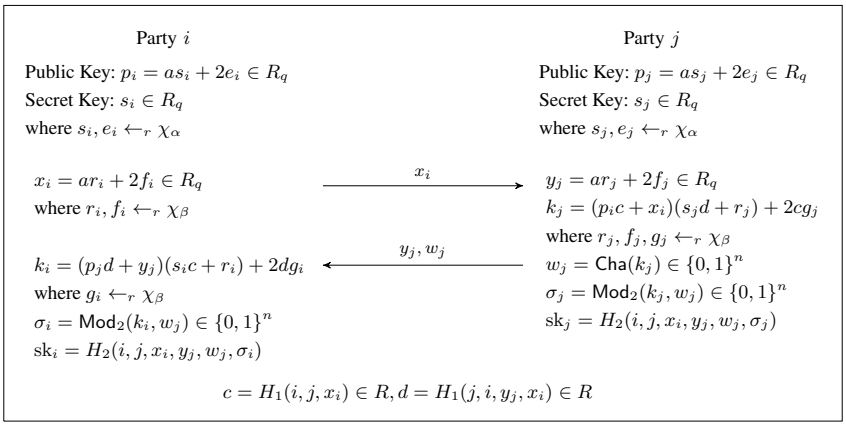

Fig. 1. Our AKE protocol based on Ring-LWE, where $R_q = \mathbb{Z}_q/(x^n + 1)$ is a ring, χ_α and χ_β are two Gaussian distributions over R_q. The two functions Cha and Mod_2 provide that $\sigma_i = \mathsf{Mod}_2(k_i, w_j) = \mathsf{Mod}_2(k_j, w_j) = \sigma_j$.

both sessions sid and sid' have the same session state k_i, the value $\sigma_i' = \mathsf{Mod}_2(k_i, w_j')$ corresponding to sid' is simply a shift of $\sigma_i = \mathsf{Mod}_2(k_i, w_j)$ corresponding to sid (by the definition of the Mod_2 function). We prevent the adversary from utilizing this weakness by setting the session key as the output of the hash function H_2 (which is modeled as a random oracle) which tightly binds the session identifier sid and the key material σ_i (*i.e.*, $\mathrm{sk}_i = H_2(\mathrm{sid}, \sigma_i)$). Our technique works due to another useful property of Mod_2, which guarantees that $\sigma_i' = \mathsf{Mod}_2(k_i, w_j')$ preserves the high min-entropy property of k_i for any w_j' (and thus is enough to generate a secure session key by the property of random oracle H_2).[6]

In order to finally get a security proof of our AKE protocol in the BR model with weakly perfect forward secrecy, we have to make use of the following property of Gaussian distributions namely some kind of "public randomization". Specifically, let χ_α and χ_β be two Gaussian distributions with standard deviation α and β, respectively. Then, the summation of the two distributions is still a Gaussian distribution χ_γ with standard deviation $\gamma = \sqrt{\alpha^2 + \beta^2}$. In particular, if $\beta \gg \alpha$ (*e.g.*, $\beta/\alpha = 2^{\omega(\log \kappa)}$ for some security parameter κ), we have that the distribution χ_γ is statistically close to χ_β. This technique is also known as "noise flooding" and has been applied, for instance, in proving robustness of the LWE assumption [37]. [7] Using this technique allows to statistically hide the distribution of χ_α in a bigger distribution χ_β. The security proof of our protocol is based on this observation, and for now let us keep it in mind that a large distribution will be used to hide a small one.

To better illustrate our technique, we take party j as an example, who combines his static and ephemeral secret keys by computing $\hat{r}_j = s_j d + r_j$ where $d = H_1(j, i, y_j, x_i)$. We notice that the value $\hat{r}_j$ actually behaves like a "signature" on the messages that party j knows so far. In other words, it should be difficult to compute $\hat{r}_j$ if we do not know the corresponding "signing key" s_j. Indeed, this combination is necessary to provide the implicit entity authentication. However, it also posts an obstacle to get a security proof since the simulator may also be unaware of s_j. Fortunately, if the randomness r_j is chosen from a big enough Gaussian

[6] We remark that this is also the reason why the nice reconciliation mechanism in [61] cannot be used in our protocol. Specifically, it is unclear whether the reconciliation function $\mathsf{rec}(\cdot, \cdot)$ in [61] could also preserve the high min-entropy property of the first input (*i.e.*, which might not be uniformly random) for any (maliciously chosen) second input.

[7] Actually, noise flooding works conditioned on the size of the random variable, and thus does not require to be distributed according to χ_α.

distribution, then the value $\hat{r}_j$ almost obliterates all information of s_j. More specifically, the simulator can directly choose $\hat{r}_j$ such that $\hat{r}_j = s_j d + r_j$ for some unknown r_j by computing $y_j = (a\hat{r}_j + 2\hat{f}_j) - p_j d$, and programming the random oracle $d = H_1(j, i, y_j, x_i)$ correspondingly. Combining the properties of Gaussian distributions and the random oracle H_1, we have that y_j is almost identically distributed as that in the real run of the protocol. Now, we check the randomness of $k_j = (p_i c + x_i)\hat{r}_j + 2cg_j$. Note that for the test session, we can always guarantee that at least one of p_i and x_i is honestly generated (and thus is computationally indistinguishable from uniformly distributed element under the Ring-LWE assumption), or else there is no "secrecy" to protect at all if both p_i and x_i are chosen by the adversary. That is, the value $p_i c + x_i$ is always uniformly distributed if c is invertible in R_q. Again, by programming $c = H_1(i, j, x_i)$, the simulator can actually replace $p_i c + x_i$ with $\hat{x}_i = c^{-1} u_i$ for a uniformly distributed ring element u_i. In this case, we have that $k_j = \hat{x}_i \hat{r}_j + 2cg_j = c(u_i \hat{r}_j + 2g_j)$ should be computationally indistinguishable from a uniformly distributed element under the Ring-LWE assumption. In other words, k_j can be used to derive a high min-entropy key material σ_j as required by using the Mod_2 function.

Unfortunately, directly using "noise flooding" has a significant drawback, *i.e.*, the requirement of a super-polynomially big standard deviation β, which may lead to a nightmare for practical performance due to a super-polynomially big modulus q for correctness and a very large ring dimension n for the hardness of the underlying Ring-LWE problems. Fortunately, we can somehow reduce the big cost by further employing the rejection sampling technique [52]. Rejection sampling is a crucial technique in signature schemes to make the distribution of signatures independent of the signing key. Since [52] it has been applied in many other lattice-based signature schemes [38,30,3,40].

In our case the combination of the static and ephemeral secret keys, $\hat{r}_j = s_j d + r_j$, at party j is essentially a signature on all the public messages under party j's public key (we again take party j as an example, but note that similar analysis also holds for party i). Thus, we can freely use the rejection sampling technique to relax the requirement on a super-polynomially big β. In other words, we can use a much smaller β, but require party j to use r_j if $\hat{r}_j = s_j d + r_j$ follows the distribution χ_β, and to resample a new r_j otherwise. We note that by deploying rejection sampling in our AKE it is the first time that rejection sampling is used beyond signature schemes. As for signatures, rejection sampling is done locally, and thus will not affect the interaction between the two parties, *i.e.*, two-pass messages. Even though the computational performance of each execution might become worse with certain (small) probability (due to rejection and repeated sampling), the average computational cost is much better than the setting of using a super-polynomially big β.

1.3 Related Work, Comparison and Discussion

In the past few years, many cryptographers have put effort into constructing different kinds of KE protocols from lattices. At Asiacrypt 2009, Katz and Vaikuntanathan [43] proposed the first password-based authenticated key exchange protocol that can be proven secure based on the LWE assumption. Ding *et al.* [27] proposed a passive-secure KE protocol based on (Ring-)LWE. Like the standard DH protocol, the protocol in [27] could not provide authentication—*i.e.*, it is not an AKE protocol—and is thus weak to man-in-the-middle attacks. Lei *et al.* [49] presented a KE protocol based on NTRU encryption and a new "NTRU-KE" assumption.

To the best of our knowledge, there are four papers focusing on designing AKEs from lattices [32,61,33,9]. In general, all known lattice-based AKE protocols work by following generic transformations from key encapsulation mechanisms (KEM) to AKEs and explicitly using signatures to provide authentication. Fujioka *et al.* [32] proposed a generic construction of AKE from KEMs, which can be proven secure in the CK model. Informally, they showed that if there is a CCA secure KEM with high min-entropy keys and a family

Table 1. Comparison of Lattice-based AKEs (CCA* means CCA-security with high min-entropy keys [32], and EUF-CMA means existential unforgeability under chosen message attacks)

Protocols	KEM/PKE	Signature	Message-pass	Model	RO?	Num. of R_q
FSXY12 [32]	CCA*	-	2-pass	CK	$\times$	$\gg 7$
FSXY13 [33]	OW-CCA	-	2-pass	CK	$\surd$	7
Peikert14 [61]	CPA	EUF-CMA	3-pass	SK-security	$\surd$	> 2 [a]
BCNS14 [9]	CPA	EUF-CMA	4-pass	ACCE	$\surd$	2 for KEM [b]
Ours	-	-	2-pass	BR with wPFS	$\surd$	2

[a] The actual number of ring elements depends on the choice of the concrete lattice-based signatures.
[b] Since the protocol uses traditional signatures to provide authentication, it does not contain any other ring elements.

of pseudorandom functions (PRF), then there is a secure AKE protocol in the standard model. Instantiated with lattice-based CCA secure KEMs such as [62,59], it is possible to construct lattice-based AKE protocols in the standard model. However, as the authors commented, their construction was just of theoretic interest due to huge public keys and the lack of an efficient and direct construction of PRFs from (Ring-)LWE. Following [32], the paper [33] tried to get a practical AKE protocol, and gave a generic construction from any one-way CCA-secure KEM in the random oracle model. The two protocols in [32,33] share some similarities such as having two-pass messages, and involving three times encryptions (*i.e.*, two encryptions under each party's static public keys and one encryption under an ephemeral public key). For concreteness, instantiated with the CPA-secure encryption from Ring-LWE [54] (*i.e.*, by first transforming it into a CCA-secure one using the Fujisaki-Okamoto (FO) transformation in the random oracle model), the protocol in [33] requires to exchange seven ring elements in total.

Recently, Peikert [61] presented an efficient KEM based on Ring-LWE, which was then transformed into an AKE protocol by using the same structure as SIGMA [45]. The resulting protocol involved one encryption, and two signatures and two MACs for explicit entity authentication. As the SIGMA protocol, the protocol in [61] has three-pass messages and was proven SK-secure [16] in the random oracle model. Bos *et al.* [9] treated Peikert's KEM as a DH-like KE protocol, and integrated it into the Transport Layer Security (TLS) protocol. Thus, their AKE protocol also employed signatures to provide explicit authentication. In fact, they used the traditional digital signatures such as RSA and ECDSA to provide authentication (*i.e.,* it is not a pure post-quantum AKE protocol). The security of their protocol was proven in the authenticated and confidential channel establishment (ACCE) security model [42], which is based on the BR model, but has many differences to capture entity authentication and channel security.

Since the lack of concrete security analysis and parameter choices in the literature, we only give a theoretical comparison of lattice-based AKEs in Table 1. In summary, our protocol only has two-pass messages (about two ring elements) and does not use signatures/MACs at al, and its security solely relies on the hardness of Ring-LWE. To the best of our knowledge there is not a single post-quantum authenticated key exchange protocol (until this work) which solely relies on a quantum-hard computational problem and does not make use of explicit cryptographic primitives except hash functions.

1.4 Roadmap

In the preliminaries section, we recall the BR model and several useful tools on lattices. Then, we give a two-pass AKE protocol from ideal lattices in Section 3, and prove its security based on Ring-LWE problems in

Section 4. In Section 5, we present the one-pass variant of our protocol. The concrete choices of parameters and timings are given in Section 6.

2 Preliminaries

2.1 Notation

Let κ be the natural security parameter, and all quantities are implicitly dependent on κ. Let $\mathrm{poly}(\kappa)$ denote an unspecified function $f(\kappa) = O(\kappa^c)$ for some constant c. The function $\log$ denotes the natural logarithm. We use standard notation O, ω to classify the growth of functions. If $f(\kappa) = O(g(\kappa) \cdot \log^c \kappa)$, we denote $f(\kappa) = \tilde{O}(g(\kappa))$. We say a function $f(\kappa)$ is negligible if for every $c > 0$, there exists a N such that $f(\kappa) < 1/\kappa^c$ for all $\kappa > N$. We use $\mathrm{negl}(\kappa)$ to denote a negligible function of κ, and we say a probability is overwhelming if it is $1 - \mathrm{negl}(\kappa)$.

The set of real numbers (integers) is denoted by $\mathbb{R}$ ($\mathbb{Z}$, resp.). We use $\leftarrow_r$ to denote randomly choosing an element from some distribution (or the uniform distribution over some finite set). Vectors are in column form and denoted by bold lower-case letters (*e.g.*, $\mathbf{x}$). The ℓ_2 and ℓ_∞ norms we designate by $\|\cdot\|$ and $\|\cdot\|_\infty$. The ring of polynomials over $\mathbb{Z}$ ($\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$, resp.) we denote by $\mathbb{Z}[x]$ ($\mathbb{Z}_q[x]$, resp.).

Let X be a distribution over finite set S. The min-entropy of X is defined as

$$H_\infty(X) = -\log(\max_{s \in S} \Pr[X = s]).$$

Intuitively, the min-entropy says that if we (privately) choose x from X at random, then no (unbounded) algorithm can guess the value of x correctly with probability greater than $2^{-H_\infty(X)}$.

2.2 Security Model for AKE

We now recall the Bellare-Rogaway security model [8], restricted to the case of two-pass AKE protocol.

Sessions We fix a positive integer N to be the maximum number of honest parties that use the AKE protocol. Each party is uniquely identified by an integer i in $\{1, 2, \ldots, N\}$, and has a static key pair consisting of a static secret key sk_i and static public key pk_i, which is signed by a Certificate Authority (CA). A single run of the protocol is called a *session*. A session is activated at a party by an incoming message of the form (Π, I, i, j) or the form (Π, R, j, i, X_i), where Π is a protocol identifier; I and R are role identifiers; i and j are party identifiers. If party i receives a message of the form (Π, I, i, j), we say that i is the session initiator. Party i then outputs the response X_i intended for party j. If party j receives a message of the form (Π, R, j, i, X_i), we say that j is the session responder; party j then outputs a response Y_j to party i. After exchanging these messages, both parties compute a session key.

If a session is activated at party i with i being the initiator, we associate with it a *session identifier* $\mathrm{sid} = (\Pi, I, i, j, X_i)$ or $\mathrm{sid} = (\Pi, I, i, j, X_i, Y_j)$. Similarly, if a session is activated at party j with j being the responder, the session identifier has the form $\mathrm{sid} = (\Pi, R, j, i, X_i, Y_j)$. For a session identifier $\mathrm{sid} = (\Pi, *, i, j, *[, *])$, the third coordinate—that is, the first party identifier—is called the owner of the session; the other party is called the peer of the session. A session is said to be *completed* when its owner computes a session key. The *matching session* of $\mathrm{sid} = (\Pi, I, i, j, X_i, Y_j)$ is the session with identifier $\widetilde{\mathrm{sid}} = (\Pi, R, j, i, X_i, Y_j)$ and vice versa.

Adversarial Capabilities We model the adversary $\mathcal{A}$ as a probabilistic polynomial time (PPT) Turing machine with full control over all communications channels between parties, including control over session activations. In particular, $\mathcal{A}$ can intercept all messages, read them all, and remove or modify any desired messages as well as inject its own messages. We also suppose $\mathcal{A}$ is capable of obtaining hidden information about the parties, including static secret keys and session keys to model potential leakage of them in genuine protocol executions. These abilities are formalized by providing $\mathcal{A}$ with the following oracles (we split the Send query in [15] into Send_0, Send_1 and Send_2 queries for the case of two-pass protocols):

- $\mathsf{Send}_0(\Pi, I, i, j)$: $\mathcal{A}$ activates party i as an initiator. The oracle returns a message X_i intended for party j.
- $\mathsf{Send}_1(\Pi, R, j, i, X_i)$: $\mathcal{A}$ activates party j as a responder using message X_i. The oracle returns a message Y_j intended for party i.
- $\mathsf{Send}_2(\Pi, R, i, j, X_i, Y_j)$: $\mathcal{A}$ sends party i the message Y_j to complete a session previously activated with a $\mathsf{Send}_0(\Pi, I, i, j)$ query that returned X_i.
- $\mathsf{SessionKeyReveal}(\mathrm{sid})$: The oracle returns the session key associated with the session sid if it has been generated.
- $\mathsf{Corrupt}(i)$: The oracle returns the static secret key belonging to party i. A party whose key is given to $\mathcal{A}$ in this way is called *dishonest*; a party not compromised in this way is called *honest*.
- $\mathsf{Test}(\mathrm{sid}^*)$: 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^*. Note that we impose some restrictions on this query. We only allow $\mathcal{A}$ to query this oracle once, and only on a fresh (see Definition 1) session sid^*.

Definition 1 (Freshness). *Let* $\mathrm{sid}^* = (\Pi, I, i^*, j^*, X_i, Y_j)$ *or* $(\Pi, R, j^*, i^*, X_i, Y_j)$ *be a completed session with initiator party* i^* *and responder party* j^**. If the matching session exists, denote it* $\widetilde{\mathrm{sid}}^*$*. We say that* sid^* *is fresh if the following conditions all hold:*

- $\mathcal{A}$ *has not made a* $\mathsf{SessionKeyReveal}$ *query on* sid^**.*
- $\mathcal{A}$ *has not made a* $\mathsf{SessionKeyReveal}$ *query on* $\widetilde{\mathrm{sid}}^*$ *(if it exists).*
- *Neither party* i^* *nor* j^* *is dishonest if* $\widetilde{\mathrm{sid}}^*$ *does not exist. I.e.,* $\mathcal{A}$ *has not made a* $\mathsf{Corrupt}$ *query on either of them.*

Recall that in the original BR model [8], no corruption query is allowed. In the above freshness definition, we allow the adversary to corrupt both parties of sid^* if the matching session exists, *i.e.*, the adversary can obtain the parties's secret key in advance and then passively eavesdrops the session sid^* (and thus $\widetilde{\mathrm{sid}}^*$). We remark that this is actually stronger than what is needed for capturing wPFS [46], where the adversary is only allowed to corrupt a party after an honest session sid^* (and thus $\widetilde{\mathrm{sid}}^*$) has been completed.

Security Game The security of a two-pass AKE protocol is defined in terms of the following game. The adversary $\mathcal{A}$ makes any sequence of queries to the oracles above, so long as only one Test query is made on a fresh session, as mentioned above. The game ends when $\mathcal{A}$ outputs a guess b' for b. We say $\mathcal{A}$ wins the game if its guess is correct, so that $b' = b$. The advantage of $\mathcal{A}$, $\mathbf{Adv}_{\Pi,\mathcal{A}}$, is defined as $\Pr[b' = b] - 1/2$.

Definition 2 (Security). *We say that an AKE protocol* Π *is secure if the following conditions hold:*

- *If two honest parties complete matching sessions then they compute the same session key with overwhelming probability.*
- *For any PPT adversary* $\mathcal{A}$*, the advantage* $\mathbf{Adv}_{\Pi,\mathcal{A}}$ *is negligible.*

2.3 The Gaussian Distributions and Rejection Sampling

For any positive real $\alpha \in \mathbb{R}$, and vectors $\mathbf{c} \in \mathbb{R}^m$, the continuous Gaussian distribution over $\mathbb{R}^m$ with standard deviation α centered at $\mathbf{v}$ is defined by the probability function $\rho_{\alpha,\mathbf{c}}(\mathbf{x}) = (\frac{1}{\sqrt{2\pi\sigma^2}})^m \exp\left(\frac{-\|\mathbf{x}-\mathbf{v}\|^2}{2\sigma^2}\right)$. For integer vectors $\mathbf{c} \in \mathbb{R}^n$, let $\rho_{s,\mathbf{c}}(\mathbb{Z}^m) = \sum_{\mathbf{x}\in\mathbb{Z}^m} \rho_{s,\mathbf{c}}(\mathbf{x})$. Then, we define the discrete Gaussian distribution over $\mathbb{Z}^m$ as $D_{\mathbb{Z}^m,s,\mathbf{c}}(\mathbf{x}) = \frac{\rho_{s,\mathbf{c}}(\mathbf{x})}{\rho_{s,\mathbf{c}}(\mathbb{Z}^m)}$, where $\mathbf{x} \in \mathbb{Z}^m$. The subscripts s and $\mathbf{c}$ are taken to be 1 and $\mathbf{0}$ (respectively) when omitted. The following lemma says that for large enough α, almost all the samples from $D_{\mathbb{Z}^m,\alpha}$ are small.

Lemma 1 ([58]). *Letting real $\alpha = \omega(\sqrt{\log m})$, constant $d > 1/\sqrt{2\pi}$, then $\Pr_{\mathbf{x}\leftarrow_r D_{\mathbb{Z}^m,\alpha}}[\|\mathbf{x}\| > d\cdot\alpha\sqrt{m}] \leq \frac{1}{2}D^n$, where $D = d\sqrt{2\pi e}\cdot e^{-\pi\cdot d^2}$. In particular, we have $\Pr_{\mathbf{x}\leftarrow_r D_{\mathbb{Z}^m,\alpha}}[\|\mathbf{x}\| > \alpha\sqrt{m}] \leq 2^{-m+1}$.*

Now, we recall rejection sampling in Theorem 1 from [52], which will be used in the security proof of our AKE protocol. Informally, the rejection sampling theorem says that for large enough α, the distributions $D_{\mathbb{Z}^m,\alpha,\mathbf{c}}$ and $D_{\mathbb{Z}^m,\alpha}$ are statistically indistinguishable even given vector $\mathbf{c} \in \mathbb{Z}$.

