Authenticated Key Exchange from Ideal Lattices

Jiang Zhang[1], Zhenfeng Zhang[1,⋆], Jintai Ding[2,3,⋆],
Michael Snook[3], and Özgür Dagdelen[4]

[1] Trusted Computing and Information Assurance Laboratory, SKLCS,
Institute of Software, Chinese Academy of Sciences, Beijing, China
[2] Heshi Inc., China
[3] Department of Mathematical Sciences, University of Cincinnati, Cincinnati, USA
[4] Technische Universität Darmstadt, Germany
jiangzhang09@gmail.com, zfzhang@tca.iscas.ac.cn, jintai.ding@gmail.com,
snookml@mail.uc.edu, oezguer.dagdelen@cased.de

Abstract. In this paper, we present a practical and provably secure two-pass authenticated key exchange protocol over 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 method does not involve other cryptographic primitives—in particular, it does not use signatures—which simplifies the protocol and enables us to base the security directly on the hardness of the ring learning with errors problem. The security is proven in the Bellare-Rogaway model with weak perfect forward secrecy in the random oracle model. 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 [1], cryptographers have devised a wide selection of KE protocols with various use-cases. One such class is Authenticated Key Exchange (AKE), which enables each party to verify the other's identity so that an adversary cannot impersonate an honest 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. During an execution of the protocol, each party generates a pair of ephemeral keys—an *ephemeral secret key* and an *ephemeral public key*—and sends the *ephemeral public key* to the other party. Then, these 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 a common session key. Intuitively,

⋆ Corresponding authors.

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 [2] 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 [3] and LaMacchia *et al.* [4], that also account 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 the Heartbleed attack[5]). As shown in [5], no two-pass *implicit* AKE protocol based on public-key authentication can achieve PFS (but this may not be true for two-pass AKEs with *explicit* authentication [6]). Thus, the notion of weak PFS (wPFS) is usually considered for two-pass implicit 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 [5].

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 MACs), 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 [7,8], SIGMA [9], SSL [10], TLS [11–15], as well as the standard in German electronic identity cards, namely EAC [16], and the standardized protocols OPACITY [17] and PLAID [18]. Another line of designing AKEs follows the idea of MTI [19] and MQV [20],[6] which aims at providing implicit authentication by directly utilizing the algebraic structure of DH problems (*e.g.*, HMQV [5] and OAKE [26]). All the above AKEs are based on classic hard problems, such as factoring, the RSA problem, or the computational/decisional DH problem. Since these hard problems are vulnerable to quantum computers [27] 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 [28]. 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 [29,30]. Our method avoids introducing extra cryptographic primitives, thus simplifying the design and reducing overhead. In particular, the parties are not required to either encrypt any messages with the other's public key, nor sign any

[5] `http://heartbleed.com/`

[6] Note that MQV has been widely standardized by ANS [21,22], ISO/IEC [23] and IEEE [24], and recommended by NIST and NSA Suite B [25].

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 directly rely on the hardness of Ring-LWE in the random oracle model.

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 [5]. 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 "nicely-behaved" cyclic groups, the security of our protocol cannot be proven in the CK model [3] due to the underlying noise-based algebraic structures. Fortunately, we prove the security in the BR model (adapted to the public-key setting [31]), which is the most common model considered as it is usually strong enough for many practical applications and it comes with composability [32]. In addition, our protocol achieves the weak PFS property, which is known as the best PFS notion achievable by two-pass AKEs with implicit authentication [5].

As MQV [20] and HMQV [5], we also present a one-pass variant of our basic protocol (*i.e.*, only a single message is needed 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. Though the implementation has not undergone any real optimization, the performance results already indicate that our protocols are practical.