Theorem 1 (Rejection Sampling [52]). *Let V be a subset of $\mathbb{Z}^m$ in which all the elements have norms less than T, $\alpha = \omega(T\sqrt{\log m})$ be a real, and $\psi : V \to \mathbb{R}$ be a probability distribution. Then there exists a constant $M = O(1)$ such that the distribution of the following algorithm Samp_1 :*

1: $\mathbf{c} \leftarrow_r \psi$
2: $\mathbf{z} \leftarrow_r D_{\mathbb{Z}^m,\alpha,\mathbf{c}}$
3: output $(\mathbf{z},\mathbf{c})$ *with probability* $\min\left(\frac{D_{\mathbb{Z}^m,\alpha}(\mathbf{z})}{MD_{\mathbb{Z}^m,\alpha,\mathbf{c}}(\mathbf{z})}, 1\right)$.

is within statistical distance $\frac{2^{-\omega(\log m)}}{M}$ of the distribution of the following algorithm Samp_2 :

1: $\mathbf{c} \leftarrow_r \psi$
2: $\mathbf{z} \leftarrow_r D_{\mathbb{Z}^m,\alpha}$
3: output $(\mathbf{z},\mathbf{c})$ *with probability* $1/M$.

Moreover, the probability that Samp_1 outputs something is at least $\frac{1-2^{-\omega(\log m)}}{M}$. More concretely, if $\alpha = \tau T$ for any positive τ, then $M = e^{12/\tau+1/(2\tau^2)}$ and the output of algorithm Samp_1 is within statistical distance $\frac{2^{-100}}{M}$ of the output of Samp_2, and the probability that $\mathcal{A}$ outputs something is at least $\frac{1-2^{-100}}{M}$.

2.4 Ring Learning with Errors

Let the integer n be a power of 2, and consider the ring $R = \mathbb{Z}[x]/(x^n+1)$. For any positive integer q, we define the ring $R_q = \mathbb{Z}_q[x]/(x^n+1)$ analogously. For any polynomial $y(x)$ in R (or R_q), we identify y with its coefficient vector in $\mathbb{Z}^n$ (or $\mathbb{Z}_q^n$). Then we define the norm of a polynomial to be the norm of its coefficient vector.

Lemma 2. *For any $s,t \in R$, we have $\|s\cdot t\| \leq \sqrt{n}\cdot\|s\|\cdot\|t\|$ and $\|s\cdot t\|_\infty \leq n\cdot\|s\|_\infty\cdot\|t\|_\infty$.*

Besides, the discrete Gaussian distribution over the ring R can be naturally defined as the distribution of ring elements whose coefficient vectors are distributed according to the discrete Gaussian distribution over $\mathbb{Z}^n$, *e.g.*, $D_{\mathbb{Z}^n,\alpha}$ for some positive real α. Letting χ_α be the discrete Gaussian distribution over $\mathbb{Z}^n$ with standard deviation α centered at $\mathbf{0}$, *i.e.,* $\chi_\alpha := D_{\mathbb{Z}^n,\alpha}$, we now adopt the following notational convention: since bold-face letters denote vectors, $\mathbf{x} \leftarrow_r \chi_\alpha$ means we sample the vector $\mathbf{x}$ from the distribution χ_α; for

normal weight variables (*e.g.* $y \leftarrow_r \chi_\alpha$) we sample an element of R whose coefficient vector is distributed according to χ_α.

Now we come to the statement of the Ring-LWE assumption; we will use a special case detailed in [54]. Let R_q be defined as above, and $s \leftarrow_r R_q$. We define A_{s,χ_α} to be the distribution of the pair $(a, as + x) \in R_q \times R_q$, where $a \leftarrow_r R_q$ is uniformly chosen and $x \leftarrow_r \chi_\alpha$ is independent of a.

Definition 3 (Ring-LWE Assumption). *Let R_q and χ_α be defined as above, and let $s \leftarrow_r R_q$. The Ring-LWE assumption RLWE$_{q,\alpha}$ states that it is hard for any PPT algorithm to distinguish A_{s,χ_α} from the uniform distribution on $R_q \times R_q$ with only polynomially many samples.*

The following lemma says that the hardness of the Ring-LWE assumption can be reduced to some hard lattice problems such as the Shortest Independet Vectors Problem (SIVP) over ideal lattices.

Proposition 1 (A special case of [54]). *Let n be a power of 2, let α be a real number in $(0, 1)$, and q a prime such that $q \bmod 2n = 1$ and $\alpha q > \omega(\sqrt{\log n})$. Define $R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$ as above. Then there exists a polynomial time quantum reduction from $\tilde{O}(\sqrt{n}/\alpha)$-SIVP in the worst case to average-case RLWE$_{q,\beta}$ with ℓ samples, where $\beta = \alpha q \cdot (n\ell/\log(n\ell))^{1/4}$.*

It has been proven that the Ring-LWE assumption still holds even if the secret s is chosen according to the error distribution χ_β rather than uniformly [1,54]. This variant is known as the *normal form*, and is preferable for controlling the size of the error term [11,10]. The underlying Ring-LWE assumption also holds when scaling the error by a constant t relatively prime to q [11], *i.e.*, using the pair $(a_i, a_i s + t x_i)$ rather than $(a_i, a_i s + x_i)$. Several lattice-based cryptographic schemes have been constructed based on this variant [11,10]. In our case, we will fix $t = 2$. Besides, recall that the RLWE$_{q,\beta}$ assumption guarantees that for some prior fixed (but randomly chosen) s, the tuple $(a, as + 2x)$ is computationally indistinguishable from the uniform distribution over $R_q \times R_q$ if $a \leftarrow_r R_q$ and $x \leftarrow \chi_\beta$. In this paper, we will use a matrix form ring-LWE assumption. Formally, let $B_{\chi_\beta,\ell_1,\ell_2}$ be the distribution of $(\mathbf{a}, \mathbf{B} = (b_{i,j})) \in R_q^{\ell_1} \times R_q^{\ell_1 \times \ell_2}$, where $\mathbf{a} = (a_0, \ldots, a_{\ell_1 - 1}) \leftarrow_r R_q^{\ell_1}$, $\mathbf{s} = (s_0, \ldots, s_{\ell_2 - 1}) \leftarrow_r R_q^{\ell_2}$, $e_{i,j} \leftarrow_r \chi_\beta$, and $b_{i,j} = a_i s_j + 2e_{i,j}$ for $i \in \{0, \ldots, \ell_1 - 1\}$ and $j \in \{0, \ldots, \ell_2 - 1\}$. For polynomially bounded ℓ_1 and ℓ_2, one can show that the distribution of $B_{\chi_\beta,\ell_1,\ell_2}$ is pseudorandom based on the RLWE$_{q,\beta}$ assumption [62].

3 Authenticated Key Exchange from Ring-LWE

We now introduce some notation before presenting our protocol. For odd prime $q > 2$, denote $\mathbb{Z}_q = \{-\frac{q-1}{2}, \ldots, \frac{q-1}{2}\}$ and define the subset $E := \{-\lfloor \frac{q}{4} \rfloor, \ldots, \lfloor \frac{q}{4} \rceil\}$ as the middle half of $\mathbb{Z}_q$. We also define Cha to be the characteristic function of *the complement of* E, so $\mathsf{Cha}(v) = 0$ if $v \in E$ and 1 otherwise. Obviously, for any v in $\mathbb{Z}_q$, $v + \mathsf{Cha}(v) \cdot \frac{q-1}{2} \bmod q$ belongs to E. We define an auxiliary modular function, $\mathsf{Mod}_2 \colon \mathbb{Z}_q \times \{0, 1\} \to \{0, 1\}$:

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

In the following lemma, we show that given the bit $b = \mathsf{Cha}(v)$, and a value $w = v + 2e$ with sufficiently small e, we can recover $\mathsf{Mod}_2(v, \mathsf{Cha}(v))$. In particular, we have $\mathsf{Mod}_2(v, b) = \mathsf{Mod}_2(w, b)$.

Lemma 3. *Let q be an odd prime, $v \in \mathbb{Z}_q$ and $e \in \mathbb{Z}_q$ such that $|e| < q/8$. Then, for $w = v + 2e$, we have $\mathsf{Mod}_2(v, \mathsf{Cha}(v)) = \mathsf{Mod}_2(w, \mathsf{Cha}(v))$.*

Proof. Note that $w + \mathsf{Cha}(v)\frac{q-1}{2} \bmod q = v + \mathsf{Cha}(v)\frac{q-1}{2} + 2e \bmod q$. Now, $v + \mathsf{Cha}(v)\frac{q-1}{2} \bmod q$ is in E as we stated above; that is, $-\lfloor \frac{q}{4} \rfloor \leq v + \mathsf{Cha}(v)\frac{q-1}{2} \bmod q \leq \lfloor \frac{q}{4} \rceil$. Thus, since $-q/8 < e < q/8$, we have $-\lfloor \frac{q}{2} \rfloor \leq v + \mathsf{Cha}(v)\frac{q-1}{2} \bmod q + 2e \leq \lfloor \frac{q}{2} \rceil$. Therefore, we have $v + \mathsf{Cha}(v)\frac{q-1}{2} \bmod q + 2e = v + \mathsf{Cha}(v)\frac{q-1}{2} + 2e \bmod q = w + \mathsf{Cha}(v)\frac{q-1}{2} \bmod q$. Thus, $\mathsf{Mod}_2(w, \mathsf{Cha}(v)) = \mathsf{Mod}_2(v, \mathsf{Cha}(v))$.

Now, we extend the functions Cha and Mod_2 to ring R_q by applying them coefficient-wise to ring elements. Namely, for ring element $v = (v_0, \ldots, v_{n-1}) \in R_q$ and binary-vector $\mathbf{b} = (b_0, \ldots, b_{n-1}) \in \{0,1\}^n$, define $\widetilde{\mathsf{Cha}}(v) = (\mathsf{Cha}(v_0), \ldots, \mathsf{Cha}(v_{n-1}))$ and $\widetilde{\mathsf{Mod}}_2(v, \mathbf{b}) = (\mathsf{Mod}_2(v_0, b_0), \ldots, \mathsf{Mod}_2(v_{n-1}, b_{n-1}))$. For simplicity, we slightly abuse the notations and still use Cha and Mod_2 to denote $\widetilde{\mathsf{Cha}}$ and $\widetilde{\mathsf{Mod}}_2$, respectively. Clearly, the result in Lemma 3 still holds when extending to ring elements.

In our AKE protocol, the two involved parties will use Cha and Mod_2 to derive a common key material. Concretely, the responder will publicly send the result of Cha on his own secret ring element to the initiator in order to compute a shared key material from two "closed" ring elements (by applying the Mod_2 function). Ideally, for uniformly chosen element v from R_q at random, we hope that the output of $\mathsf{Mod}_2(v, \mathsf{Cha}(v))$ is uniformly distributed $\{0,1\}^n$. However, this can never happen when q is a odd prime. Fortunately, we can show that the output of $\mathsf{Mod}_2(v, \mathsf{Cha}(v))$ conditioned on $\mathsf{Cha}(v)$ has high min-entropy, thus can be used to extract an (almost) uniformly session key. Actually, we can prove a stronger result.

Lemma 4. *Let q be any odd prime and R_q be the ring defined above. Then, for any $\mathbf{b} \in \{0,1\}^n$ and any $v' \in R_q$, the output distribution of $\mathsf{Mod}_2(v + v', \mathbf{b})$ given $\mathsf{Cha}(v)$ has min-entropy at least $-n\log(\frac{1}{2} + \frac{1}{|E|-1})$, where v is uniformly chosen from R_q at random. In particular, when $q > 203$, we have $-n\log(\frac{1}{2} + \frac{1}{|E|-1}) > 0.97n$.*

Proof. Since each coefficient of v is independently and uniformly chosen from $\mathbb{Z}_q$ at random, we can simplify the proof by focusing on the first coefficient of v. Formally, letting $v = (v_0, \ldots, v_{n-1})$, $v' = (v'_0, \ldots, v'_{n-1})$ and $\mathbf{b} = (b_0, \ldots, b_{n-1})$, we condition on $\mathsf{Cha}(v_0)$:

- If $\mathsf{Cha}(v_0) = 0$, then $v_0 + v'_0 + b_0 \cdot \frac{q-1}{2}$ is uniformly distributed over $v'_0 + b_0 \cdot \frac{q-1}{2} + E \bmod q$. This shifted set has $(q+1)/2$ elements, which are either consecutive integers—if the shift is small enough—or two sets of consecutive integers—if the shift is large enough to cause wrap-around. Thus, we must distinguish a few cases:
  - If $|E|$ is even and no wrap-around occurs, then the result of $\mathsf{Mod}_2(v_0 + v'_0, b_0)$ is clearly uniform on $\{0,1\}$. Namely, the result of $\mathsf{Mod}_2(v_0 + v'_0, b_0)$ has no bias.
  - If $|E|$ is odd and no wrap-around occurs, then the result of $\mathsf{Mod}_2(v_0 + v'_0, b_0)$ has a bias with probability $\frac{1}{2|E|}$ over $\{0,1\}$. In other words, the $\mathsf{Mod}_2(v_0 + v'_0, b_0)$ will output either 0 or 1 with probability exactly $\frac{1}{2} + \frac{1}{2|E|}$.
  - If $|E|$ is odd and wrap-around does occur, then the set $v'_0 + b_0 \cdot \frac{q-1}{2} + E \bmod q$ splits into two parts, one with an even number of elements, and one with an odd number of elements. This leads to the same situation as with no wrap-around.
  - If $|E|$ is even and wrap-around occurs, then our sample space is split into either two even-sized sets, or two odd sized sets. If both are even, then once again the result of $\mathsf{Mod}_2(v_0 + v'_0, b_0)$ is uniform. If both are odd, it is easy to calculate that the result of $\mathsf{Mod}_2(v_0 + v'_0, b_0)$ has a bias with probability $\frac{1}{|E|}$ over $\{0,1\}$.
- If $\mathsf{Cha}(v_0) = 1$, $v_0 + v'_0 + b_0 \cdot \frac{q-1}{2}$ is uniformly distributed over $v'_0 + b_0 \cdot \frac{q-1}{2} + \tilde{E}$, where $\tilde{E} = \mathbb{Z}_q \setminus E$. Now $|\tilde{E}| = |E| - 1$, so by splitting into the same cases as $\mathsf{Cha}(v_0) = 0$, the result of $\mathsf{Mod}_2(v_0 + v'_0, b)$ has a bias with probability $\frac{1}{|E|-1}$ over $\{0,1\}$.

In all, we have that the result of $\mathsf{Mod}_2(v_0 + v_0', b_0)$ conditioned on $\mathsf{Cha}(v_0)$ has min-entropy at least $-\log(\frac{1}{2} + \frac{1}{|E|-1})$. Since the bits in the result of $\mathsf{Mod}_2(v + v', \mathbf{b})$ are independent, we have that given $\mathsf{Cha}(v)$, the min-entropy $H_\infty(\mathsf{Mod}_2(v + v', \mathbf{b})) \geq -n\log(\frac{1}{2} + \frac{1}{|E|-1})$. This completes the first claim. The second claim directly follows from the fact that $-\log(\frac{1}{2} + \frac{1}{|E|-1}) > -\log(0.51) > 0.97$ when $q > 203$. □

Remark 1 (On Uniformly Distributed Keys). It is known that randomness extractor can be used to obtain an almost uniformly distributed key from a biased bit-string with high min-entropy [19,66,67,28,4]. In practice, as recommended by NIST [5], one can actually use the standard cryptographic hash functions such as SHA-2 to derive a uniformly distributed key if the source string has at least 2κ min-entropy, where κ is the length of the cryptographic hash function.

3.1 The Protocol

We now describe our protocol in detail. Let n be a power of 2, and q be an odd prime such that $q \bmod 2n = 1$. Take $R = \mathbb{Z}[x]/(x^n+1)$ and $R_q = \mathbb{Z}_q[x]/(x^n+1)$ as above. For $\gamma \in \mathbb{R}^+$, let $H_1 \colon \{0,1\}^* \to \chi_\gamma = D_{\mathbb{Z}^n,\gamma}$ be a hash function that always output invertible elements in R_q.[8] Let $H_2 \colon \{0,1\}^* \to \{0,1\}^\kappa$ be the key derivation function, where κ is the bit-length of the final shared key. We model both functions as random oracles [7]. Let χ_α, χ_β be two discrete Gaussian distributions with parameters $\alpha, \beta \in \mathbb{R}^+$. Let $a \in R_q$ be the global public parameter uniformly chosen from R_q at random, and M be a constant determined by Theorem 1. Let $p_i = as_i + 2e_i \in R_q$ be party i's static public key, where (s_i, e_i) is the corresponding static secret key; both s_i and e_i are taken from the distribution χ_α. Similarly, party j has static public key $p_j = as_j + 2e_j$ and static secret key (s_j, e_j).

Initiation Party i proceeds as follows:

1. Sample $r_i, f_i \leftarrow_r \chi_\beta$ and compute $x_i = ar_i + 2f_i$;
2. Compute $c = H_1(i, j, x_i)$, $\hat{r}_i = s_i c + r_i$ and $\hat{f}_i = e_i c + f_i$;
3. Letting $\mathbf{z} \in \mathbb{Z}^{2n}$ be the coefficient vector of $\hat{r}_i$ concatenated with the coefficient vector of $\hat{f}_i$, and $\mathbf{z}_1 \in \mathbb{Z}^{2n}$ be the coefficient vector of $s_i c$ concatenated with the coefficient vector of $e_i c$, repeat the steps $1 \sim 3$ with probability $1 - \min\left(\frac{D_{\mathbb{Z}^{2n},\beta}(\mathbf{z})}{MD_{\mathbb{Z}^{2n},\beta,\mathbf{z}_1}(\mathbf{z})}, 1\right)$.
4. Send x_i to party j.

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

1′. Sample $r_j, f_j \leftarrow_r \chi_\beta$ and compute $y_j = ar_j + 2f_j$;
2′. Compute $d = H_1(j, i, y_j, x_i)$, $\hat{r}_j = s_j d + r_j$ and $\hat{f}_j = e_j d + f_j$;
3′. Letting $\mathbf{z} \in \mathbb{Z}^{2n}$ be the coefficient vector of $\hat{r}_j$ concatenated with the coefficient vector of $\hat{f}_j$, and $\mathbf{z}_1 \in \mathbb{Z}^{2n}$ be the coefficient vector of $s_j d$ concatenated with the coefficient vector of $e_j d$, repeat the steps $1' \sim 3'$ with probability $1 - \min\left(\frac{D_{\mathbb{Z}^{2n},\beta}(\mathbf{z})}{MD_{\mathbb{Z}^{2n},\beta,\mathbf{z}_1}(\mathbf{z})}, 1\right)$;
4′. Sample $g_j \leftarrow_r \chi_\beta$ and compute $k_j = (p_i c + x_i)\hat{r}_j + 2cg_j$ where $c = H_1(i, j, x_i)$;
5′. Compute $w_j = \mathsf{Cha}(k_j) \in \{0,1\}^n$ and send (y_j, w_j) to party i;
6′. Compute $\sigma_j = \mathsf{Mod}_2(k_j, w_j)$ and derive the session key $\mathrm{sk}_j = H_2(i, j, x_i, y_j, w_j, \sigma_j)$.

Finish Party i receives the pair (y_j, w_j) from party j, and proceeds as follows:

[8] In practice, one can first use a hash function such as SHA-2 to obtain a uniformly random string, and then use it to sample from $D_{\mathbb{Z}^n,\gamma}$. The algorithm output a sample only if it is invertible in R_q, otherwise, it tries another sample and repeats. By Lemma 10 in [65], we can have a good probability to sample an invertible element in each trial for an appropriate choice of γ.

5. Sample $g_i \leftarrow_r \chi_\beta$ and compute $k_i = (p_j d + y_j)\hat{r}_i + 2dg_i$ where $d = H_1(j, i, y_j, x_i)$;
6. Compute $\sigma_i = \mathsf{Mod}_2(k_i, w_j)$ and derive the session key $\mathrm{sk}_i = H_2(i, j, x_i, y_j, w_j, \sigma_i)$.

In the above protocol, both parties will make use of rejection sampling, *i.e.,* they will repeat the first three steps with certain probability. By Theorem 1, the probability that each party will repeat the steps with probability about $\frac{1}{M}$ for some constant M and appropriately chosen β. Thus, one can hope that both parties will send something to each other after an averaged M times repetitions of the first three steps. In the following subsection, we will show that once they send something to each other, both parties will finally compute a shared session key.

3.2 Correctness

To show the correctness of our AKE protocol, *i.e.*, that both parties compute the same session key $\mathrm{sk}_i = \mathrm{sk}_j$, it suffices to show that $\sigma_i = \sigma_j$. Since σ_i and σ_j are both the output of Mod_2 with $\mathsf{Cha}(k_j)$ as the second argument, we need only to show that k_i and k_j are sufficiently close by Lemma 3. Note that the two parties will compute k_i and k_j as follows:

$$\begin{aligned} k_i &= (p_j d + y_j)\hat{r}_i + 2dg_i \\ &= a(s_j d + r_j)\hat{r}_i + 2(e_j d + f_j)\hat{r}_i + 2dg_i \\ &= a\hat{r}_i\hat{r}_j + 2\widetilde{g}_i \end{aligned} \qquad \begin{aligned} k_j &= (p_i c + x_i)\hat{r}_j + 2cg_j \\ &= a(s_i c + r_i)\hat{r}_j + 2(e_i c + f_i)\hat{r}_j + 2cg_j \\ &= a\hat{r}_i\hat{r}_j + 2\widetilde{g}_j \end{aligned}$$

where $\widetilde{g}_i = \hat{f}_j\hat{r}_i + dg_i$, and $\widetilde{g}_j = \hat{f}_i\hat{r}_j + cg_j$. Then $k_i = k_j + 2(\widetilde{g}_i - \widetilde{g}_j)$, and we have $\sigma_i = \sigma_j$ if $\|\widetilde{g}_i - \widetilde{g}_j\|_\infty < q/8$ by Lemma 3.

4 Security

Theorem 2. *Let n be a power of 2 satisfying $0.97n \geq 2\kappa$, prime $q > 203$ satisfying $q = 1 \mod 2n$, $\beta = \omega(\alpha\gamma n\sqrt{n \log n})$. Then, if RLWE$_{q,\alpha}$ is hard, the proposed AKE is secure with respect to Definition 2 in the random oracle model.*

The intuition behind our proof is quite simple. Since the public element a and the public key of each party (*e.g.*, $p_i = as_i + 2e_i$) actually consist of a RLWE$_{q,\alpha}$ tuple with Gaussian parameter α (scaled by 2), the parties' static public keys are computationally indistinguishable from uniformly distributed elements in R_q under the Ring-LWE assumption. Similarly, both the exchanged elements x_i and y_j are also computationally indistinguishable from uniformly distributed elements in R_q under the RLWE$_{q,\beta}$ assumption.

Without loss of generality, we take party j as an example to check the distribution of the session key. Note that if k_j is uniformly distributed over R_q, we have $\sigma_j \in \{0,1\}^n$ has high min-entropy even conditioned on w_j by Lemma 4 (*e.g.,* $0.97n > 2\kappa$). Since H_2 is a random oracle, we have that sk_j is uniformly distributed over $\{0,1\}^\kappa$ as expected. Now, let's check the distribution of $k_j = (p_i c + x_i)(s_j d + r_j) + 2cg_j$. As one can imagine, we want to establish the randomness of k_j based on pseudorandomness of "Ring-LWE samples" with public element $\hat{a}_j = c^{-1}(p_i c + x_i) = p_i + c^{-1}x_i$, the secret $\hat{s}_j = s_j d + r_j$, as well as the error term $2g_j$ (thus we have $k_j = c(\hat{a}_j\hat{s}_j + 2g_j)$). Actually, k_j is pseudorandom due to the following fact: 1) c is invertible in R_q; 2) $\hat{a}_j$ is uniformly distributed over R_q whenever p_i or x_i is uniform, and $\hat{s}_j$ has distribution statistically close to χ_β by the strategy of rejection sampling in Theorem 1. In other words, $\hat{a}_j\hat{s}_j + 2g_j$ is statistically close to a RLWE$_{q,\beta}$ sample, and thus is pseudorandom.

Formally, let N be the maximum number of parties, and m be maximum number of sessions for each party. We distinguish the following five types of adversaries:

Type I: $\mathrm{sid}^* = (\Pi, I, i^*, j^*, x_{i^*}, (y_{j^*}, w_{j^*}))$ is the test session, and y_{j^*} is output by a session activated at party j by a $\mathsf{Send}_1(\Pi, R, j^*, i^*, x_{i^*})$ query.

Type II: $\mathrm{sid}^* = (\Pi, I, i^*, j^*, x_{i^*}, (y_{j^*}, w_{j^*}))$ is the test session, and y_{j^*} is **not** output by a session activated at party j^* by a $\mathsf{Send}_1(\Pi, R, j^*, i^*, x_{i^*})$ query.

Type III: $\mathrm{sid}^* = (\Pi, R, j^*, i^*, x_{i^*}, (y_{j^*}, w_{j^*}))$ is the test session, and x_{i^*} is **not** output by a session activated at party i^* by a $\mathsf{Send}_0(\Pi, I, i^*, j^*)$ query.

Type IV: $\mathrm{sid}^* = (\Pi, R, j^*, i^*, x_{i^*}, (y_{j^*}, w_{j^*}))$ is the test session, and x_{i^*} is output by a session activated at party i^* by a $\mathsf{Send}_0(\Pi, I, i^*, j^*)$ query, but i^* either never completes the session, or i^* completes it with exact y_{j^*}.

Type V: $\mathrm{sid}^* = (\Pi, R, j^*, i^*, x_{i^*}, (y_{j^*}, w_{j^*}))$ is the test session, and x_{i^*} is output by a session activated at party i^* by a $\mathsf{Send}_0(\Pi, I, i^*, j^*)$ query, but i^* completes the session with another $y'_j \neq y_{j^*}$.

The five types of adversaries give a complete partition of all the adversaries. The weak perfect forward secrecy (wPFS) is captured by allowing **Type I** and **Type IV** adversaries to obtain the static secret keys of both party i^* and j^* by using $\mathsf{Corrupt}$ queries. Since sid^* definitely has no matching session for **Type II**, **Type III**, and **Type V** adversaries, no corruption to either party i^* or party j^* is allowed by Definition 1. The security proofs for the five types of adversaries are similar, except the forking lemma [6] is involved for **Type II**, **Type III**, and **Type V** adversaries by using the assumption that H_1 is a random oracle. Informally, the adversary must first "commit" x_i (or y_j) before seeing c (or d), thus it cannot determine the value $p_i c + x_i$ (or $p_j d + y_i$) in advance (but the simulator can determine the values by programming H_1 when it tries to embed Ring-LWE instances with respect to either $p_i c + x_i$ or $p_j d + y_i$ as discussed before).

4.1 Type I Adversary

In this subsection, we prove that our AKE is secure against any PPT **Type I** adversary $\mathcal{A}$.

Lemma 5. *Let n be a power of 2 satisfying $0.97n \geq 2\kappa$, prime $q > 203$ satisfying $q = 1 \mod 2n$, $\beta = \omega(\alpha\gamma n\sqrt{n \log n})$. Then, if RLWE$_{q,\alpha}$ is hard, the proposed AKE is secure against any PPT **Type I** adversary $\mathcal{A}$ in the random oracle model.*

Proof. We prove this lemma via a sequence of games $G_{1,l}$ for $0 \leq l \leq 7$. Boxes are used to highlight the changes of each game with respect to its previous game.

Game $G_{1,0}$ $\mathcal{S}$ chooses $i^*, j^* \leftarrow_r \{1, \ldots, N\}$, $s_{i^*}, s_{j^*} \leftarrow_r \{1, \ldots, m\}$, and hopes that the adversary will use $\mathrm{sid}^* = (\Pi, I, i^*, j^*, x_{i^*}, (y_{j^*}, w_{j^*}))$ as the test session, where x_{i^*} is output by the s_{i^*}-th session of party i^*, and y_{j^*} is output by the s_j^*-th session of party j^* activated by a $\mathsf{Send}_1(\Pi, R, j^*, i^*, x_{i^*})$ query. Then, $\mathcal{S}$ chooses $a \leftarrow_r R_q$, generates static public keys for all parties (by choosing $s_i, e_i \leftarrow_r \chi_\alpha$), and simulates the security game for $\mathcal{A}$. Specifically, $\mathcal{S}$ maintains two tables L_1, L_2 for the random oracles H_1, H_2, respectively, and answers the queries from $\mathcal{A}$ as follows:

- $H_1(in)$: If there doesn't exist a tuple (in, out) in L_1, choose an element $out \leftarrow_r \chi_\gamma$, and add (in, out) into L_1. Then, return out to $\mathcal{A}$.
- $H_2(in)$ queries: If there doesn't exist a tuple (in, out) in L_2, choose a vector $out \leftarrow_r \{0,1\}^\kappa$, and add (in, out) into L_2. Then, return out to $\mathcal{A}$.
- $\mathsf{Send}_0(\Pi, I, i, j)$: $\mathcal{A}$ activates a new session of i with intended party j, $\mathcal{S}$ proceeds as follows:
  1. Sample $r_i, f_i \leftarrow_r \chi_\beta$ and compute $x_i = ar_i + 2f_i$;
  2. Compute $c = H_1(i, j, x_i)$, $\hat{r}_i = s_i c + r_i$ and $\hat{f}_i = e_i c + f_i$;

3. Letting $\mathbf{z} \in \mathbb{Z}^{2n}$ be the coefficient vector of $\hat{r}_i$ concatenated with the coefficient vector of $\hat{f}_i$, and $\mathbf{z}_1 \in \mathbb{Z}^{2n}$ be the coefficient vector of $s_i c$ concatenated with the coefficient vector of $e_i c$, repeat the steps $1 \sim 3$ with probability $1 - \min\left(\frac{D_{\mathbb{Z}^{2n},\beta}(\mathbf{z})}{MD_{\mathbb{Z}^{2n},\beta,\mathbf{z}_1}(\mathbf{z})}, 1\right)$.
4. Return x_i to $\mathcal{A}$;

– $\mathsf{Send}_1(\Pi, R, j, i, x_i)$: $\mathcal{S}$ proceeds as follows:

$1'$. Sample $r_j, f_j \leftarrow_r \chi_\beta$ and compute $y_j = ar_j + 2f_j$;
$2'$. Compute $d = H_1(j, i, y_j, x_i)$, $\hat{r}_j = s_j d + r_j$ and $\hat{f}_j = e_j d + f_j$;
$3'$. Letting $\mathbf{z} \in \mathbb{Z}^{2n}$ be the coefficient vector of $\hat{r}_j$ concatenated with the coefficient vector of $\hat{f}_j$, and $\mathbf{z}_1 \in \mathbb{Z}^{2n}$ be the coefficient vector of $s_j d$ concatenated with the coefficient vector of $e_j d$, repeat the steps $1' \sim 3'$ with probability $1 - \min\left(\frac{D_{\mathbb{Z}^{2n},\beta}(\mathbf{z})}{MD_{\mathbb{Z}^{2n},\beta,\mathbf{z}_1}(\mathbf{z})}, 1\right)$;
$4'$. Sample $g_j \leftarrow_r \chi_\beta$ and compute $k_j = (p_i c + x_i)\hat{r}_j + 2cg_j$ where $c = H_1(i, j, x_i)$;
$5'$. Compute $w_j = \mathsf{Cha}(k_j) \in \{0,1\}^n$ and return (y_j, w_j) to $\mathcal{A}$;
$6'$. Compute $\sigma_j = \mathsf{Mod}_2(k_j, w_j)$ and derive the session key $\mathrm{sk}_j = H_2(i, j, x_i, y_j, w_j, \sigma_j)$.

– $\mathsf{Send}_2(\Pi, I, i, j, x_i, (y_j, w_j))$: $\mathcal{S}$ computes k_i and sk_i as follows:

5. Sample $g_i \leftarrow_r \chi_\beta$ and compute $k_i = (p_j d + y_j)\hat{r}_i + 2dg_i$ where $d = H_1(j, i, y_j, x_i)$;
6. Compute $\sigma_i = \mathsf{Mod}_2(k_i, w_j)$ and derive the session key $\mathrm{sk}_i = H_2(i, j, x_i, y_j, w_j, \sigma_i)$.

– $\mathsf{SessionKeyReveal}(\mathrm{sid})$: Let $\mathrm{sid} = (\Pi, *, i, *, *, *, *)$, $\mathcal{S}$ returns sk_i if the session key of sid has been generated.
– $\mathsf{Corrupt}(i)$: Return the static secret key s_i of i to $\mathcal{A}$.
– $\mathsf{Test}(\mathrm{sid})$: Let $\mathrm{sid} = (\Pi, I, i, j, x_i, (y_j, w_j))$, $\mathcal{S}$ aborts if $(i, j) \neq (i^*, j^*)$, or x_i and y_j are not output by the s_{i^*}-th session of i^* and the s_j^*-th session of j^*, respectively. Else, $\mathcal{S}$ chooses $b \leftarrow_r \{0,1\}$, returns $sk_i' \leftarrow_r \{0,1\}^\kappa$ if $b = 0$. Otherwise, return the session key sk_i of sid.

Claim 1 *The probability that $\mathcal{S}$ will not abort in $G_{1,0}$ is at least $\frac{1}{m^2 N^2}$.*

Proof. This claim directly follows from the fact that $\mathcal{S}$ randomly chooses $i^*, j^* \leftarrow_r \{1, \ldots, N\}$ and $s_{i^*}, s_j^* \leftarrow_r \{1, \ldots, m\}$ independently from the view of $\mathcal{A}$. □

Game $G_{1,1}$ $\mathcal{S}$ behaves almost the same as in $G_{1,0}$ except in the following case:

– $\mathsf{Send}_1(\Pi, R, j, i, x_i)$: If $(i, j) \neq (i^*, j^*)$, or it is not the s_j^*-th session of j^*, $\mathcal{S}$ answers the query as in Game $G_{1,0}$. Otherwise, it proceeds as follows:

$1'$. Sample $r_j, f_j \leftarrow_r \chi_\beta$ and compute $y_j = ar_j + 2f_j$;
$2'$. Sample an invertible element $d \leftarrow_r \chi_\gamma$, compute $\hat{r}_j = s_j d + r_j$ and $\hat{f}_j = e_j d + f_j$;
$3'$. Letting $\mathbf{z} \in \mathbb{Z}^{2n}$ be the coefficient vector of $\hat{r}_j$ concatenated with the coefficient vector of $\hat{f}_j$, and $\mathbf{z}_1 \in \mathbb{Z}^{2n}$ be the coefficient vector of $s_j d$ concatenated with the coefficient vector of $e_j d$, repeat the steps $1' \sim 3'$ with probability $1 - \min\left(\frac{D_{\mathbb{Z}^{2n},\beta}(\mathbf{z})}{MD_{\mathbb{Z}^{2n},\beta,\mathbf{z}_1}(\mathbf{z})}, 1\right)$;
$4'$. Abort if there is a tuple $((j, i, y_j, x_i), *)$ in L_1. Else, add $((j, i, y_j, x_i), d)$ into L_1. Then, sample $g_j \leftarrow_r \chi_\beta$ and compute $k_j = (p_i c + x_i)\hat{r}_j + 2cg_j$ where $c = H_1(i, j, x_i)$;
$5'$. Compute $w_j = \mathsf{Cha}(k_j) \in \{0,1\}^n$ and return (y_j, w_j) to $\mathcal{A}$;
$6'$. Compute $\sigma_j = \mathsf{Mod}_2(k_j, w_j)$ and derive the session key $\mathrm{sk}_j = H_2(i, j, x_i, y_j, w_j, \sigma_j)$.

In the following, we denote $F_{1,l}$ as the event that $\mathcal{A}$ outputs a guess b' that equals to b in Game $G_{1,l}$.

Claim 2 *If RLWE$_{q,\beta}$ is hard, then* $\Pr[F_{1,l}] = \Pr[F_{1,0}] - \mathrm{negl}(\kappa)$.

Proof. Since H_1 is a random oracle, Game $G_{1,0}$ and Game $G_{1,1}$ are identical if the adversary $\mathcal{A}$ does not make a H_1 query $((j, i, y_j, x_i), *)$ before $\mathcal{S}$ generates y_j. Thus, the claim follows if the probability that $\mathcal{A}$ makes such a query in both Games is negligible. Actually, if $\mathcal{A}$ can make the query before seeing y_j with non-negligible probability, we can construct an algorithm $\mathcal{B}$ that breaks the RLWE$_{q,\beta}$ assumption.

Formally, after given a ring-LWE challenge tuple $(a, \mathbf{b}) \in R_q \times R_q^\ell$ in matrix form for some polynomially bounded ℓ, $\mathcal{B}$ sets $a = u$ and behaves like in Game $G_{1,0}$ until $\mathcal{B}$ has to generate y_j for the s_j^*-th session of j^* intended for party i^*. Instead of generating a fresh y_j, $\mathcal{B}$ simply sets y_j as the first unused elements in $\mathbf{b} = (b_0, \ldots, b_{\ell-1})$, and checks if there is a tuple $((j, i, y_j, x_i), *)$ in L_1. If yes, it returns 1 and aborts, else it returns 0 and aborts.

It is easy to check that $\mathcal{A}$ has the same view as in $G_{1,0}$ and $G_{1,1}$ until the point that $\mathcal{B}$ has to compute y_j. Moreover, if $\mathbf{b} = (b_0 = us_0 + 2x_0, \ldots, b_\ell = us_\ell + 2x_\ell)$ for some randomly choose $s, x \leftarrow_r \chi_\beta$, we have the probability that $\mathcal{A}$ will make the H_1 query with (j, i, y_j, x_i) is non-negligible by assumption. While if $\mathbf{b}$ is uniformly distributed over $\mathbb{R}_q^\ell$, we have the probability that $\mathcal{A}$ will make the H_1 query with (j, i, y_j, x_i) is negligible. This shows that $\mathcal{B}$ can be used to solve Ring-LWE assumption by interacting with $\mathcal{A}$. □

Game $G_{1,2}$ $\mathcal{S}$ behaves almost the same as in $G_{1,1}$ except in the following case:

- $\mathsf{Send}_1(\Pi, R, j, i, x_i)$: If $(i, j) \neq (i^*, j^*)$, or it is not the s_j^*-th session of j^*, $\mathcal{S}$ answers the query as in Game $G_{1,1}$. Otherwise, it proceeds as follows:
  1′. Sample an invertible element $d \leftarrow_r \chi_\gamma$, and $\mathbf{z} \leftarrow_r D_{\mathbb{Z}^{2n},\beta}$;
  2′. Interpreting $\mathbf{z}$ as two ring elements $\hat{r}_j, \hat{f}_j \in R_q$, and define $y_j = a\hat{r}_j + 2\hat{f}_j - p_j d$.
  3′. Repeat the steps $1' \sim 3'$ with probability $1 - 1/M$;
  4′. Abort if there is a tuple $((j, i, y_j, x_i), *)$ in L_1. Else, add $((j, i, y_j, x_i), d)$ into L_1. Then, sample $g_j \leftarrow_r \chi_\beta$ and compute $k_j = (p_i c + x_i)\hat{r}_j + 2cg_j$ where $c = H_1(i, j, x_i)$;
  5′. Compute $w_j = \mathsf{Cha}(k_j) \in \{0, 1\}^n$ and return (y_j, w_j) to $\mathcal{A}$;
  6′. Compute $\sigma_j = \mathsf{Mod}_2(k_j, w_j)$ and derive the session key $\mathrm{sk}_j = H_2(i, j, x_i, y_j, w_j, \sigma_j)$.

Claim 3 *If* $\beta = \omega(\alpha\gamma n\sqrt{n \log n})$, *then* $\Pr[F_{1,2}] = \Pr[F_{1,1}] - \mathrm{negl}(\kappa)$.

Proof. By Lemma 1 and Lemma 2, we have that both $\|s_j d\| \leq \alpha\gamma n\sqrt{n}$ and $\|e_j d\| \leq \alpha\gamma n\sqrt{n}$ (in Game $G_{1,1}$) hold with overwhelming probability. This means that $\beta = \omega(\alpha\gamma n\sqrt{n \log n})$ satisfies the requirement in Theorem 1, and thus the distribution of $(d, \mathbf{z})$ in Game $G_{1,2}$ is statistically close to that in $G_{1,1}$. The claims follows from the fact that the equation $y_j = a\hat{r}_j + 2\hat{f}_j - p_j d$ holds in both Game $G_{1,1}$ and $G_{1,2}$.

Game $G_{1,3}$ $\mathcal{S}$ behaves almost the same as in $G_{1,2}$, except for the following case:

- $\mathsf{Send}_0(\Pi, I, i, j)$: If $(i, j) \neq (i^*, j^*)$, or it is not the s_{i^*}-th session of i^*, $\mathcal{S}$ answers as in Game $G_{1,2}$. Otherwise, it proceeds as follows:
  1. Sample $r_i, f_i \leftarrow_r \chi_\beta$ and compute $x_i = ar_i + 2f_i$;
  2. Sample an invertible element $c \leftarrow_r \chi_\gamma$, compute $\hat{r}_i = s_i c + r_i$ and $\hat{f}_i = e_i c + f_i$;
  3. Letting $\mathbf{z} \in \mathbb{Z}^{2n}$ be the coefficient vector of $\hat{r}_i$ concatenated with the coefficient vector of $\hat{f}_i$, and $\mathbf{z}_1 \in \mathbb{Z}^{2n}$ be the coefficient vector of $s_i c$ concatenated with the coefficient vector of $e_i c$, repeat the steps $1 \sim 3$ with probability $1 - \min\left(\frac{D_{\mathbb{Z}^{2n},\beta}(\mathbf{z})}{M D_{\mathbb{Z}^{2n},\beta,\mathbf{z}_1}(\mathbf{z})}, 1\right)$.

4. $\boxed{\text{Abort if there is a tuple } ((i,j,x_i),*) \text{ in } L_1. \text{ Else, add } ((i,j,x_i),c) \text{ into } L_1}$. Return x_i to $\mathcal{A}$.

Claim 4 *If RLWE$_{q,\beta}$ is hard, then* $\Pr[F_{1,3}] = \Pr[F_{1,2}] - \mathrm{negl}(\kappa)$.

Proof. The proof is similar to the proof of Claim 2, we omit the details. □

Game $G_{1,4}$ $\mathcal{S}$ behaves almost the same as in $G_{1,3}$ except for the following case:

- $\mathsf{Send}_0(\Pi, I, i, j)$: If $(i,j) \neq (i^*, j^*)$, or it is not the s_{i^*}-th session of i^*, $\mathcal{S}$ answers as in Game $G_{1,3}$. Otherwise, it proceeds as follows:
  1. $\boxed{\text{Sample an invertible element } c \leftarrow_r \chi_\gamma, \text{ and } \mathbf{z} \leftarrow_r D_{\mathbb{Z}^{2n},\beta}}$;
  2. $\boxed{\text{Interpreting } \mathbf{z} \text{ as two ring elements } \hat{r}_i, \hat{f}_i \in R_q, \text{ and define } x_i = a\hat{r}_i + 2\hat{f}_i - p_i c}$.
  3. $\boxed{\text{Repeat the steps } 1 \sim 3 \text{ with probability } 1 - 1/M}$;
  4. Abort if there is a tuple $((i,j,x_i),*)$ in L_1. Else, add $((i,j,x_i),c)$ into L_1. Return x_i to $\mathcal{A}$.

Claim 5 *If* $\beta = \omega(\alpha\gamma n\sqrt{n \log n})$*, then* $\Pr[F_{1,4}] = \Pr[F_{1,3}] - \mathrm{negl}(\kappa)$.

Proof. The proof is similar to the proof of Claim 3, we omit the details. □

Game $G_{1,5}$ $\mathcal{S}$ behaves almost the same as in $G_{1,4}$ except for the following case:

- $\mathsf{Send}_2(\Pi, I, i, j, x_i, (y_j, w_j))$: If $(i,j) \neq (i^*, j^*)$, or it is not the s_{i^*}-th session of i^*, $\mathcal{S}$ behaves as in Game $G_{1,4}$. Otherwise, if (y_j, w_j) is output by the s_j^*-th session of party j^*, $\mathcal{S}$ sets $\boxed{\mathrm{sk}_i = \mathrm{sk}_j}$, where sk_j is the session key of $\mathrm{sid} = (\Pi, R, j, i, x_i, (y_j, w_j))$. Else, $\mathcal{S}$ samples $g_i \leftarrow_r \chi_\beta$ and computes $k_i = (p_j d + y_j)\hat{r}_i + 2dg_i$ where $d = H_1(j, i, y_j, x_i)$. Finally, it computes $\sigma_i = \mathsf{Mod}_2(k_i, w_j)$ and derive the session key $\mathrm{sk}_i = H_2(i, j, x_i, y_j, w_j, \sigma_i)$.

Claim 6 $\Pr[F_{1,5}] = \Pr[F_{1,4}] - \mathrm{negl}(\kappa)$.

Proof. This claim follows since $G_{1,5}$ is just a conceptual change of $G_{1,4}$ by the correctness of the protocol.□

Game $G_{1,6}$ $\mathcal{S}$ behaves almost the same as in $G_{1,5}$ except in the following case:

- $\mathsf{Send}_0(\Pi, I, i, j)$: If $(i,j) \neq (i^*, j^*)$, or it is not the s_{i^*}-th session of i^*, $\mathcal{S}$ answers as in Game $G_{1,5}$. Otherwise, it proceeds as follows:
  1. Sample an invertible element $c \leftarrow_r \chi_\gamma$, and $\boxed{\hat{x}_i \leftarrow_r R_q}$;
  2. $\boxed{\text{Define } x_i = \hat{x}_i - p_i c}$.
  3. Repeat the steps $1 \sim 3$ with probability $1 - 1/M$;
  4. Abort if there is a tuple $((i,j,x_i),*)$ in L_1. Else, add $((i,j,x_i),c)$ into L_1. Return x_i to $\mathcal{A}$.
- $\mathsf{Send}_2(\Pi, I, i, j, x_i, (y_j, w_j))$: If $(i,j) \neq (i^*, j^*)$, or it is not the s_{i^*}-th session of i^*, or (y_j, w_j) is output by the s_j^*-th session of party j^*, $\mathcal{S}$ behaves the same as in $G_{1,5}$. Otherwise, it proceeds as follows:
  5. $\boxed{\text{Randomly choose } k_i \leftarrow_r R_q}$;
  6. Compute $\sigma_i = \mathsf{Mod}_2(k_i, w_j)$ and derive the session key $\mathrm{sk}_i = H_2(i, j, x_i, y_j, w_j, \sigma_i)$.

Note that in Game $G_{1,6}$, we have made two changes: 1) The term $a\hat{r}_i + 2\hat{f}_i$ in Game $G_{1,5}$ is replaced by a uniformly chosen element $\hat{x} \in R_q$ at random; 2) The value $k_i = (p_j d + y_j)\hat{r}_i + 2dg_i$ in Game $G_{1,5}$ is replaced by a uniformly chosen string $k_i \leftarrow_r R_q$, when (y_j, w'_j) is output by the s^*_j-th session of party j^* but $w_j \neq w'_j$. In the following, we will employ the "deferred analysis" proof technique in [34], which informally allows us to proceed the security games by patiently postponing some tough probability analysis to a later game. Specially, for $\ell = 5, 6, 7$, denote $Q_{1,l}$ as the event in Game $G_{1,\ell}$ that 1) (y_j, w'_j) is output by the s^*_j-th session of party j^* but $w_j \neq w'_j$, and 2) $\mathcal{A}$ makes a query to H_2 that is exactly used to generate the session key sk_i for the s_{i^*}-th session of party i^*, *i.e.*, $\mathrm{sk}_i = H_1(i, j, x_i, y_j, w_j, \sigma_i)$ for $\sigma_i = \mathsf{Mod}_2(k_i, w_j)$. Ideally, if $Q_{1,5}$ does not happen, then the adversary cannot distinguish whether a correctly computed k_i or a randomly chosen one is used (since H_2 is a random oracle, and the adversary gains no information about k_i even if it obtains the session key sk_i). However, we cannot prove the claim immediately due to technical reason. Instead, we will show that $\Pr[Q_{1,5}] \approx \Pr[Q_{1,6}] \approx \Pr[Q_{1,7}]$ and $\Pr[Q_{1,7}]$ is negligible in κ.

Claim 7 *If RLWE$_{q,\beta}$ is hard,* $\Pr[Q_{1,6}] = \Pr[Q_{1,5}] - \mathrm{negl}(\kappa)$*, and* $\Pr[F_{1,6}|\neg Q_{1,6}] = \Pr[F_{1,5}|\neg Q_{1,5}] - \mathrm{negl}(\kappa)$.

Proof. Note that H_2 is a random oracle, the event $Q_{1,5}$ is independent from the distribution of the corresponding sk_i. Namely, no matter whether or not $\mathcal{A}$ obtains sk_i, $\Pr[Q_{1,5}]$ is the same, which also holds for $\Pr[Q_{1,6}]$. In addition, under the RLWE$_{q,\beta}$ assumption, we have $\hat{x}_i = a\hat{r}_i + 2\hat{f}_i$ in $G_{1,5}$ is computationally indistinguishable from uniform distribution over R_q, and thus the public information (*i.e.*, static public keys and public transcripts) in $G_{1,5}$ and $G_{1,6}$ is computationally indistinguishable. In particular, the view of the adversary $\mathcal{A}$ before $Q_{1,\ell}$ happens for $\ell = 5, 6$ is computationally indistinguishable, which implies that $\Pr[Q_{1,6}] = \Pr[Q_{1,5}] - \mathrm{negl}(\kappa)$. Besides, if $Q_{1,l}$ for $l = 5, 6$ does not happen, the distribution of sk_i is the same in both games. In other words, $\Pr[F_{1,6}|\neg Q_{1,6}] = \Pr[F_{1,5}|\neg Q_{1,5}] - \mathrm{negl}(\kappa)$. □

Game $G_{1,7}$ $\mathcal{S}$ behaves almost the same as in $G_{1,6}$ except in the following case:

- $\mathsf{Send}_1(\Pi, R, j, i, x_i)$: If $(i, j) \neq (i^*, j^*)$, or it is not the s^*_j-th session of j^*, $\mathcal{S}$ answers the query as in Game $G_{1,6}$. Otherwise, it proceeds as follows:
  1′. Sample an invertible element $d \leftarrow_r \chi_\gamma$, and $\boxed{\hat{y}_j \leftarrow_r R_q}$;
  2′. $\boxed{\text{Define } y_j = \hat{y}_j - p_j d}$.
  3′. Repeat the steps $1' \sim 3'$ with probability $1 - 1/M$;
  4′. Abort if there is a tuple $((j, i, y_j, x_i), *)$ in L_1. Else, add $((j, i, y_j, x_i), d)$ into L_1. Then, the simulator $\mathcal{S}$ $\boxed{\text{uniformly chooses } k_j \leftarrow_r R_q \text{ at random}}$;
  5′. Compute $w_j = \mathsf{Cha}(k_j) \in \{0, 1\}^n$ and return (y_j, w_j) to $\mathcal{A}$;
  6′. Compute $\sigma_j = \mathsf{Mod}_2(k_j, w_j)$ and derive the session key $\mathrm{sk}_j = H_2(i, j, x_i, y_j, w_j, \sigma_j)$.

Claim 8 *Let n be a power of 2, prime $q > 203$ satisfying $q = 1 \mod 2n$, $\beta = \omega(\alpha\gamma n\sqrt{n \log n})$. Then, if RLWE$_{q,\beta}$ is hard, Game $G_{1,6}$ and $G_{1,7}$ are computationally indistinguishable. In particular, we have* $\Pr[Q_{1,7}] = \Pr[Q_{1,6}] - \mathrm{negl}(\kappa)$*, and* $\Pr[F_{1,7}|\neg Q_{1,7}] = \Pr[F_{1,6}|\neg Q_{1,6}] - \mathrm{negl}(\kappa)$.

Proof. Assume there is an adversary that distinguishes Game $G_{1,6}$ and $G_{1,7}$, we now construct a distinguisher $\mathcal{D}$ that solves the Ring-LWE problem. Specifically, let $(\mathbf{u} = (u_0, \ldots, u_{\ell-1}), \mathbf{B}) \in R_q^\ell \times R_q^{\ell \times \ell}$ be a challenge Ring-LWE tuple in matrix form for some polynomially bounded ℓ, $\mathcal{D}$ first sets public parameter $a = u_0$. Then, it randomly chooses invertible elements $\mathbf{v} = (v_1, \ldots, v_{\ell-1}) \leftarrow \chi_\gamma^{\ell-1}$, and compute $\hat{\mathbf{u}} = (v_1 \cdot u_1, \ldots, v_{\ell-1}u_{\ell-1})$. Finally, $\mathcal{D}$ behaves the same as $\mathcal{S}$ in Game $G_{1,6}$, except for the following cases:

– $\mathsf{Send}_0(\Pi, I, i, j)$: If $(i,j) \neq (i^*, j^*)$, or it is not the s_{i^*}-th session of i^*, $\mathcal{S}$ answers as in Game $G_{1,5}$. Otherwise, it proceeds as follows:
  1. Set c and $\hat{x}_i$ be the first unused element in $\mathbf{v}$ and $\hat{\mathbf{u}}$, respectively;
  2. Define $x_i = \hat{x}_i - p_i c$.
  3. Repeat the steps $1 \sim 3$ with probability $1 - 1/M$;
  4. Abort if there is a tuple $((i, j, x_i), *)$ in L_1. Else, add $((i, j, x_i), c)$ into L_1. Return x_i to $\mathcal{A}$.
– $\mathsf{Send}_1(\Pi, R, j, i, x_i)$: If $(i,j) \neq (i^*, j^*)$, or it is not the s_j^*-th session of j^*, $\mathcal{S}$ answers the query as in Game $G_{1,6}$. Otherwise, it proceeds as follows:
  $1'$. Sample an invertible element $d \leftarrow_r \chi_\gamma$, and set $\hat{y}_j$ be the first unused element in $\mathbf{b}_0 = (b_{0,0}, \ldots, b_{0,\ell-1})$;
  $2'$. Define $y_j = \hat{y}_j - p_j d$.
  $3'$. Repeat the steps $1' \sim 3'$ with probability $1 - 1/M$;
  $4'$. Abort if there is a tuple $((j, i, y_j, x_i), *)$ in L_1. Else, add $((j, i, y_j, x_i), d)$ into L_1. Then, let $\ell_1 \geq 1$ be the index that $\hat{x}_i$ appears in $\hat{\mathbf{u}}$, and $\ell_2 \geq 0$ be the index that $\hat{y}_j$ appears in $\mathbf{b}_0$, the simulator $\mathcal{S}$ sets $k_j = cb_{\ell_1,\ell_2}$;
  $5'$. Compute $w_j = \mathsf{Cha}(k_j) \in \{0,1\}^n$ and return (y_j, w_j) to $\mathcal{A}$;
  $6'$. Compute $\sigma_j = \mathsf{Mod}_2(k_j, w_j)$ and derive the session key $\mathrm{sk}_j = H_2(i, j, x_i, y_j, w_j, \sigma_j)$.

Since $\mathbf{v}$ is randomly and independently chosen from $\chi_\gamma^{\ell-1}$, the distribution of c is identical to Game $G_{1,6}$ and $G_{1,7}$. Besides, since each v_i is invertible in R_q, we have $\hat{\mathbf{u}}$ is uniformly distributed over $R_q^{\ell-1}$, which shows that the distribution of $\hat{x}_i$ is identical to Game $G_{1,6}$ and $G_{1,7}$ except with negligible probability. Moreover, if $(\mathbf{u}, \mathbf{B}) \in R_q^\ell \times R_q^{\ell\times\ell}$ is a Ring-LWE challenge tuple in matrix form, we have $\hat{y}_j = u_0 s_{\ell_2} + 2e_{0,\ell_2}$ and $k_j = cb_{\ell_1,\ell_2} = cu_{\ell_1} s_{\ell_2} + 2ce_{\ell_1,\ell_2} = \hat{x}_i s_{\ell_2} + 2ce_{\ell_1,\ell_2} = (x_i + p_i c)s_{\ell_2} + 2ce_{\ell_1,\ell_2}$ for some randomly chosen $s_{\ell_2}, e_{0,\ell_2}, e_{\ell_1,\ell_2} \leftarrow_r \chi_\beta$. This shows that the view of $\mathcal{A}$ is the same as in Game $G_{1,6}$. While if $(\mathbf{u}, \mathbf{B}) \in R_q^\ell \times R_q^{\ell\times\ell}$ is uniformly distributed over $R_q^\ell \times R_q^{\ell\times\ell}$, we have both $\hat{y}_j$ and $k_j = cb_{\ell_1,\ell_2}$ are uniformly distributed over R_q (since c is invertible). Thus, the view of $\mathcal{A}$ is the same as in $G_{1,7}$. In all, we have shown that $\mathcal{D}$ can be used to break Ring-LWE assumption if $\mathcal{A}$ can distinguish Game $G_{1,6}$ and $G_{1,7}$. □

Claim 9 *If* $0.97n > 2\kappa$*, we have* $\Pr[Q_{1,7}] = \mathrm{negl}(\kappa)$

Proof. Let $k_{i,\ell}$ be the element "computed" by $\mathcal{S}$ for the s_i^*-th session at party i^* in Games $G_{1,\ell}$, and $k_{j,\ell}$ be the element "computed" by $\mathcal{S}$ for the s_j^*-th session at party j^*. By the correctness of the protocol, we have that $k_{i,5} = k_{j,5} + \hat{g}$ for some $\hat{g}$ with small coefficients in $G_{1,5}$. Since we have proven that the view of the adversary before $Q_{1,\ell}$ happens in Game $G_{1,5}$, $G_{1,6}$ and $G_{1,7}$ is computationally indistinguishable, the equation $k_{i,7} = k_{j,7} + \hat{g}'$ should still holds for some $\hat{g}'$ with small coefficients in the adversary's view until $Q_{1,7}$ happens in $G_{1,7}$. Let (y_j, w_j) be output by the s_j^*-th session of party $j = j^*$, and (y_j, w_j') be the message that is used to complete the test session (*i.e.*, the s_{i^*}-th session of party $i = i^*$). Note that $k_{j,7}$ is randomly chosen from R_q, and the adversary can only obtain the information of $k_{j,7}$ from the public w_j, the dependence of $\hat{g}$ on k_j should be totally determined by the information of w_j. Thus, we have that $\sigma_i' = \mathsf{Mod}_2(k_i, w_j') = \mathsf{Mod}_2(k_j + \hat{g}', w_j')$ conditioned on w_j has high min-entropy by Lemma 4. In other words, the probability that the adversary makes a query $H_2(i, j, x_i, y_j, w_j', \sigma_i')$ is at most $2^{-0.97n} + \mathrm{negl}(\kappa)$, which is negligible in κ. □

Claim 10 $\Pr[F_{1,7} | \neg Q_{1,7}] = 1/2 + \mathrm{negl}(\kappa)$

Proof. Let (y_j, w_j) be output by the s_j^*-th session of party $j = j^*$, (y_j, w_j') be the message that is used to complete the test session (*i.e.*, the s_{i^*}-th session of party $i = i^*$). We distinguish the following two cases:

– $w_j = w'_j$: In this case, we have $\mathrm{sk}_i = \mathrm{sk}_j = H_2(i, j, x_i, y_j, w_j, \sigma_i)$, where $\sigma_i = \sigma_j = \mathsf{Mod}_2(k_j, w_j)$. Note that in $G_{1,7}$, k_j is randomly chosen from the uniform distribution over R_q, we have that $\sigma_j \in \{0,1\}^n$ (conditioned on w_j) has min-entropy at least $0.97n$ by Lemma 4. Thus, the probability that $\mathcal{A}$ has made a H_2 query with σ_i is less than $2^{-0.97n} + \mathrm{negl}(\kappa)$.
– $w_j \neq w'_j$: By assumption that $Q_{1,7}$ does not happen, we have $\mathcal{A}$ will never make a H_2 query with σ_i.

In all, the probability that $\mathcal{A}$ has made a H_2 query with σ_i is negligible. This claim follows from the fact that if the adversary doesn't make a query with σ_i exactly, the distribution of sk_i is uniform over $\{0,1\}^k$ due to the random oracle property of H_2, *i.e.*, $\Pr[F_{1,7}|\neg Q_{1,7}] = 1/2 + \mathrm{negl}(\kappa)$. □

Combining the claims 1∼10, we have that Lemma 5 follows.

4.2 Type II Adversary

In this subsection, we prove that our AKE is secure against any PPT **Type II** adversary $\mathcal{A}$.

Lemma 6. *Let n be a power of 2 satisfying $0.97n \geq 2\kappa$, prime $q > 203$ satisfying $q = 1 \mod 2n$, $\beta = \omega(\alpha\gamma n\sqrt{n \log n})$. Then, if RLWE$_{q,\alpha}$ is hard, the proposed AKE is secure against any PPT **Type II** adversary $\mathcal{A}$ in the random oracle model.*

Proof. We prove this lemma via a sequence of games $G_{2,l}$ for $0 \leq l \leq 6$.

Game $G_{2,0}$. $\mathcal{S}$ chooses $i^*, j^* \leftarrow_r \{1, \ldots, N\}$ and $s_{i^*} \leftarrow_r \{1, \ldots, m\}$, and hopes that the adversary will choose $\mathrm{sid}^* = (\Pi, I, i^*, j^*, x_{i^*}, (y_{j^*}, w_{j^*}))$ as the test session, where x_{i^*} is output by the s_{i^*}-th session of party i^* with intended party j^* (note that sid^* has no matching session for **Type II** adversary). Then, $\mathcal{S}$ chooses $a \leftarrow_r R_q$, honestly generates static public keys for all parties (by randomly choosing s_i and e_i from χ_α), and simulates the attack environment for $\mathcal{A}$. Specifically, $\mathcal{S}$ maintains two tables L_1, L_2 for the random oracles H_1, H_2, respectively, and answers the queries from $\mathcal{A}$ as follows:

– $H_1(in)$: If there doesn't exist a tuple (in, out) in the L_1 list, choose an element $out \leftarrow_r \chi_\gamma$, and add (in, out) to the L_1 list. Then, return out to $\mathcal{A}$.
– $H_2(in)$ queries: If there doesn't exist a tuple (in, out) in the L_2 list, choose an element $out \leftarrow_r \{0,1\}^k$, and add (in, out) to the L_2 list. Then, return out to $\mathcal{A}$.
– $\mathsf{Send}_0(\Pi, I, i, j)$: $\mathcal{A}$ activates a new session of i with intended party j, $\mathcal{S}$ proceeds as follows:
  1. Sample $r_i, f_i \leftarrow_r \chi_\beta$ and compute $x_i = ar_i + 2f_i$;
  2. Compute $c = H_1(i, j, x_i)$, $\hat{r}_i = s_i c + r_i$ and $\hat{f}_i = e_i c + f_i$;
  3. Letting $\mathbf{z} \in \mathbb{Z}^{2n}$ be the coefficient vector of $\hat{r}_i$ concatenated with the coefficient vector of $\hat{f}_i$, and $\mathbf{z}_1 \in \mathbb{Z}^{2n}$ be the coefficient vector of $s_i c$ concatenated with the coefficient vector of $e_i c$, repeat the steps $1 \sim 3$ with probability $1 - \min\left(\frac{D_{\mathbb{Z}^{2n},\beta}(\mathbf{z})}{MD_{\mathbb{Z}^{2n},\beta,\mathbf{z}_1}(\mathbf{z})}, 1\right)$;
  4. Return x_i to $\mathcal{A}$.
– $\mathsf{Send}_1(\Pi, R, j, i, x_i)$: $\mathcal{S}$ proceeds as follows:
  1′. Sample $r_j, f_j \leftarrow_r \chi_\beta$ and compute $y_j = ar_j + 2f_j$;
  2′. Compute $d = H_1(j, i, y_j, x_i)$, $\hat{r}_j = s_j d + r_j$ and $\hat{f}_j = e_j d + f_j$;
  3′. Letting $\mathbf{z} \in \mathbb{Z}^{2n}$ be the coefficient vector of $\hat{r}_j$ concatenated with the coefficient vector of $\hat{f}_j$, and $\mathbf{z}_1 \in \mathbb{Z}^{2n}$ be the coefficient vector of $s_j d$ concatenated with the coefficient vector of $e_j d$, repeat the steps $1' \sim 3'$ with probability $1 - \min\left(\frac{D_{\mathbb{Z}^{2n},\beta}(\mathbf{z})}{MD_{\mathbb{Z}^{2n},\beta,\mathbf{z}_1}(\mathbf{z})}, 1\right)$;

4′. Sample $g_j \leftarrow_r \chi_\beta$ and compute $k_j = (p_i c + x_i)\hat{r}_j + 2cg_j$ where $c = H_1(i, j, x_i)$;
5′. Compute $w_j = \mathsf{Cha}(k_j) \in \{0,1\}^n$ and return (y_j, w_j) to $\mathcal{A}$;
6′. Compute $\sigma_j = \mathsf{Mod}_2(k_j, w_j)$ and derive the session key $\mathrm{sk}_j = H_2(i, j, x_i, y_j, w_j, \sigma_j)$.

– $\mathsf{Send}_2(\Pi, I, i, j, x_i, (y_j, w_j))$: $\mathcal{S}$ computes k_i and sk_i as follows:
5. Sample $g_i \leftarrow_r \chi_\beta$ and compute $k_i = (p_j d + y_j)\hat{r}_i + 2dg_i$ where $d = H_1(j, i, y_j, x_i)$;
6. Compute $\sigma_i = \mathsf{Mod}_2(k_i, w_j)$ and derive the session key $\mathrm{sk}_i = H_2(i, j, x_i, y_j, w_j, \sigma_i)$.

– $\mathsf{SessionKeyReveal}(\mathrm{sid})$: Let $\mathrm{sid} = (\Pi, *, i, *, *, *, *)$, $\mathcal{S}$ returns sk_i if the session key of sid has been generated.
– $\mathsf{Corrupt}(i)$: Return the static secret key s_i of i to $\mathcal{A}$.
– $\mathsf{Test}(\mathrm{sid})$: Let $\mathrm{sid} = (\Pi, I, i, j, x_i, (y_j, w_j))$, if $(i, j) \neq (i^*, j^*)$, or x_i and y_j are not output by the s_{i^*}-th session of i^* and the s_j^*-th session of j^*, respectively, $\mathcal{S}$ aborts. Otherwise, $\mathcal{S}$ chooses $b \leftarrow_r \{0,1\}$ and $sk_i' \leftarrow_r \{0,1\}^k$. If $b = 0$, $\mathcal{S}$ returns sk_i', else it returns the real session sk_i of sid.

Claim 11 *The probability that $\mathcal{S}$ will not abort in $G_{2,0}$ is at least $\frac{1}{mN^2}$.*

Proof. This claim directly follows from the fact that $\mathcal{S}$ randomly chooses $i^*, j^* \leftarrow_r \{1, \ldots, N\}$ and $s_{i^*} \leftarrow_r \{1, \ldots, m\}$ without $\mathcal{A}$ knowing it. □

Game $G_{2,1}$. $\mathcal{S}$ behaves almost the same as in $G_{2,0}$, except in the following cases:

– $\mathsf{Send}_0(\Pi, I, i, j)$: If $i \neq j^*$, $\mathcal{S}$ answers the query as in Game $G_{2,0}$. Otherwise, it proceeds as follows:
1. Sample an invertible element $c \leftarrow_r \chi_\gamma$, and $\mathbf{z} \leftarrow_r D_{\mathbb{Z}^{2n},\beta}$;
2. Interpreting $\mathbf{z}$ as two ring elements $\hat{r}_i, \hat{f}_i \in R_q$, and define $x_i = a\hat{r}_i + 2\hat{f}_i - p_i c$.
3. Repeat the steps $1 \sim 3$ with probability $1 - 1/M$;
4. Abort if there is a tuple $((i, j, x_i), *)$ in L_1. Else, add $((i, j, x_i), c)$ into L_1. Return x_i to $\mathcal{A}$.

– $\mathsf{Send}_1(\Pi, R, j, i, x_i)$: If $j \neq j^*$, $\mathcal{S}$ answers the query as in Game $G_{2,0}$. Otherwise, it proceeds as follows:
1′. Sample an invertible element $d \leftarrow_r \chi_\gamma$, and $\mathbf{z} \leftarrow_r D_{\mathbb{Z}^{2n},\beta}$;
2′. Interpreting $\mathbf{z}$ as two ring elements $\hat{r}_j, \hat{f}_j \in R_q$, and define $y_j = a\hat{r}_j + 2\hat{f}_j - p_j d$.
3′. Repeat the steps $1' \sim 3'$ with probability $1 - 1/M$;
4′. Abort if there is a tuple $((j, i, y_j, x_i), *)$ in L_1. Else, add $((j, i, y_j, x_i), d)$ into L_1. Then, sample $g_j \leftarrow_r \chi_\beta$ and compute $k_j = (p_i c + x_i)\hat{r}_j + 2cg_j$ where $c = H_1(i, j, x_i)$;
5′. Compute $w_j = \mathsf{Cha}(k_j) \in \{0,1\}^n$ and return (y_j, w_j) to $\mathcal{A}$;
6′. Compute $\sigma_j = \mathsf{Mod}_2(k_j, w_j)$ and derive the session key $\mathrm{sk}_j = H_2(i, j, x_i, y_j, w_j, \sigma_j)$.

In the following, let $F_{2,l}$ denote the event that $\mathcal{A}$ outputs a guess b' that equals to b in Game $G_{2,l}$.

Claim 12 *If $\beta = \omega(\alpha\gamma n\sqrt{n \log n})$ and RLWE$_{q,\beta}$ is hard, then $\Pr[F_{2,1}] = \Pr[F_{2,0}] - \mathrm{negl}(\kappa)$.*

Proof. This claim can be proven via a sequence of games similar to that from $G_{1,0}$ to $G_{1,4}$, we omit the details. □

Game $G_{2,2}$. $\mathcal{S}$ behaves almost the same as in $G_{2,1}$, except it replaces the public key for party j^* with a uniformly chosen $\boxed{p_{j^*} \leftarrow_r R_q}$.

Claim 13 *If RLWE$_{q,\alpha}$ is hard, then* $\Pr[F_{2,2}] = \Pr[F_{2,1}] - \mathrm{negl}(\kappa)$.

Proof. Since the only difference between $G_{2,1}$ and $G_{2,2}$ is that $\mathcal{S}$ replaces $p_{j^*} = as_{j^*} + 2e_{j^*}$ with a randomly chosen element in R_q, an adversary that can distinguish the difference between $G_{2,1}$ and $G_{2,2}$ could be directly used to solve the RLWE$_{q,\alpha}$ problem. □

Game $G_{2,3}$. $\mathcal{S}$ behaves almost the same as in $G_{2,2}$, except in the following cases:

- $\mathsf{Send}_0(\Pi, I, i, j)$: If $(i, j) \neq (i^*, j^*)$, or it is not the s_{i^*}-th session of i^*, $\mathcal{S}$ answers the query as in Game $G_{2,2}$. Otherwise, it proceeds as follows:
  1. $\boxed{\text{Sample an invertible element } c \leftarrow_r \chi_\gamma, \text{ and } \mathbf{z} \leftarrow_r D_{\mathbb{Z}^{2n},\beta}}$;
  2. $\boxed{\text{Interpreting } \mathbf{z} \text{ as two ring elements } \hat{r}_i, \hat{f}_i \in R_q, \text{ and define } x_i = a\hat{r}_i + 2\hat{f}_i - p_i c}$.
  3. $\boxed{\text{Repeat the steps } 1 \sim 3 \text{ with probability } 1 - 1/M}$;
  4. $\boxed{\text{Abort if there is a tuple } ((i, j, x_i), *) \text{ in } L_1. \text{ Else, add } ((i, j, x_i), c) \text{ into } L_1}$. Return x_i to $\mathcal{A}$.

Claim 14 *If* $\beta = \omega(\alpha\gamma n\sqrt{n \log n})$ *and RLWE$_{q,\beta}$ is hard, then* $\Pr[F_{2,3}] = \Pr[F_{2,2}] - \mathrm{negl}(\kappa)$.

Proof. This claim can be proven via a sequence of games similar to that from $G_{1,2}$ to $G_{1,4}$, we omit the details. □

Game $G_{2,4}$. $\mathcal{S}$ first randomly chooses $u \leftarrow_r R_q$, then computes $\mathbf{v}_0 = (v_{0,1}, \ldots, v_{0,\ell-1}) \in R_q^\ell$ and $\mathbf{v}_1 = (v_{1,1}, \ldots, v_{1,\ell-1}) \in R_q^\ell$ where $v_{0,\ell'} = a\hat{r}_{\ell'} + 2\hat{f}_{\ell'}, v_{1,\ell'} = u\hat{r}_{\ell'} + 2g_{\ell'}$ for $\hat{r}_{\ell'}, \hat{f}_{\ell_1}, g_{\ell'} \leftarrow \chi_\beta$. Then, it $\boxed{\text{sets } p_{j^*} = u}$, and behaves almost the same as in $G_{2,3}$, except in the following cases:

- $\mathsf{Send}_0(\Pi, I, i, j)$: If $(i, j) \neq (i^*, j^*)$, or it is not the s_{i^*}-th session of i^*, $\mathcal{S}$ answers the query as in Game $G_{2,3}$. Otherwise, it proceeds as follows:
  1. Sample an invertible element $c \leftarrow_r \chi_\gamma$, and $\boxed{\text{set } \hat{x}_i \text{ be the first unused element in } \mathbf{v}_0}$;
  2. $\boxed{\text{Define } x_i = \hat{x}_i - p_i c}$.
  3. Repeat the steps $1 \sim 3$ with probability $1 - 1/M$;
  4. Abort if there is a tuple $((i, j, x_i), *)$ in L_1. Else, add $((i, j, x_i), c)$ into L_1. Return x_i to $\mathcal{A}$.
- $\mathsf{Send}_2(\Pi, I, i, j, x_i, (y_j, w_j))$: If $(i, j) \neq (i^*, j^*)$, or it is not the s_{i^*}-th session of i^*, $\mathcal{S}$ answers the query as in Game $G_{2,3}$. Otherwise, it proceeds as follows::
  5. Let ℓ^* be the index that $\hat{x}_i$ appears in $\mathbf{v}_0$, namely, $\hat{x}_i = a\hat{r}_{\ell^*} + 2\hat{f}_{\ell^*}$, the simulator $\mathcal{S}$ sets $d = H_1(j, i, y_j, x_i)$, and computes $\boxed{k_i = dv_{1,\ell^*} + y_j\hat{r}_{\ell^*} = d(u\hat{r}_{\ell^*} + 2g_{\ell^*}) + y_j\hat{r}_{\ell^*} = (p_j d + y_j)\hat{r}_{\ell^*} + 2dg_{\ell^*}}$;
  6. Compute $\sigma_i = \mathsf{Mod}_2(k_i, w_j)$ and derive the session key $\mathrm{sk}_i = H_2(i, j, x_i, y_j, w_j, \sigma_i)$.

Claim 15 *Game* $G_{2,4}$ *is identical to Game* $G_{2,3}$. *In particular, we have* $\Pr[F_{2,4}] = \Pr[F_{2,3}]$.

Proof. This claim simply from the fact that Game $G_{2,4}$ is just a conceptual change of Game $G_{2,3}$. □

Game $G_{2,5}$. $\mathcal{S}$ chooses$u_0, u_1 \leftarrow_r R_q$, and $\boxed{\mathbf{v}_0 = (v_{0,1}, \ldots, v_{0,\ell-1}), \mathbf{v}_1 = (v_{1,1}, \ldots, v_{1,\ell-1}) \leftarrow_r R_q^{\ell}}$. Then, it sets $a = u_0$ and $p_{j^*} = u_1$, and behaves almost the same as in $G_{2,4}$, except in the following case:

– $\mathsf{Send}_2(\Pi, I, i, j, x_i, (y_j, w_j))$: If $(i, j) \neq (i^*, j^*)$, or it is not the s_{i^*}-th session of i^*, $\mathcal{S}$ answers the query as in Game $G_{2,3}$. Otherwise, it proceeds as follows:
  5. $\boxed{\text{Randomly choose } k_i \leftarrow_r R_q}$;
  6. Compute $\sigma_i = \mathsf{Mod}_2(k_i, w_j)$ and derive the session key $\mathrm{sk}_i = H_2(i, j, x_i, y_j, w_j, \sigma_i)$.

Claim 16 *Let n be a power of 2 satisfying $0.97n \geq 2\kappa$, prime $q > 203$ satisfying $q = 1 \mod 2n$, $\beta = \omega(\alpha\gamma n\sqrt{n \log n})$. Then, if RLWE$_{q,\beta}$ is hard, Game $G_{2,5}$ is computationally indistinguishable from $G_{2,4}$. In particular, $\Pr[F_{2,4}] = \Pr[F_{2,5}] + \mathrm{negl}(\kappa) = 1/2 + \mathrm{negl}(\kappa)$.*

Proof. Note that before generating k_i, the only difference between $G_{2,4}$ and $G_{2,5}$ is that $\mathcal{S}$ replaces the Ring-LWE tuple $((u_0, u_1), (\mathbf{v}_0, \mathbf{v}_1)) \in R_q^2 \times R_q^{2\times\ell}$ with randomly chosen elements in $R_q^2 \times R_q^{2\times\ell}$. In other words, under the Ring-LWE assumption, we have that the adversary's views before generating k_i in both $G_{2,4}$ and $G_{2,5}$ are computationally indistinguishable. Besides, since H_2 is a random oracle, no matter whether $\mathcal{A}$ obtains sk_i or not, the value of sk_i will not affect the view of the adversary, especially about the knowledge of k_i.

Now, let $k_i = dv_{1,\ell^*} + y_j r_{\ell^*}$ be the element computed by using $d = H_1(j^*, i^*, y_j, x_i)$ in Game $G_{2,4}$ with non-negligible probability δ. Fixing $\mathbf{v}_0$ and $\mathbf{v}_1$ (note that all those values are chosen by $\mathcal{S}$, and are independent from the adversary's behaviors), $\mathcal{S}$ reprograms the hash query $H_1(j^*, i^*, y_j, x_i) = \tilde{d} \neq d$ by using another randomly chosen $\tilde{d} \leftarrow_r \chi_\gamma$ (recall that y_j is not generated by the simulator $\mathcal{S}$, such a H_1 query is made by the adversary $\mathcal{A}$). According to the forking lemma [6], $\mathcal{A}$ will use the same y_j to complete the test session with probability at least $\delta(\delta/q_h - 2^{-n})$, where q_h is maximum number of H_1 queries. In such a case, we have $\tilde{k}_i = \tilde{d}v_{1,\ell^*} + y_j\hat{r}_{\ell^*}$ and $\tilde{k}_i = k_i + (\tilde{d} - d)v_{1,\ell^*}$.

Denote $\hat{k}_i'$ and k_i' (*i.e.*, the values determined before and after $\mathcal{S}$ reprograms the H_1 query) as the target values in the $\mathcal{A}$'s view in Game $G_{2,5}$, we have that $\tilde{k}_i' = k_i' + (\tilde{d} - d)v_{1,\ell^*}$ should still hold in the adversary's view. Considering the fact that v_{1,ℓ^*} is uniformly and randomly chosen from R_q and $(\tilde{d} - d)$ is invertible with overwhelming probability for appropriate choice of γ and q by Lemma 10 in [65], we have that the adversary essentially has no knowledge of $\tilde{k}_i'$ and k_i'. Therefore, the choice of $k_j \leftarrow_r R_q$ in Game $G_{2,5}$ is correctly distributed in the adversary's view, and thus will not help the adversary $\mathcal{A}$ to distinguish $G_{2,5}$ from $G_{2,4}$. In other words, we have that $\Pr[F_{2,4}] = \Pr[F_{2,5}] + \mathrm{negl}(\kappa)$.

Finally, since k_i is randomly and uniformly chosen from R_q in Game $G_{2,5}$, and is independent from $\mathcal{A}$'s view, we have $\sigma_i = \mathbf{Mod}_2(k_i, w_j) \in \{0,1\}^n$ has very high min-entropy by Lemma 4. In other words, the probability that the adversary makes a query $H_2(i, j, x_i, y_j, w_j, \sigma_i)$ is at most $2^{-0.97n} + \mathrm{negl}(\kappa)$, which is negligible in κ. Since H_2 is a random oracle, we have $\Pr[F_{2,5}] = 1/2 + \mathrm{negl}(\kappa)$, which completes the proof. □

4.3 Type III Adversary

In this subsection, we prove that our AKE is secure against any PPT **Type III** adversary $\mathcal{A}$.

Lemma 7. *Let n be a power of 2 satisfying $0.97n \geq 2\kappa$, prime $q > 203$ satisfying $q = 1 \mod 2n$, $\beta = \omega(\alpha\gamma n\sqrt{n \log n})$. Then, if RLWE$_{q,\alpha}$ is hard, the proposed AKE is secure against any PPT **Type III** adversary $\mathcal{A}$ in the random oracle model.*

Proof. We prove this lemma via a sequence of games $G_{3,l}$ for $0 \leq l \leq 6$.

Game $G_{3,0}$. $\mathcal{S}$ chooses $i^*, j^* \leftarrow_r \{1, \ldots, N\}$ and $s_{j^*} \leftarrow_r \{1, \ldots, m\}$, and hopes that the adversary will choose $\mathrm{sid}^* = (\Pi, R, j^*, i^*, x_{i^*}, (y_{j^*}, w_{j^*}))$ as the test session, where (y_{j^*}, w_{j^*}) is output by the s_j^*-th session of party j^* activated by a $\mathsf{Send}_0(\Pi, R, j^*, i^*, x_{i^*})$ for some x_{i^*}. Then, $\mathcal{S}$ chooses $a \leftarrow_r R_q$, honestly generates static public keys for all parties (by randomly choosing s_i and e_i from χ_α), and simulates the attack environment for $\mathcal{A}$. Specifically, $\mathcal{S}$ maintains two tables L_1, L_2 for the random oracles H_1, H_2, respectively, and answers the queries from $\mathcal{A}$ as follows:

- $H_1(in)$: If there doesn't exist a tuple (in, out) in the L_1 list, choose an element $out \leftarrow_r \chi_\gamma$, and add (in, out) to the L_1 list. Then, return out to $\mathcal{A}$.
- $H_2(in)$ queries: If there doesn't exist a tuple (in, out) in the L_2 list, choose an element $out \leftarrow_r \{0,1\}^k$, and add (in, out) to the L_2 list. Then, return out to $\mathcal{A}$.
- $\mathsf{Send}_0(\Pi, I, i, j)$: $\mathcal{A}$ activates a new session of i with intended party j, $\mathcal{S}$ proceeds as follows:
  1. Sample $r_i, f_i \leftarrow_r \chi_\beta$ and compute $x_i = ar_i + 2f_i$;
  2. Compute $c = H_1(i, j, x_i)$, $\hat{r}_i = s_i c + r_i$ and $\hat{f}_i = e_i c + f_i$;
  3. Letting $\mathbf{z} \in \mathbb{Z}^{2n}$ be the coefficient vector of $\hat{r}_i$ concatenated with the coefficient vector of $\hat{f}_i$, and $\mathbf{z}_1 \in \mathbb{Z}^{2n}$ be the coefficient vector of $s_i c$ concatenated with the coefficient vector of $e_i c$, repeat the steps $1 \sim 3$ with probability $1 - \min\left(\frac{D_{\mathbb{Z}^{2n},\beta}(\mathbf{z})}{MD_{\mathbb{Z}^{2n},\beta,\mathbf{z}_1}(\mathbf{z})}, 1\right)$.
  4. Return x_i to $\mathcal{A}$.
- $\mathsf{Send}_1(\Pi, R, j, i, x_i)$: $\mathcal{S}$ proceeds as follows:
  $1'$. Sample $r_j, f_j \leftarrow_r \chi_\beta$ and compute $y_j = ar_j + 2f_j$;
  $2'$. Compute $d = H_1(j, i, y_j, x_i)$, $\hat{r}_j = s_j d + r_j$ and $\hat{f}_j = e_j d + f_j$;
  $3'$. Letting $\mathbf{z} \in \mathbb{Z}^{2n}$ be the coefficient vector of $\hat{r}_j$ concatenated with the coefficient vector of $\hat{f}_j$, and $\mathbf{z}_1 \in \mathbb{Z}^{2n}$ be the coefficient vector of $s_j d$ concatenated with the coefficient vector of $e_j d$, repeat the steps $1' \sim 3'$ with probability $1 - \min\left(\frac{D_{\mathbb{Z}^{2n},\beta}(\mathbf{z})}{MD_{\mathbb{Z}^{2n},\beta,\mathbf{z}_1}(\mathbf{z})}, 1\right)$;
  $4'$. Sample $g_j \leftarrow_r \chi_\beta$ and compute $k_j = (p_i c + x_i)\hat{r}_j + 2cg_j$ where $c = H_1(i, j, x_i)$;
  $5'$. Compute $w_j = \mathsf{Cha}(k_j) \in \{0,1\}^n$ and return (y_j, w_j) to $\mathcal{A}$;
  $6'$. Compute $\sigma_j = \mathsf{Mod}_2(k_j, w_j)$ and derive the session key $\mathrm{sk}_j = H_2(i, j, x_i, y_j, w_j, \sigma_j)$.
- $\mathsf{Send}_2(\Pi, I, i, j, x_i, (y_j, w_j))$: $\mathcal{S}$ computes k_i and sk_i as follows:
  5. Sample $g_i \leftarrow_r \chi_\beta$ and compute $k_i = (p_j d + y_j)\hat{r}_i + 2dg_i$ where $d = H_1(j, i, y_j, x_i)$;
  6. Compute $\sigma_i = \mathsf{Mod}_2(k_i, w_j)$ and derive the session key $\mathrm{sk}_i = H_2(i, j, x_i, y_j, w_j, \sigma_i)$.
- $\mathsf{SessionKeyReveal}(\mathrm{sid})$: Let $\mathrm{sid} = (\Pi, *, i, *, *, *, *)$, $\mathcal{S}$ returns sk_i if the session key of sid has been generated.
- $\mathsf{Corrupt}(i)$: Return the static secret key s_i of i to $\mathcal{A}$.
- $\mathsf{Test}(\mathrm{sid})$: Let $\mathrm{sid} = (\Pi, I, i, j, x_i, (y_j, w_j))$, if $(i, j) \neq (i^*, j^*)$, or x_i and y_j are not output by the s_{i^*}-th session of i^* and the s_j^*-th session of j^*, respectively, $\mathcal{S}$ aborts. Otherwise, $\mathcal{S}$ chooses $b \leftarrow_r \{0,1\}$ and $sk_i' \leftarrow_r \{0,1\}^k$. If $b = 0$, $\mathcal{S}$ returns sk_i', else it returns the real session sk_i of sid.

Claim 17 *The probability that $\mathcal{S}$ will not abort in $G_{3,0}$ with probability at least $\frac{1}{mN^2}$.*

Proof. This claim directly follows from the fact that $\mathcal{S}$ randomly chooses $i^*, j^* \leftarrow_r \{1, \ldots, N\}$ and $s_j^* \leftarrow_r \{1, \ldots, m\}$ independently from the view of $\mathcal{A}$. □

Game $G_{3,1}$. $\mathcal{S}$ behaves almost the same as in $G_{3,0}$, except in the following cases:

- $\mathsf{Send}_0(\Pi, I, i, j)$: If $i \neq i^*$, $\mathcal{S}$ answers the query as in Game $G_{2,0}$. Otherwise, it proceeds as follows:
  1. Sample an invertible element $c \leftarrow_r \chi_\gamma$, and $\mathbf{z} \leftarrow_r D_{\mathbb{Z}^{2n},\beta}$;
  2. Interpreting $\mathbf{z}$ as two ring elements $\hat{r}_i, \hat{f}_i \in R_q$, and define $x_i = a\hat{r}_i + 2\hat{f}_i - p_i c$.
  3. Repeat the steps $1 \sim 3$ with probability $1 - 1/M$;
  4. Abort if there is a tuple $((i, j, x_i), *)$ in L_1. Else, add $((i, j, x_i), c)$ into L_1. Return x_i to $\mathcal{A}$.
- $\mathsf{Send}_1(\Pi, R, j, i, x_i)$: If $j \neq i^*$, $\mathcal{S}$ answers the query as in Game $G_{2,0}$. Otherwise, it proceeds as follows:
  $1'$. Sample an invertible element $d \leftarrow_r \chi_\gamma$, and $\mathbf{z} \leftarrow_r D_{\mathbb{Z}^{2n},\beta}$;
  $2'$. Interpreting $\mathbf{z}$ as two ring elements $\hat{r}_j, \hat{f}_j \in R_q$, and define $y_j = a\hat{r}_j + 2\hat{f}_j - p_j d$.
  $3'$. Repeat the steps $1' \sim 3'$ with probability $1 - 1/M$;
  $4'$. Abort if there is a tuple $((j, i, y_j, x_i), *)$ in L_1. Else, add $((j, i, y_j, x_i), d)$ into L_1. Then, sample $g_j \leftarrow_r \chi_\beta$ and compute $k_j = (p_i c + x_i)\hat{r}_j + 2cg_j$ where $c = H_1(i, j, x_i)$;
  $5'$. Compute $w_j = \mathsf{Cha}(k_j) \in \{0, 1\}^n$ and return (y_j, w_j) to $\mathcal{A}$;
  $6'$. Compute $\sigma_j = \mathsf{Mod}_2(k_j, w_j)$ and derive the session key $\mathrm{sk}_j = H_2(i, j, x_i, y_j, w_j, \sigma_j)$.

  In the following, we use $F_{3,l}$ to denote the event that $\mathcal{A}$ outputs a guess b' that equals to b in Game $G_{3,l}$.

Claim 18 *If* $\beta = \omega(\alpha\gamma n\sqrt{n \log n})$ *and RLWE*$_{q,\beta}$ *is hard, then* $\Pr[F_{3,1}] = \Pr[F_{3,0}] - \mathrm{negl}(\kappa)$.

Proof. This claim can be proven via a sequence of games similar to that from $G_{1,0}$ to $G_{1,4}$, we omit the details. □

Game $G_{3,2}$. $\mathcal{S}$ behaves almost the same as in $G_{3,1}$, except it replaces the public key for party i^* with a randomly chosen $p_{i^*} \leftarrow_r R_q$.

Claim 19 *If RLWE*$_{q,\alpha}$ *is hard, then* $\Pr[F_{3,2}] = \Pr[F_{3,1}] - \mathrm{negl}(\kappa)$.

Proof. Since the only difference between $G_{3,1}$ and $G_{3,2}$ is that $\mathcal{S}$ replaces $p_{i^*} = as_{i^*} + 2e_{i^*}$ with a randomly chosen element in R_q, an adversary that can distinguish the difference between $G_{3,1}$ and $G_{3,2}$ could be directly used to solve the RLWE$_{q,\alpha}$ problem. □

Game $G_{3,3}$. $\mathcal{S}$ behaves almost the same as in $G_{3,2}$, except in the following case:

- $\mathsf{Send}_1(\Pi, R, j, i, x_i)$: If $(i, j) \neq (i^*, j^*)$, or it is not the s_j^*-th session of j^*, $\mathcal{S}$ answers the query as in Game $G_{3,2}$. Otherwise, it proceeds as follows:
  $1'$. Sample an invertible element $d \leftarrow_r \chi_\gamma$, and $\mathbf{z} \leftarrow_r D_{\mathbb{Z}^{2n},\beta}$;
  $2'$. Interpreting $\mathbf{z}$ as two ring elements $\hat{r}_j, \hat{f}_j \in R_q$, and define $y_j = a\hat{r}_j + 2\hat{f}_j - p_j d$.
  $3'$. Repeat the steps $1' \sim 3'$ with probability $1 - 1/M$;
  $4'$. Abort if there is a tuple $((j, i, y_j, x_i), *)$ in L_1. Else, add $((j, i, y_j, x_i), d)$ into L_1. Then, sample $g_j \leftarrow_r \chi_\beta$ and compute $k_j = (p_i c + x_i)\hat{r}_j + 2cg_j$ where $c = H_1(i, j, x_i)$;
  $5'$. Compute $w_j = \mathsf{Cha}(k_j) \in \{0, 1\}^n$ and return (y_j, w_j) to $\mathcal{A}$;
  $6'$. Compute $\sigma_j = \mathsf{Mod}_2(k_j, w_j)$ and derive the session key $\mathrm{sk}_j = H_2(i, j, x_i, y_j, w_j, \sigma_j)$.

Claim 20 *If* $\beta = \omega(\alpha\gamma n\sqrt{n \log n})$ *and RLWE*$_{q,\beta}$ *is hard, then* $\Pr[F_{3,3}] = \Pr[F_{3,2}] - \mathrm{negl}(\kappa)$.

Proof. This claim can be proven via a sequence of games similar to that from $G_{1,0}$ to $G_{1,2}$, we omit the details. □

Game $G_{3,4}$. $\mathcal{S}$ first randomly chooses $u \leftarrow_r R_q$, then computes $\mathbf{v}_0 = (v_{0,1}, \ldots, v_{0,\ell-1}) \in R_q^\ell$ and $\mathbf{v}_1 = (v_{1,1}, \ldots, v_{1,\ell-1}) \in R_q^\ell$ where $v_{0,\ell'} = a\hat{r}_{\ell'} + 2\hat{f}_{\ell'}, v_{1,\ell'} = u\hat{r}_{\ell'} + 2g_{\ell'}$ for $\hat{r}_{\ell'}, \hat{f}_{\ell_1}, g_{\ell'} \leftarrow \chi_\beta$. Then, it $\boxed{\text{sets } p_{i^*} = u}$, and behaves almost the same as in $G_{3,3}$, except in the following case:

- $\mathsf{Send}_1(\Pi, R, j, i, x_i)$: If $(i, j) \neq (i^*, j^*)$, or it is not the s_j^*-th session of j^*, $\mathcal{S}$ answers the query as in Game $G_{3,2}$. Otherwise, it proceeds as follows:
  1′. Sample an invertible element $d \leftarrow_r \chi_\gamma$, and $\boxed{\text{set } \hat{y}_j \text{ be the first unused element in } \mathbf{v}_0}$;
  2′. $\boxed{\text{Define } y_j = \hat{y}_j - p_j d}$;
  3′. Repeat the steps $1' \sim 3'$ with probability $1 - 1/M$;
  4′. Abort if there is a tuple $((j, i, y_j, x_i), *)$ in L_1. Else, add $((j, i, y_j, x_i), d)$ into L_1. Then, the simulator $\mathcal{S}$ sets $\boxed{k_j = cv_{1,\ell^*} + x_i\hat{r}_{\ell^*} = c(u\hat{r}_{\ell^*} + 2g_{\ell^*}) + x_i\hat{r}_{\ell^*} = (p_i c + x_i)\hat{r}_{\ell^*} + 2cg_{\ell^*}}$, where ℓ^* is the index that $\hat{y}_j$ appears in $\mathbf{v}_0$ (*i.e.*, $\hat{y}_j = a\hat{r}_{\ell^*} + 2\hat{f}_{\ell^*}$) and $c = H_1(i, j, x_i)$;
  5′. Compute $w_j = \mathsf{Cha}(k_j) \in \{0, 1\}^n$ and return (y_j, w_j) to $\mathcal{A}$;
  6′. Compute $\sigma_j = \mathsf{Mod}_2(k_j, w_j)$ and derive the session key $\mathrm{sk}_j = H_2(i, j, x_i, y_j, w_j, \sigma_j)$.

Claim 21 *Game* $G_{3,4}$ *is identical to Game* $G_{3,3}$. *In particular, we have* $\Pr[F_{3,4}] = \Pr[F_{3,3}]$.

Proof. This claim simply from the fact that Game $G_{3,4}$ is just a conceptual change of Game $G_{3,3}$. □

Game $G_{3,5}$. $\mathcal{S}$ chooses $u_0, u_1 \leftarrow_r R_q$, and $\boxed{\mathbf{v}_0 = (v_{0,1}, \ldots, v_{0,\ell-1}), \mathbf{v}_1 = (v_{1,1}, \ldots, v_{1,\ell-1}) \leftarrow_r R_q^\ell}$. Then, it sets $a = u_0$ and $p_{j^*} = u_1$, and behaves almost the same as in $G_{3,4}$ except the following cases:

- $\mathsf{Send}_1(\Pi, R, j, i, x_i)$: If $(i, j) \neq (i^*, j^*)$, or it is not the s_j^*-th session of j^*, $\mathcal{S}$ answers the query as in Game $G_{3,2}$. Otherwise, it proceeds as follows:
  1′. Sample an invertible element $d \leftarrow_r \chi_\gamma$, and set $\hat{y}_j$ be the first unused element in $\mathbf{v}_0$;
  2′. Define $y_j = \hat{y}_j - p_j d$;
  3′. Repeat the steps $1' \sim 3'$ with probability $1 - 1/M$;
  4′. Abort if there is a tuple $((j, i, y_j, x_i), *)$ in L_1. Else, add $((j, i, y_j, x_i), d)$ into L_1. Then, $\mathcal{S}$ randomly chooses $\boxed{k_j \leftarrow_r R_q}$;
  5′. Compute $w_j = \mathsf{Cha}(k_j) \in \{0, 1\}^n$ and return (y_j, w_j) to $\mathcal{A}$;
  6′. Compute $\sigma_j = \mathsf{Mod}_2(k_j, w_j)$ and derive the session key $\mathrm{sk}_j = H_2(i, j, x_i, y_j, w_j, \sigma_j)$.

Claim 22 *Let* n *be a power of* 2 *satisfying* $0.97n \geq 2\kappa$, *prime* $q > 203$ *satisfying* $q = 1 \mod 2n$, $\beta = \omega(\alpha\gamma n\sqrt{n \log n})$. *Then, if RLWE*$_{q,\beta}$ *is hard, Game* $G_{3,5}$ *is computationally indistinguishable from* $G_{3,4}$. *In particular,* $\Pr[F_{3,4}] = \Pr[F_{3,5}] + \mathrm{negl}(\kappa) = 1/2 + \mathrm{negl}(\kappa)$.

Proof. Note that before generating k_j, the only difference between $G_{3,4}$ and $G_{3,5}$ is that $\mathcal{S}$ replaces the Ring-LWE tuple $((u_0, u_1), (\mathbf{v}_0, \mathbf{v}_1)) \in R_q^2 \times R_q^{2\times\ell}$ with randomly chosen elements in $R_q^2 \times R_q^{2\times\ell}$. In other words, under the Ring-LWE assumption, we have that the adversary's views before generating k_j in both $G_{3,4}$ and $G_{3,5}$ are computationally indistinguishable. Besides, since H_2 is a random oracle, no matter whether $\mathcal{A}$ obtains sk_j or not, the value of sk_j will not affect the view of the adversary, especially about the knowledge of k_j.

Now, let $k_j = cv_{1,\ell^*} + x_i r_{\ell^*}$ be the element computed by using $c = H_1(i^*, j^*, x_i)$ in Game $G_{3,4}$ with non-negligible probability δ. Fixing $\mathbf{v}_0$ and $\mathbf{v}_1$ (note that all those values are chosen by $\mathcal{S}$, and are independent from the adversary's behaviors), $\mathcal{S}$ reprograms the hash query $H_1(i^*, j^*, x_i) = \tilde{c} \neq c$ by using

another randomly chosen $\tilde{c} \leftarrow_r \chi_\gamma$ (recall that x_i is not generated by the simulator $\mathcal{S}$, such a H_1 query is made by the adversary $\mathcal{A}$). According to the forking lemma [6], $\mathcal{A}$ will use the same x_i to complete the test session with probability at least $\delta(\delta/q_h - 2^{-n})$, where q_h is maximum number of H_1 queries. In such a case, we have $\tilde{k}_j = \tilde{c}v_{1,\ell^*} + x_i\hat{r}_{\ell^*}$ and $\tilde{k}_j = k_j + (\tilde{c} - c)v_{1,\ell^*}$.

Denote $\tilde{k}'_j$ and k'_j (*i.e.*, the values determined before and after $\mathcal{S}$ reprograms the H_1 query) as the target values in the $\mathcal{A}$'s view in Game $G_{3,5}$, we have that $\tilde{k}'_j = k'_j + (\tilde{c} - c)v_{1,\ell^*}$ should still hold in the adversary's view before seeing corresponding w_j's. Considering the fact that v_{1,ℓ^*} is uniformly and randomly chosen from R_q and $(\tilde{c} - c)$ is invertible with overwhelming probability for appropriate choice of γ and q by Lemma 10 in [65], we have that the adversary essentially has no knowledge of $\tilde{k}'_j$ and k'_j before seeing w_j's. In other words, before seeing w_j, the distribution of k_j in the adversary's view should be uniformly distributed over R_q. Therefore, the choice of $k_j \leftarrow_r R_q$ in Game $G_{3,5}$ is correctly distributed in the adversary's view, and thus the adversary should not distinguish the difference between $G_{3,5}$ and $G_{3,4}$ without the knowledge of w_j's. Note that the w_j's in both $G_{3,5}$ and $G_{3,4}$ are computed from k_j following the protocol, the knowledge of w_j will not help the adversary $\mathcal{A}$ to distinguish $G_{3,5}$ from $G_{3,4}$. In other words, we have that $\Pr[F_{3,4}] = \Pr[F_{3,5}] + \mathrm{negl}(\kappa)$.

Finally, since k_j is randomly and uniformly chosen from R_q in Game $G_{3,5}$, and is independent from $\mathcal{A}$'s view, we have $\sigma_j = \mathbf{Mod}_2(k_j, w_j) \in \{0,1\}^n$ has very high min-entropy by Lemma 4. In other words, the probability that the adversary makes a query $H_2(i, j, x_i, y_j, w_j, \sigma_j)$ is at most $2^{-0.97n} + \mathrm{negl}(\kappa)$, which is negligible in κ. Since H_2 is a random oracle, we have $\Pr[F_{3,5}] = 1/2 + \mathrm{negl}(\kappa)$, which completes the proof. □

4.4 Type IV Adversary

In this subsection, we prove that our AKE is secure against any PPT **Type IV** adversary $\mathcal{A}$.

Lemma 8. *Let n be a power of 2 satisfying $0.97n \geq 2\kappa$, prime $q > 203$ satisfying $q = 1 \mod 2n$, $\beta = \omega(\alpha\gamma n\sqrt{n \log n})$. Then, if RLWE$_{q,\alpha}$ is hard, the proposed AKE is secure against any PPT **Type IV** adversary $\mathcal{A}$ in the random oracle model.*

Proof. We prove this lemma via a sequence of games $G_{4,l}$ for $0 \leq l \leq 4$.

Game $G_{4,0}$. $\mathcal{S}$ first chooses $i^*, j^* \leftarrow_r \{1, \ldots, N\}$ and $s_{i^*}, s_{j^*} \leftarrow_r \{1, \ldots, m\}$, and hopes that the adversary will choose $\mathrm{sid}^* = (\Pi, R, j^*, i^*, x_{i^*}, (y_{j^*}, w_{j^*}))$ as the test session, where x_{i^*} is output by the s_{i^*}-th session of party i^*, and (y_{j^*}, w_{j^*}) is output by the s^*_j-th session of party j^* activated by a $\mathsf{Send}_1(\Pi, R, j^*, i^*, x_{i^*})$. Then, $\mathcal{S}$ chooses $a \leftarrow_r R_q$, honestly generates static public keys for all parties (by randomly choosing s_i and e_i from χ_α), and simulates the attack environment for $\mathcal{A}$. Specifically, $\mathcal{S}$ maintains two tables L_1, L_2 for the random oracles H_1, H_2, respectively, and answers the queries from $\mathcal{A}$ as follows:

- $H_1(in)$: If there doesn't exist a tuple (in, out) in the L_1 list, choose an element $out \leftarrow_r \chi_\gamma$, and add (in, out) to the L_1 list. Then, return out to $\mathcal{A}$.
- $H_2(in)$ queries: If there doesn't exist a tuple (in, out) in the L_2 list, choose an element $out \leftarrow_r \{0,1\}^k$, and add (in, out) to the L_2 list. Then, return out to $\mathcal{A}$.
- $\mathsf{Send}_0(\Pi, I, i, j)$: $\mathcal{A}$ activates a new session of i with intended party j, $\mathcal{S}$ proceeds as follows:
  1. Sample $r_i, f_i \leftarrow_r \chi_\beta$ and compute $x_i = ar_i + 2f_i$;
  2. Compute $c = H_1(i, j, x_i)$, $\hat{r}_i = s_i c + r_i$ and $\hat{f}_i = e_i c + f_i$;

3. Letting $\mathbf{z} \in \mathbb{Z}^{2n}$ be the coefficient vector of $\hat{r}_i$ concatenated with the coefficient vector of $\hat{f}_i$, and $\mathbf{z}_1 \in \mathbb{Z}^{2n}$ be the coefficient vector of $s_i c$ concatenated with the coefficient vector of $e_i c$, repeat the steps $1 \sim 3$ with probability $1 - \min\left(\frac{D_{\mathbb{Z}^{2n},\beta}(\mathbf{z})}{MD_{\mathbb{Z}^{2n},\beta,\mathbf{z}_1}(\mathbf{z})}, 1\right)$.
4. Return x_i to $\mathcal{A}$.

– $\mathsf{Send}_1(\Pi, R, j, i, x_i)$: $\mathcal{S}$ proceeds as follows:

$1'$. Sample $r_j, f_j \leftarrow_r \chi_\beta$ and compute $y_j = ar_j + 2f_j$;
$2'$. Compute $d = H_1(j, i, y_j, x_i)$, $\hat{r}_j = s_j d + r_j$ and $\hat{f}_j = e_j d + f_j$;
$3'$. Letting $\mathbf{z} \in \mathbb{Z}^{2n}$ be the coefficient vector of $\hat{r}_j$ concatenated with the coefficient vector of $\hat{f}_j$, and $\mathbf{z}_1 \in \mathbb{Z}^{2n}$ be the coefficient vector of $s_j d$ concatenated with the coefficient vector of $e_j d$, repeat the steps $1' \sim 3'$ with probability $1 - \min\left(\frac{D_{\mathbb{Z}^{2n},\beta}(\mathbf{z})}{MD_{\mathbb{Z}^{2n},\beta,\mathbf{z}_1}(\mathbf{z})}, 1\right)$;
$4'$. Sample $g_j \leftarrow_r \chi_\beta$ and compute $k_j = (p_i c + x_i)\hat{r}_j + 2cg_j$ where $c = H_1(i, j, x_i)$;
$5'$. Compute $w_j = \mathsf{Cha}(k_j) \in \{0,1\}^n$ and return (y_j, w_j) to $\mathcal{A}$;
$6'$. Compute $\sigma_j = \mathsf{Mod}_2(k_j, w_j)$ and derive the session key $\mathrm{sk}_j = H_2(i, j, x_i, y_j, w_j, \sigma_j)$.

– $\mathsf{Send}_2(\Pi, I, i, j, x_i, (y_j, w_j))$: $\mathcal{S}$ computes k_i and sk_i as follows:

5. Sample $g_i \leftarrow_r \chi_\beta$ and compute $k_i = (p_j d + y_j)\hat{r}_i + 2dg_i$ where $d = H_1(j, i, y_j, x_i)$;
6. Compute $\sigma_i = \mathsf{Mod}_2(k_i, w_j)$ and derive the session key $\mathrm{sk}_i = H_2(i, j, x_i, y_j, w_j, \sigma_i)$.

– $\mathsf{SessionKeyReveal}(\mathrm{sid})$: Let $\mathrm{sid} = (\Pi, *, i, *, *, *, *)$, $\mathcal{S}$ returns sk_i if the session key of sid has been generated.
– $\mathsf{Corrupt}(i)$: Return the static secret key s_i of i to $\mathcal{A}$.
– $\mathsf{Test}(\mathrm{sid})$: Let $\mathrm{sid} = (\Pi, I, i, j, x_i, (y_j, w_j))$, if $(i, j) \neq (i^*, j^*)$, or x_i and y_j are not output by the s_{i^*}-th session of i^* and the s_j^*-th session of j^*, respectively, $\mathcal{S}$ aborts. Otherwise, $\mathcal{S}$ chooses $b \leftarrow_r \{0,1\}$ and $sk_i' \leftarrow_r \{0,1\}^k$. If $b = 0$, $\mathcal{S}$ returns sk_i', else it returns the real session sk_i of sid.

Claim 23 *The probability that $\mathcal{S}$ will not abort in $G_{4,0}$ is at least $\frac{1}{m^2 N^2}$.*

Proof. This claim directly follows from the fact that $\mathcal{S}$ randomly chooses $i^*, j^* \leftarrow_r \{1, \ldots, N\}$ and $s_{i^*}, s_j^* \leftarrow_r \{1, \ldots, m\}$ independently from the view of $\mathcal{A}$. □

Game $G_{4,1}$. $\mathcal{S}$ behaves almost the same as in $G_{4,0}$, except in the following case:

– $\mathsf{Send}_1(\Pi, R, j, i, x_i)$: If $(i, j) \neq (i^*, j^*)$, or it is not the s_j^*-th session of j^*, $\mathcal{S}$ answers the query as in Game $G_{4,0}$. Otherwise, it proceeds as follows:

$1'$. Sample an invertible element $d \leftarrow_r \chi_\gamma$, and $\mathbf{z} \leftarrow_r D_{\mathbb{Z}^{2n},\beta}$;
$2'$. Interpreting $\mathbf{z}$ as two ring elements $\hat{r}_j, \hat{f}_j \in R_q$, and define $y_j = a\hat{r}_j + 2\hat{f}_j - p_j d$.
$3'$. Repeat the steps $1' \sim 3'$ with probability $1 - 1/M$;
$4'$. Abort if there is a tuple $((j, i, y_j, x_i), *)$ in L_1. Else, add $((j, i, y_j, x_i), d)$ into L_1. Then, sample $g_j \leftarrow_r \chi_\beta$ and compute $k_j = (p_i c + x_i)\hat{r}_j + 2cg_j$ where $c = H_1(i, j, x_i)$;
$5'$. Compute $w_j = \mathsf{Cha}(k_j) \in \{0,1\}^n$ and return (y_j, w_j) to $\mathcal{A}$;
$6'$. Compute $\sigma_j = \mathsf{Mod}_2(k_j, w_j)$ and derive the session key $\mathrm{sk}_j = H_2(i, j, x_i, y_j, w_j, \sigma_j)$.

In the following, we define $F_{4,l}$ as the event that $\mathcal{A}$ outputs a guess b' that equals to b in Game $G_{4,l}$.

Claim 24 *If $\beta = \omega(\alpha\gamma n\sqrt{n \log n})$ and RLWE$_{q,\beta}$ is hard, then* $\Pr[F_{4,1}] = \Pr[F_{4,0}] - \mathrm{negl}(\kappa)$.

Proof. This claim can be proven via a sequence of games similar to that from $G_{1,0}$ to $G_{1,2}$, we omit the details. □

Game $G_{4,2}$. $\mathcal{S}$ behaves almost the same as in $G_{4,1}$, except for the following cases:

- $\mathsf{Send}_0(\Pi, I, i, j)$: If $(i, j) \neq (i^*, j^*)$, or it is not the s_{i^*}-th session of i^*, $\mathcal{S}$ answers the query as in Game $G_{4,1}$. Otherwise, it proceeds as follows:
  1. Sample an invertible element $c \leftarrow_r \chi_\gamma$, and $\mathbf{z} \leftarrow_r D_{\mathbb{Z}^{2n},\beta}$;
  2. Interpreting $\mathbf{z}$ as two ring elements $\hat{r}_i, \hat{f}_i \in R_q$, and define $x_i = a\hat{r}_i + 2\hat{f}_i - p_i c$.
  3. Repeat the steps $1 \sim 3$ with probability $1 - 1/M$;
  4. Abort if there is a tuple $((i, j, x_i), *)$ in L_1. Else, add $((i, j, x_i), c)$ into L_1. Return x_i to $\mathcal{A}$.
- $\mathsf{Send}_2(\Pi, I, i, j, x_i, (y_j, w_j))$: If $(i, j) \neq (i^*, j^*)$, or it is not the s_{i^*}-th session of i^*, $\mathcal{S}$ answers the query as in Game $G_{4,1}$. Otherwise, if (y_j, w_j) is output by the s_j^*-th session of party j^*, let sk_j be the session key of session $\mathrm{sid} = (\Pi, R, j, i, x_i, (y_j, w_j))$, $\mathcal{S}$ sets $\mathrm{sk}_i = \mathrm{sk}_j$. Else, $\mathcal{S}$ samples $g_i \leftarrow_r \chi_\beta$ and computes $k_i = (p_j d + y_j)\hat{r}_i + 2dg_i$ where $d = H_1(j, i, y_j, x_i)$. Finally, it computes $\sigma_i = \mathsf{Mod}_2(k_i, w_j)$ and derive the session key $\mathrm{sk}_i = H_2(i, j, x_i, y_j, w_j, \sigma_i)$.

Claim 25 *If* $\beta = \omega(\alpha\gamma n\sqrt{n \log n})$ *and RLWE*$_{q,\beta}$ *is hard, then* $\Pr[F_{4,2}] = \Pr[F_{4,1}] - \mathrm{negl}(\kappa)$.

Proof. This claim can be proven via a sequence of games similar to that from $G_{1,3}$ to $G_{1,5}$, we omit the details. □

Game $G_{4,3}$ $\mathcal{S}$ behaves almost the same as in $G_{4,2}$ except in the following case:

- $\mathsf{Send}_0(\Pi, I, i, j)$: If $(i, j) \neq (i^*, j^*)$, or it is not the s_{i^*}-th session of i^*, $\mathcal{S}$ answers as in Game $G_{4,2}$. Otherwise, it proceeds as follows:
  1. Sample an invertible element $c \leftarrow_r \chi_\gamma$, and $\hat{x}_i \leftarrow_r R_q$;
  2. Define $x_i = \hat{x}_i - p_i c$.
  3. Repeat the steps $1 \sim 3$ with probability $1 - 1/M$;
  4. Abort if there is a tuple $((i, j, x_i), *)$ in L_1. Else, add $((i, j, x_i), c)$ into L_1. Return x_i to $\mathcal{A}$.
- $\mathsf{Send}_2(\Pi, I, i, j, x_i, (y_j, w_j))$: If $(i, j) \neq (i^*, j^*)$, or it is not the s_{i^*}-th session of i^*, or (y_j, w_j) is output by the s_j^*-th session of party j^*, $\mathcal{S}$ behaves the same as in $G_{4,2}$. Otherwise, it proceeds as follows:
  5. Randomly choose $k_i \leftarrow_r R_q$;
  6. Compute $\sigma_i = \mathsf{Mod}_2(k_i, w_j)$ and derive the session key $\mathrm{sk}_i = H_2(i, j, x_i, y_j, w_j, \sigma_i)$.

Denote $Q_{4,l}$ be event that in Game $G_{4,l}$ for $l = 2, 3, 4$, $\mathcal{A}$ makes a H_2 query with σ_i for the s_{i^*}-th session of party i^*, when (y_j, w'_j) is output by the s_j^*-th session of party j^* but $w_j \neq w'_j$.

Claim 26 *If RLWE*$_{q,\beta}$ *is hard,* $\Pr[Q_{4,3}] = \Pr[Q_{4,2}] - \mathrm{negl}(\kappa)$, *and* $\Pr[F_{4,3}|\neg Q_{4,3}] = \Pr[F_{4,2}|\neg Q_{4,2}] - \mathrm{negl}(\kappa)$.

Proof. The proof is similar to the proof of Claim 7, we omit the details. □

Game $G_{4,4}$ $\mathcal{S}$ behaves almost the same as in $G_{4,3}$ except in the following case:

- $\mathsf{Send}_1(\Pi, R, j, i, x_i)$: If $(i, j) \neq (i^*, j^*)$, or it is not the s_j^*-th session of j^*, $\mathcal{S}$ answers the query as in Game $G_{4,3}$. Otherwise, it proceeds as follows:
  1′. Sample an invertible element $d \leftarrow_r \chi_\gamma$, and $\hat{y}_j \leftarrow_r R_q$;

$2'$. $\boxed{\text{Define } y_j = \hat{y}_j - p_j d}$.
$3'$. Repeat the steps $1' \sim 3'$ with probability $1 - 1/M$;
$4'$. Abort if there is a tuple $((j, i, y_j, x_i), *)$ in L_1. Else, add $((j, i, y_j, x_i), d)$ into L_1. Then, the simulator $\mathcal{S}$ $\boxed{\text{uniformly chooses } k_j \leftarrow_r R_q \text{ at random}}$;
$5'$. Compute $w_j = \mathsf{Cha}(k_j) \in \{0,1\}^n$ and return (y_j, w_j) to $\mathcal{A}$;
$6'$. Compute $\sigma_j = \mathsf{Mod}_2(k_j, w_j)$ and derive the session key $\mathrm{sk}_j = H_2(i, j, x_i, y_j, w_j, \sigma_j)$.

Claim 27 *Let n be a power of 2 satisfying $0.97n \geq 2\kappa$, prime $q > 203$ satisfying $q = 1 \mod 2n$, $\beta = \omega(\alpha\gamma n\sqrt{n \log n})$. Then, if RLWE$_{q,\alpha}$ is hard, Game $G_{4,3}$ and $G_{4,4}$ are computationally indistinguishable. In particular, we have $\Pr[Q_{4,4}] = \Pr[Q_{4,3}]$, and $\Pr[F_{4,4}|\neg Q_{4,4}] = \Pr[F_{4,3}|\neg Q_{4,3}] - \mathrm{negl}(\kappa)$.*

Proof. The proof is similar to the proof of Claim 8, we omit the details. □

Claim 28 *If $0.97n > 2\kappa$, we have $\Pr[Q_{4,4}] = \mathrm{negl}(\kappa)$.*

Proof. The proof is similar to the proof of Claim 9, we omit the details. □

Claim 29 $\Pr[F_{4,4}|\neg Q_{4,4}] = 1/2 + \mathrm{negl}(\kappa)$.

Proof. The proof is similar to the proof of Claim 10, we omit the details. □
Combining the claims 23∼29, we have that Lemma 8 follows. □

4.5 Type V Adversary

In this subsection, we prove that our AKE is secure against any PPT **Type V** adversary $\mathcal{A}$.

Lemma 9. *Let n be a power of 2 satisfying $0.97n \geq 2\kappa$, prime $q > 203$ satisfying $q = 1 \mod 2n$, $\beta = \omega(\alpha\gamma n\sqrt{n \log n})$. Then, if RLWE$_{q,\alpha}$ is hard, the proposed AKE is secure against any PPT **Type V** adversary $\mathcal{A}$ in the random oracle model.*

Proof. We prove this lemma via a sequence of games $G_{5,l}$ for $0 \leq l \leq 4$.

Game $G_{5,0}$. $\mathcal{S}$ chooses $i^*, j^* \leftarrow_r \{1, \ldots, N\}$ and $s_{i^*}, s_{j^*} \leftarrow_r \{1, \ldots, m\}$, and hopes that the adversary will choose $\mathrm{sid}^* = (\Pi, R, j^*, i^*, x_{i^*}, (y_{j^*}, w_{j^*}))$ as the test session, where x_{i^*} is output by the s_{i^*}-th session of party i^*, and (y_{j*}, w_{j^*}) is output by the s_j^*-th session of party j^* activated by a $\mathsf{Send}_1(\Pi, R, j^*, i^*, x_{i^*})$. Then, $\mathcal{S}$ chooses $a \leftarrow_r R_q$, honestly generates static public keys for all parties (by randomly choosing s_i and e_i from χ_α), and simulates the attack environment for $\mathcal{A}$. Specifically, $\mathcal{S}$ maintains two tables L_1, L_2 for the random oracles H_1, H_2, respectively, and answers the queries from $\mathcal{A}$ as follows:

- $H_1(in)$: If there doesn't exist a tuple (in, out) in the L_1 list, choose an element $out \leftarrow_r \chi_\gamma$, and add (in, out) to the L_1 list. Then, return out to $\mathcal{A}$.
- $H_2(in)$ queries: If there doesn't exist a tuple (in, out) in the L_2 list, choose an element $out \leftarrow_r \{0,1\}^k$, and add (in, out) to the L_2 list. Then, return out to $\mathcal{A}$.
- $\mathsf{Send}_0(\Pi, I, i, j)$: $\mathcal{A}$ activates a new session of i with intended party j, $\mathcal{S}$ proceeds as follows:
  1. Sample $r_i, f_i \leftarrow_r \chi_\beta$ and compute $x_i = ar_i + 2f_i$;
  2. Compute $c = H_1(i, j, x_i)$, $\hat{r}_i = s_i c + r_i$ and $\hat{f}_i = e_i c + f_i$;

3. Letting $\mathbf{z} \in \mathbb{Z}^{2n}$ be the coefficient vector of $\hat{r}_i$ concatenated with the coefficient vector of $\hat{f}_i$, and $\mathbf{z}_1 \in \mathbb{Z}^{2n}$ be the coefficient vector of $s_i c$ concatenated with the coefficient vector of $e_i c$, repeat the steps $1 \sim 3$ with probability $1 - \min\left(\frac{D_{\mathbb{Z}^{2n},\beta}(\mathbf{z})}{MD_{\mathbb{Z}^{2n},\beta,\mathbf{z}_1}(\mathbf{z})}, 1\right)$;
4. Return x_i to $\mathcal{A}$.

– $\mathsf{Send}_1(\Pi, R, j, i, x_i)$: $\mathcal{S}$ proceeds as follows:

$1'$. Sample $r_j, f_j \leftarrow_r \chi_\beta$ and compute $y_j = ar_j + 2f_j$;
$2'$. Compute $d = H_1(j, i, y_j, x_i)$, $\hat{r}_j = s_j d + r_j$ and $\hat{f}_j = e_j d + f_j$;
$3'$. Letting $\mathbf{z} \in \mathbb{Z}^{2n}$ be the coefficient vector of $\hat{r}_j$ concatenated with the coefficient vector of $\hat{f}_j$, and $\mathbf{z}_1 \in \mathbb{Z}^{2n}$ be the coefficient vector of $s_j d$ concatenated with the coefficient vector of $e_j d$, repeat the steps $1' \sim 3'$ with probability $1 - \min\left(\frac{D_{\mathbb{Z}^{2n},\beta}(\mathbf{z})}{MD_{\mathbb{Z}^{2n},\beta,\mathbf{z}_1}(\mathbf{z})}, 1\right)$;
$4'$. Sample $g_j \leftarrow_r \chi_\beta$ and compute $k_j = (p_i c + x_i)\hat{r}_j + 2cg_j$ where $c = H_1(i, j, x_i)$;
$5'$. Compute $w_j = \mathsf{Cha}(k_j) \in \{0,1\}^n$ and return (y_j, w_j) to $\mathcal{A}$;
$6'$. Compute $\sigma_j = \mathsf{Mod}_2(k_j, w_j)$ and derive the session key $\mathrm{sk}_j = H_2(i, j, x_i, y_j, w_j, \sigma_j)$.

– $\mathsf{Send}_2(\Pi, I, i, j, x_i, (y_j, w_j))$: $\mathcal{S}$ computes k_i and sk_i as follows:

5. Sample $g_i \leftarrow_r \chi_\beta$ and compute $k_i = (p_j d + y_j)\hat{r}_i + 2dg_i$ where $d = H_1(j, i, y_j, x_i)$;
6. Compute $\sigma_i = \mathsf{Mod}_2(k_i, w_j)$ and derive the session key $\mathrm{sk}_i = H_2(i, j, x_i, y_j, w_j, \sigma_i)$.

– $\mathsf{SessionKeyReveal}(\mathrm{sid})$: Let $\mathrm{sid} = (\Pi, *, i, *, *, *, *)$, $\mathcal{S}$ returns sk_i if the session key of sid has been generated.
– $\mathsf{Corrupt}(i)$: Return the static secret key s_i of i to $\mathcal{A}$.
– $\mathsf{Test}(\mathrm{sid})$: Let $\mathrm{sid} = (\Pi, I, i, j, x_i, (y_j, w_j))$, if $(i, j) \neq (i^*, j^*)$, or x_i and y_j are not output by the s_{i^*}-th session of i^* and the s_j^*-th session of j^*, respectively, $\mathcal{S}$ aborts. Otherwise, $\mathcal{S}$ chooses $b \leftarrow_r \{0,1\}$ and $sk_i' \leftarrow_r \{0,1\}^k$. If $b = 0$, $\mathcal{S}$ returns sk_i', else it returns the real session sk_i of sid.

Claim 30 *The probability that $\mathcal{S}$ will not abort in $G_{5,0}$ with probability at least $\frac{1}{m^2N^2}$.*

Proof. This claim directly follows from the fact that $\mathcal{S}$ randomly chooses $i^*, j^* \leftarrow_r \{1, \ldots, N\}$ and $s_{i^*}, s_j^* \leftarrow_r \{1, \ldots, m\}$ independently from the view of $\mathcal{A}$. □

Game $G_{5,1}$. $\mathcal{S}$ behaves almost the same as in $G_{5,0}$, except in the following case:

– $\mathsf{Send}_1(\Pi, R, j, i, x_i)$: If $(i, j) \neq (i^*, j^*)$, or it is not the s_j^*-th session of j^*, $\mathcal{S}$ answers the query as in Game $G_{5,0}$. Otherwise, it proceeds as follows:

$1'$. Sample an invertible element $d \leftarrow_r \chi_\gamma$, and $\mathbf{z} \leftarrow_r D_{\mathbb{Z}^{2n},\beta}$;
$2'$. Interpreting $\mathbf{z}$ as two ring elements $\hat{r}_j, \hat{f}_j \in R_q$, and define $y_j = a\hat{r}_j + 2\hat{f}_j - p_j d$.
$3'$. Repeat the steps $1' \sim 3'$ with probability $1 - 1/M$;
$4'$. Abort if there is a tuple $((j, i, y_j, x_i), *)$ in L_1. Else, add $((j, i, y_j, x_i), d)$ into L_1. Then, sample $g_j \leftarrow_r \chi_\beta$ and compute $k_j = (p_i c + x_i)\hat{r}_j + 2cg_j$ where $c = H_1(i, j, x_i)$;
$5'$. Compute $w_j = \mathsf{Cha}(k_j) \in \{0,1\}^n$ and return (y_j, w_j) to $\mathcal{A}$;
$6'$. Compute $\sigma_j = \mathsf{Mod}_2(k_j, w_j)$ and derive the session key $\mathrm{sk}_j = H_2(i, j, x_i, y_j, w_j, \sigma_j)$.

In the following, let $F_{5,l}$ denote the event that $\mathcal{A}$ outputs a guess b' that equals to b in Game $G_{5,l}$.

Claim 31 *If $\beta = \omega(\alpha\gamma n\sqrt{n \log n})$ and RLWE$_{q,\beta}$ is hard, then $\Pr[F_{5,l}] = \Pr[F_{5,0}] - \mathrm{negl}(\kappa)$.*

Proof. This claim can be proven via a sequence of games similar to that from $G_{1,0}$ to $G_{1,2}$, we omit the details. □

Game $G_{5,2}$. $\mathcal{S}$ behaves almost the same as in $G_{5,1}$, except for the following cases:

- $\mathsf{Send}_0(\Pi, I, i, j)$: If $(i, j) \neq (i^*, j^*)$, or it is not the s_{i^*}-th session of i^*, $\mathcal{S}$ answers the query as in Game $G_{5,1}$. Otherwise, it proceeds as follows:
  1. Sample an invertible element $c \leftarrow_r \chi_\gamma$, and $\mathbf{z} \leftarrow_r D_{\mathbb{Z}^{2n},\beta}$;
  2. Interpreting $\mathbf{z}$ as two ring elements $\hat{r}_i, \hat{f}_i \in R_q$, and define $x_i = a\hat{r}_i + 2\hat{f}_i - p_i c$.
  3. Repeat the steps $1 \sim 3$ with probability $1 - 1/M$;
  4. Abort if there is a tuple $((i, j, x_i), *)$ in L_1. Else, add $((i, j, x_i), c)$ into L_1. Return x_i to $\mathcal{A}$.

Claim 32 *If* $\beta = \omega(\alpha\gamma n\sqrt{n \log n})$ *and RLWE*$_{q,\beta}$ *is hard, then* $\Pr[F_{5,2}] = \Pr[F_{5,1}] - \mathrm{negl}(\kappa)$.

Proof. This claim can be proven via a sequence of games similar to that from $G_{1,2}$ to $G_{1,4}$, we omit the details. □

Game $G_{5,3}$ $\mathcal{S}$ behaves almost the same as in $G_{5,2}$ except in the following case:

- $\mathsf{Send}_0(\Pi, I, i, j)$: If $(i, j) \neq (i^*, j^*)$, or it is not the s_{i^*}-th session of i^*, $\mathcal{S}$ answers as in Game $G_{5,2}$. Otherwise, it proceeds as follows:
  1. Sample an invertible element $c \leftarrow_r \chi_\gamma$, and $\hat{x}_i \leftarrow_r R_q$;
  2. Define $x_i = \hat{x}_i - p_i c$.
  3. Repeat the steps $1 \sim 3$ with probability $1 - 1/M$;
  4. Abort if there is a tuple $((i, j, x_i), *)$ in L_1. Else, add $((i, j, x_i), c)$ into L_1. Return x_i to $\mathcal{A}$.
- $\mathsf{Send}_2(\Pi, I, i, j, x_i, (y_j, w_j))$: If $(i, j) \neq (i^*, j^*)$, or it is not the s_{i^*}-th session of i^*, or y_j is output by the s_j^*-th session of party j^*, $\mathcal{S}$ behaves the same as in $G_{5,2}$. Otherwise, it proceeds as follows:
  5. Randomly choose $k_i \leftarrow_r R_q$;
  6. Compute $\sigma_i = \mathsf{Mod}_2(k_i, w_j)$ and derive the session key $\mathrm{sk}_i = H_2(i, j, x_i, y_j, w_j, \sigma_i)$.

Claim 33 $\Pr[F_{5,3}] = \Pr[F_{5,2}] - \mathrm{negl}(\kappa)$.

Proof. To prove this claim, we have to distinguish two cases: 1) y_j is output by party $j = j^*$ at session $s_j \neq s_j^*$ in response to a $\mathsf{Send}_1(\Pi, R, j, i^*, x_i)$ query; 2) y_j is not output by party $j = j^*$ at any session in response to a $\mathsf{Send}_1(\Pi, R, j, i^*, x_i)$ query. For the first case, the adversary just replays the messages to activate two sessions at party j. This situation is very similar to a Type I adversary, and one can prove this calim via a sequence of games as that from Game $G_{1,5}$ to $G_{1,7}$. While for the second case, this claim can also be proven via a sequence of games as that from Game $G_{2,3}$ to $G_{2,5}$. We omit the details here. □

Game $G_{5,4}$. $\mathcal{S}$ behaves almost the same as in $G_{5,3}$, except in the following case:

- $\mathsf{Send}_1(\Pi, R, j, i, x_i)$: If $(i, j) \neq (i^*, j^*)$, or it is not the s_j^*-th session of j^*, $\mathcal{S}$ answers the query as in Game $G_{5,3}$. Otherwise, it proceeds as follows:
  1′. Sample an invertible element $d \leftarrow_r \chi_\gamma$, and $\hat{y}_j \leftarrow_r R_q$;
  2′. Define $y_j = \hat{y}_j - p_j d$.
  3′. Repeat the steps $1' \sim 3'$ with probability $1 - 1/M$;
  4′. Abort if there is a tuple $((j, i, y_j, x_i), *)$ in L_1. Else, add $((j, i, y_j, x_i), d)$ into L_1. Then, the simulator $\mathcal{S}$ uniformly chooses $k_j \leftarrow_r R_q$ at random;

5′. Compute $w_j = \mathsf{Cha}(k_j) \in \{0,1\}^n$ and return (y_j, w_j) to $\mathcal{A}$;
6′. Compute $\sigma_j = \mathsf{Mod}_2(k_j, w_j)$ and derive the session key $\mathrm{sk}_j = H_2(i, j, x_i, y_j, w_j, \sigma_j)$.

Claim 34 *Let n be a power of 2 satisfying $0.97n \geq 2\kappa$, prime $q > 203$ satisfying $q = 1 \mod 2n$, $\beta = \omega(\alpha\gamma n\sqrt{n \log n})$. Then, if RLWE$_{q,\alpha}$ is hard, Game $G_{5,3}$ and $G_{5,4}$ are computationally indistinguishable. In particular, we have that $\Pr[F_{5,4}] = \Pr[F_{5,3}] - \mathrm{negl}(\kappa)$.*

Proof. The proof is similar to Claim 8, we omit the details. □

Claim 35 *If $0.97n > 2\kappa$, $\Pr[F_{5,4}] = 1/2 + \mathrm{negl}(\kappa)$.*

Proof. Let (y_j, w_j) be output by the s_j^*-th session of party $j = j^*$, we have $\mathrm{sk}_j = H_2(i, j, x_i, y_j, w_j, \sigma_j)$, where $\sigma_j = \mathsf{Mod}_2(k_j, w_j)$. Note that in $G_{5,4}$, k_j is randomly chosen from the uniform distribution over R_q, we have $\sigma_j \in \{0,1\}^n$ (conditioned on w_j) has min-entropy at least $0.97n$ by Lemma 4. Thus, the probability that $\mathcal{A}$ has made a H_2 query with σ_j is less than $2^{-0.97n} + \mathrm{negl}(\kappa)$. This claim follows from the fact that if the adversary doesn't make a query with σ_j exactly, the distribution of sk_j is uniform over $\{0,1\}^k$ due to the random oracle property of H_2, *i.e.,* $\Pr[F_{5,4}] = 1/2 + \mathrm{negl}(\kappa)$. □

Combining the claims 30∼35, we have that Lemma 9 follows. □

5 One-Pass Protocol from Ring-LWE

As MQV [57] and HMQV [46], our AKE protocol has a one-pass variant, which only consists a single message from one party to the other. Let $a \in R_q$ be the global public parameter uniformly chosen from R_q at random, and M be a constant. Let $p_i = as_i + 2e_i \in R_q$ be party i's static public key, where (s_i, e_i) is the corresponding static secret key; both s_i and e_i are taken from the distribution χ_α. Similarly, party j has static public key $p_j = as_j + 2e_j$ and static secret key (s_j, e_j). The other parameters and notations used in this section are the same as before.

Initiation Party i proceeds as follows:

1. Sample $r_i, f_i \leftarrow_r \chi_\beta$ and compute $x_i = ar_i + 2f_i$;
2. Compute $c = H_1(i, j, x_i)$, $\hat{r}_i = s_i c + r_i$ and $\hat{f}_i = e_i c + f_i$;
3. Letting $\mathbf{z} \in \mathbb{Z}^{2n}$ be the coefficient vector of $\hat{r}_i$ concatenated with the coefficient vector of $\hat{f}_i$, and $\mathbf{z}_1 \in \mathbb{Z}^{2n}$ be the coefficient vector of $s_i c$ concatenated with the coefficient vector of $e_i c$, repeat the steps $1 \sim 3$ with probability $1 - \min\left(\frac{D_{\mathbb{Z}^{2n},\beta}(\mathbf{z})}{MD_{\mathbb{Z}^{2n},\beta,\mathbf{z}_1}(\mathbf{z})}, 1\right)$.
4. Sample $g_i \leftarrow_r \chi_\beta$ and compute $k_i = p_j \hat{r}_i + 2g_i$ where $c = H_1(i, j, x_i)$;
5. Compute $w_i = \mathsf{Cha}(k_i) \in \{0,1\}^n$ and send (y_i, w_i) to party j;
6. Compute $\sigma_i = \mathsf{Mod}_2(k_i, w_i)$ and derive the session key $\mathrm{sk}_i = H_2(i, j, x_i, w_i, \sigma_i)$.

Finish Party j receives the pair (x_i, w_i) from party i, and proceeds as follows:

1′. Sample $g_j \leftarrow_r \chi_\alpha$ and compute $k_j = (p_i c + x_i)s_j + 2cg_j$ where $c = H_1(i, j, x_i)$;
2′. Compute $\sigma_j = \mathsf{Mod}_2(k_j, w_i)$ and derive the session key $\mathrm{sk}_j = H_2(i, j, x_i, w_i, \sigma_j)$.

The correctness of the protocol simply follows from the fact that $k_i = p_j \hat{r}_i + 2g_i = (as_j + 2e_j)(s_i c + r_i) + 2g_i \approx a(s_i c + r_i)s_j + 2(e_i c + f_i)s_j + 2cg_j = k_j$. The security of the protocol cannot be proven in the BR model with party corruption, since the one-pass protocol inherently can never provide wPFS due to the lack of message from the receiver j. Besides, the protocol cannot prevent the adversary from replaying a previous message, even though this can be detected in practice by using other measures such as keeping

a state or using synchronized time. However, we can prove it in a weak model similar to [46] which avoids the (above) inherent insufficiencies for one-pass protocol. Since the proof goes the lines with the previous one, we omit the details.

Finally, we remark that the one-pass protocol can essentially be used as a KEM, and can be transformed into a CCA encryption in the random oracle model by combining it with a CPA-secure symmetric-key encryption together with a MAC algorithm in a standard way (where both keys are derived from the session key in the one-pass protocol). The resulting encryption has two interesting properties: 1) it allows the receiver to verify the sender's identity, but no one else can verify it (since only the receiver can compute the session key, *i.e.*, it provides some kind of sender authentication); 2) the sender can deny having created such a ciphertext, because the receiver can also create such a ciphertext by itself (*i.e.*, it is a deniable encryption).

6 Concrete Parameters and Timings

In this section, we present concrete choices of parameters, and the timings in a proof-of-concept implementation. Our selection of parameters for our AKE protocols can be found in Table 2. Those parameters were chosen such that the correctness property is satisfied with high probability and with the choice of different levels of security.

For correctness we must satisfy that the error term $\|\widetilde{g}_i - \widetilde{g}_j\|_\infty < q/8$. Note that $\widetilde{g}_i = (e_j d + f_j)(s_i c + r_i) + dg_i$, and $\widetilde{g}_j = (e_i c + f_i)(s_j d + r_j) + cg_j$, where $e_i, e_j \leftarrow_r \chi_\alpha, c, d \leftarrow_r \chi_\gamma$, and $f_i, f_j, r_i, r_j, g_i, g_j \leftarrow_r \chi_\beta$. Due to the symmetry, we only estimate the size of $\|\widetilde{g}_i\|_\infty$. At this point, we use the following fact about the product of two Gaussian distributed random values (as stated in [9]). Let $x \in R$ and $y \in R$ be two polynomials whose coefficients are distributed according to a discrete Gaussian distribution with standard deviation σ and τ, respectively. The individual coefficients of the product xy are then (approximately) normally distributed around zero with standard deviation $\sigma\tau\sqrt{n}$ where n is the degree of the polynomial.

In our case, it means that we have $\|(e_j d + f_j)(s_i c + r_i)\|_\infty \leq 6\beta^2\sqrt{n}$ and $\|dg_i\|_\infty \leq 6\gamma\beta\sqrt{n}$ with overwhelming probability (since erfc(6) is about 2^{-55}). Note that the distributions of $e_j d + f_j$ and $s_i c + r_i$ are both according to χ_β since we use the rejection sampling in the protocol. Now, to choose an appropriate β we set $d = 1/2$ in Lemma 1 such that $\|e_j d\|, \|s_i c\| \leq 1/2\alpha\gamma n$ with probability at most $2 \cdot 0.943^{-n}$. Hence, for $n \geq 1024$, we get a potential decryption error with only a probability about 2^{-87}. In order to make the rejection sampling work, it is sufficient to set $\beta \geq \tau * 1/2\alpha\gamma n = 1/2\tau\alpha\gamma n$ for some constant τ (which is much better than the worst-case bound $\beta = \omega(\alpha\gamma\sqrt{n \log n})$ in Theorem 1). For instance, if $\tau = 12$, we have an expect number of rejection sampling about $M = 2.72$ and a statistical distance about $\frac{2^{-100}}{M}$ by Theorem 1. For such a choice of β, we can safely assume that $\|\widetilde{g}_i\|_\infty \leq 6\beta^2\sqrt{n} + 6\gamma\beta\sqrt{n} \leq 7\beta^2\sqrt{n}$. Thus, it is enough to set $16 * 7\beta^2\sqrt{n} < q$ for correctness of the protocol.

Though the Ring-LWE problem enjoys a worst-case connection to some hard problems (*e.g.*, SIVP [54]) on ideal lattices, the connection as summarized in Proposition 1 seems less powerful to estimate the actual security for concrete choices of parameters. In order to assess the concrete security of our parameters, we use the approach of [21], which investigates the two most efficient ways to solve the underlying (R)LWE problem, namely the embedding and decoding attacks. As opposed to [21], the decoding attack is more efficient against our instances because in RLWE with $m \geq 2n$ one typically is close to the optimal attack dimension for the corresponding attacks. The decoding attack first uses a lattice reduction algorithm, such as BKZ [63] / BKZ 2.0 [18] and then applies a decoding algorithm, such as Babai's nearest plane [2], Lindner and Peikert's nearest planes [50], or Liu and Nguyen's pruned enumeration approach [51]. Finally, the closest vector is returned which coincides with the error polynomial, and the secret polynomial is recovered.

Table 2. Choices of Parameters (The bound 6α with $\mathrm{erfc}(6) \approx 2^{-55}$ is used to estimate the size of secret keys)

Protocol	Choice of Parameters	n	Security	α	τ	$\log \beta$	$\log q$ (bits)	Size (KB)			
								pk	sk (expt.)	init. msg	resp. msg
Two-pass	I_1	1024	80 bits	3.397	12	16.1	45	5.625 KB	1.5 KB	5.625 KB	5.75 KB
	I_2		75 bits	3.397	24	17.1	47	5.875 KB	1.5 KB	5.875 KB	6.0 KB
	II_1	2048	230 bits	3.397	12	17.1	47	11.75 KB	3.0 KB	11.75 KB	12.0 KB
	II_2		210 bits	3.397	36	18.7	50	12.50 KB	3.0 KB	12.50 KB	12.75 KB
One-pass	III_1	1024	160 bits	3.397	12	16.1	30	3.75 KB	1.5 KB	3.75 KB	3.875 KB
	III_2		140 bits	3.397	36	17.7	32	4.0 KB	1.5 KB	4.0 KB	4.125 KB
	IV_1	2048	360 bits	3.397	12	17.1	32	8.0 KB	3.0 KB	8.0 KB	8.25 KB
	IV_2		350 bits	3.397	36	18.7	33	8.25 KB	3.0 KB	8.25 KB	8.5 KB

As recommended in [50,35], it is enough to set the Gaussian parameter $\alpha \geq 3.2$ so that the discrete Gaussian $D_{\mathbb{Z}^n,\alpha}$ approximates the continuous Gaussian D_α extremely well[9]. In our experiment, we fix $\alpha = 3.397$ for a better performance of the Gaussian sampling algorithm in [30]. As for the choices of γ, we set $\gamma = \alpha$ for simplicity (actually such a choice in our experiments works very well: no rejection happened for 1000 times hash evaluations). In Table 1, we set all other parameters β, n, q for our two-pass protocol to satisfy the correctness condition. We also give the parameter choices of our one-pass protocol (in this case, we can save a factor of β in q due to the asymmetry). Note that n is required to be a power of 2 in our protocol (*i.e.*, it is very sparsely distributed[10]), we present several candidate choices of parameters for $n = 1024, 2048$, and estimate the sizes of public keys, secret keys, and communication overheads in Table 2.

Table 3. Timings of proof-of-concept implementations in ms.

Protocol	Parameters	τ	Initiation	Response	Finish
Two-pass	I_1	12	22.05 ms	30.61 ms	4.35 ms
	I_2	24	14.26 ms	19.18 ms	4.41 ms
	II_1	12	49.77 ms	60.31 ms	9.44 ms
	II_2	36	25.40 ms	36.96 ms	9.59 ms

Protocol	Parameters	τ	Initiation	Finish
One-pass	III_1	12	26.17 ms	3.64 ms
	III_2	36	14.57 ms	3.70 ms
	IV_1	12	53.78 ms	7.75 ms
	IV_2	36	32.28 ms	7.94 ms

We implement our AKE protocol by using the NTL library compiled with the option NTL_GMP_LIP=on (*i.e.*, building NTL using the GNU Multi-Precision package). The implementations are written in C++ without any parallel computations or multi-threads programming techniques. The program is run on a Dell Optiplex 780 computer with Ubuntu 12.04 TLS 64-bit system, equipped with a 2.83GHz Intel Core 2 Quad CPU and 3.8GB RAM. We use a n-dimensional Fast Fourier Transform (FFT) for the multiplications of two ring elements [20,53]. We use the CDT algorithm [60] as a tool for hashing to $D_{\mathbb{Z}^n,\gamma}$ and sampling from $D_{\mathbb{Z}^n,\alpha}$, but use the DDLL algorithm [30] for sampling from $D_{\mathbb{Z}^n,\beta}$ (because the CDT algorithm has to store large precomputed values for a big β). In Table 3, we present the timings of each operation, and the figures represent the averaged timing (in millisecond, ms) for 1000 executions. Since our protocols also allow some kind of precomputations such as sampling Gaussian distributions offline, the timings can be greatly reduced if one consider it in practice. Finally, we note that our implementation has not undergone any real optimization, and it can much improved in practice.

[9] Only α is considered because $\beta \gg \alpha$, and the (R-)LWE problem becomes harder as α grows bigger (for a fixed modulus q).

[10] We remark such a choice of n is not necessary, but it gives a simple analysis and implementation. In practice, one might use the techniques for Ring-LWE cryptography in [55] to give a tighter choice of parameters for desired security levels.

References

1. Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO*, pages 595–618. 2009.
2. László Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.
3. Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. In *CT-RSA*, pages 28–47, 2014.
4. B. Barak, R. Impagliazzo, and A. Wigderson. Extracting randomness using few independent sources. *SIAM Journal on Computing*, 36(4):1095–1118, 2006.
5. E. Barker and A. Roginsky. Recommendation for the entropy sources used for random bit generation. Draft NIST Special Publication 800-90B, August 2012.
6. Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *CCS*, pages 390–399, 2006.
7. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS*, pages 62–73, 1993.
8. Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *CRYPTO*, volume 773, pages 232–249. 1994.
9. Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. Cryptology ePrint Archive, Report 2014/599, 2014.
10. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. Fully homomorphic encryption without bootstrapping. *Innovations in Theoretical Computer Science, ITCS*, pages 309–325, 2012.
11. Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from Ring-LWE and security for key dependent messages. In *CRYPTO*, pages 505–524. 2011.
12. Christina Brzuska, Marc Fischlin, Nigel P. Smart, Bogdan Warinschi, and Stephen C. Williams. Less is more: relaxed yet composable security notions for key exchange. *Int. J. Inf. Sec.*, 12(4):267–297, 2013.
13. Christina Brzuska, Marc Fischlin, Bogdan Warinschi, and Stephen C. Williams. Composability of bellare-rogaway key exchange protocols. In *CCS*, pages 51–62, 2011.
14. BSI. Advanced security mechanism for machine readable travel documents extended access control (eac). Technical Report (BSI-TR-03110) Version 2.05 Release Candidate, Bundesamt fuer Sicherheit in der Informationstechnik (BSI), 2010.
15. Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *EUROCRYPT*, pages 453–474. 2001.
16. Ran Canetti and Hugo Krawczyk. Security analysis of IKEs signature-based key-exchange protocol. In *CRYPTO*, pages 143–161. 2002.
17. Lily Chen. Practical impacts on qutumn computing. Quantum-Safe-Crypto Workshop at the European Telecommunications Standards Institute, 2013. http://docbox.etsi.org/Workshop/2013/201309_CRYPTO/S05_DEPLOYMENT/NIST_CHEN.pdf.
18. Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In *ASIACRYPT*, pages 1–20, 2011.
19. Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. In *FOCS*, pages 429–442, 1985.
20. Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, Clifford Stein, et al. *Introduction to algorithms*, volume 2. MIT press Cambridge, 2001.
21. Özgür Dagdelen, Rachid El Bansarkhani, Florian Göpfert, Tim Güneysu, Tobias Oder, Thomas Pöppelmann, Ana Helena Snchez, and Peter Schwabe. High-speed signatures from standard lattices. In *to appear at LATINCRYPT*. 2014.
22. Özgür Dagdelen and Marc Fischlin. Security analysis of the extended access control protocol for machine readable travel documents. In *ISC*, pages 54–68, 2010.
23. Özgür Dagdelen, Marc Fischlin, Tommaso Gagliardoni, Giorgia Azzurra Marson, Arno Mittelbach, and Cristina Onete. A cryptographic analysis of OPACITY - (extended abstract). In *ESORICS*, pages 345–362, 2013.
24. Jean Paul Degabriele, Victoria Fehr, Marc Fischlin, Tommaso Gagliardoni, Felix Günther, Giorgia Azzurra Marson, Arno Mittelbach, and Kenneth G. Paterson. Unpicking PLAID - a cryptographic analysis of an ISO-standards-track authentication protocol. Cryptology ePrint Archive, Report 2014/728, 2014.
25. Tim Dierks. The transport layer security (TLS) protocol version 1.2. 2008.
26. W. Diffie and M. Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644 – 654, nov 1976.
27. Jintai Ding, Xiang Xie, and Xiaodong Lin. A simple provably secure key exchange scheme based on the learning with errors problem. Cryptology ePrint Archive, Report 2012/688, 2012.
28. Yevgeniy Dodis, Rosario Gennaro, Johan Håstad, Hugo Krawczyk, and Tal Rabin. Randomness extraction and key derivation using the CBC, Cascade and HMAC modes. In *CRYPTO*, pages 494–510. 2004.

29. Léo Ducas and Alain Durmus. Ring-LWE in polynomial rings. In *PKC*, pages 34–51, 2012.
30. Lo Ducas, Alain Durmus, Tancrde Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal Gaussians. In *CRYPTO*, pages 40–56. 2013.
31. Alan Freier. The SSL protocol version 3.0. *http://wp. netscape. com/eng/ssl3/draft302. txt*, 1996.
32. Atsushi Fujioka, Koutarou Suzuki, Keita Xagawa, and Kazuki Yoneyama. Strongly secure authenticated key exchange from factoring, codes, and lattices. In *PKC*, pages 467–484. 2012.
33. Atsushi Fujioka, Koutarou Suzuki, Keita Xagawa, and Kazuki Yoneyama. Practical and post-quantum authenticated key exchange from one-way secure key encapsulation mechanism. In *ASIACCS*, pages 83–94, 2013.
34. Rosario Gennaro and Victor Shoup. A note on an encryption scheme of Kurosawa and Desmedt. Cryptology ePrint Archive, Report 2004/194, 2004.
35. Craig Gentry, Shai Halevi, and NigelP. Smart. Homomorphic evaluation of the AES circuit. In *CRYPTO*, pages 850–867. 2012.
36. Florian Giesen, Florian Kohlar, and Douglas Stebila. On the security of TLS renegotiation. In *CCS*, pages 387–398, 2013.
37. Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In *Innovations in Computer Science*, pages 230–240, 2010.
38. Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In *CHES*, pages 530–547, 2012.
39. Dan Harkins, Dave Carrel, et al. The internet key exchange (IKE). Technical report, RFC 2409, november, 1998.
40. Jeffrey Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, and William Whyte. Practical signatures from the partial fourier recovery problem. In *ACNS*, pages 476–493, 2014.
41. ISO/IEC. 11770-3:2008 information technology – security techniques – key management – part 3: Mechanisms using asymmetric techniques.
42. Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DHE in the standard model. In *CRYPTO*, pages 273–293. 2012.
43. Jonathan Katz and Vinod Vaikuntanathan. Smooth projective hashing and password-based authenticated key exchange from lattices. In *ASIACRYPT*, pages 636–652. 2009.
44. Charlie Kaufman, Paul Hoffman, Yoav Nir, and Pasi Eronen. Internet key exchange protocol version 2 (IKEv2). Technical report, RFC 5996, September, 2010.
45. Hugo Krawczyk. SIGMA: The 'SIGn-and-MAc' approach to authenticated Diffie-Hellman and its use in the IKE protocols. In *CRYPTO*, pages 400–425. 2003.
46. Hugo Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In *CRYPTO*, pages 546–566. 2005.
47. Hugo Krawczyk, KennethG. Paterson, and Hoeteck Wee. On the security of the TLS protocol: A systematic analysis. In *CRYPTO*, pages 429–448. 2013.
48. Brian A. LaMacchia, Kristin E. Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. In *ProvSec*, pages 1–16, 2007.
49. Xinyu Lei and Xiaofeng Liao. NTRU-KE: A lattice-based public key exchange protocol. Cryptology ePrint Archive, Report 2013/718, 2013.
50. Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In *CT-RSA*, pages 319–339. 2011.
51. Mingjie Liu and Phong Q. Nguyen. Solving BDD by enumeration: An update. In *CT-RSA*, pages 293–309. 2013.
52. Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT*, pages 738–755. 2012.
53. Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. SWIFFT: A modest proposal for FFT hashing. In *FSE*, pages 54–72. 2008.
54. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, pages 1–23. 2010.
55. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for Ring-LWE cryptography. In *EUROCRYPT*, pages 35–54. 2013.
56. Nikos Mavrogiannopoulos, Frederik Vercauteren, Vesselin Velichkov, and Bart Preneel. A cross-protocol attack on the TLS protocol. In *CCS*, pages 62–72, 2012.
57. A. Menezes, M. Qu, and S. Vanstone. Some new key agreement protocols providing mutual implicit authentication. In *SAC*, 1995.
58. Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37:267–302, 2007.
59. Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *STOC*, pages 333–342, 2009.

60. Chris Peikert. An efficient and parallel Gaussian sampler for lattices. In *CRYPTO*, pages 80–97. 2010.
61. Chris Peikert. Lattice cryptography for the Internet. Cryptology ePrint Archive, Report 2014/070, 2014.
62. Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *STOC*, pages 187–196, 2008.
63. Claus-Peter Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Math. Program.*, 66:181–199, 1994.
64. P. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.
65. Damien Stehlé and Ron Steinfeld. Making NTRU as secure as worst-case problems over ideal lattices. In *EUROCRYPT*, pages 27–47. 2011.
66. L. Trevisan and S. Vadhan. Extracting randomness from samplable distributions. In *FOCS*, pages 32–, 2000.
67. Luca Trevisan. Extractors and pseudorandom generators. *J. ACM*, 48(4):860–879, July 2001.
68. Andrew Chi-Chih Yao and Yunlei Zhao. OAKE: A new family of implicitly authenticated Diffie-Hellman protocols. In *CCS*, pages 1113–1128, 2013.