Besides, we note that none of the techniques we use prevents 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 [5], 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 similarities and differences between our AKE protocol and HMQV, we first briefly recall the HMQV protocol [5]. 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, *i.e.,* 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

Party i		Party j
Public Key: $p_i = as_i + 2e_i \in R_q$		Public Key: $p_j = as_j + 2e_j \in R_q$
Secret Key: $s_i \in R_q$		Secret Key: $s_j \in R_q$
where $s_i, e_i \leftarrow_r \chi_\alpha$		where $s_j, e_j \leftarrow_r \chi_\alpha$
$x_i = ar_i + 2f_i \in R_q$	$\xrightarrow{x_i}$	$y_j = ar_j + 2f_j \in R_q$
where $r_i, f_i \leftarrow_r \chi_\beta$		$k_j = (p_i c + x_i)(s_j d + r_j) + 2cg_j$
		where $r_j, f_j, g_j \leftarrow_r \chi_\beta$
$k_i = (p_j d + y_j)(s_i c + r_i) + 2dg_i$	$\xleftarrow{y_j, w_j}$	$w_j = \mathsf{Cha}(k_j) \in \{0,1\}^n$
where $g_i \leftarrow_r \chi_\beta$		$\sigma_j = \mathsf{Mod}_2(k_j, w_j) \in \{0,1\}^n$
$\sigma_i = \mathsf{Mod}_2(k_i, w_j) \in \{0,1\}^n$		$\mathrm{sk}_j = H_2(i, j, x_i, y_j, w_j, \sigma_j)$
$\mathrm{sk}_i = H_2(i, j, x_i, y_j, w_j, \sigma_i)$		

$$c = H_1(i, j, x_i) \in R, d = H_1(j, i, y_j, x_i) \in R$$

Fig. 1. Our AKE protocol from Ring-LWE.

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 greatly relies on the nice properties 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 [33–35], the Ring-LWE problem supports some kind of "approximate" commutativity, and can be used to build a 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 choosen 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 [33]. 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 (*e.g.*, k_i at party i or k_j at party 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 with identifiers $\mathrm{sid} = (i, j, x_i, y_j, w_j)$ and $\mathrm{sid}' =$

(i, j, x_i, y_j, w'_j) have the same session state, *i.e.,* k_i at party i.[7] 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 (especially for the signal part w_j) is too loose. In particular, the fact that $\sigma_i = \mathsf{Mod}_2(k_i, w_j)$ 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), may potentially help the adversary distinguish σ_i with the knowledge of σ'_i. We prevent the adversary from utilizing this weakness by setting the session key as the output of the hash function H_2 (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 using a good randomness extractor H_2, *e.g.*, a random oracle).[8]

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 sum 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 [36]. The security proof of our protocol is based on the observation that such a technique allows to statistically hide the distribution of χ_α in a bigger distribution χ_β, 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, *i.e.*, the one 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 poses an obstacle to getting 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 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. The properties of Gaussian distributions and the random oracle H_1 implies that y_j has almost identical distribution as 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 the pair (p_i, x_i) is honestly generated (and

[7] This problem might not exist if one consider a different definition of session identifier, *e.g.,* the one that was uniquely determined at the beginning of the protocol execution.

[8] We remark that this is also the reason why the nice reconciliation mechanism in [34] cannot be used in our protocol. Specifically, it is unclear whether the reconciliation function $\mathsf{rec}(\cdot, \cdot)$ in [34] 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.

thus is computationally indistinguishable from uniformly distributed element under the Ring-LWE assumption), or else there is no "secrecy" to protect if both p_i and x_i are chosen by the adversary. That is, $p_i c + x_i$ is always random 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 $\hat{x}_i = cu_i$ 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, when proving the security one can replace k_j with a uniformly distributed element to derive a high min-entropy key material σ_j by using the Mod_2 function as required.

Unfortunately, directly using "noise flooding" has a significant drawback, *i.e.*, the requirement of a super-polynomially large standard deviation β, which may lead to a nightmare for practical performance due to a super-polynomially large modulus q for correctness and a very large ring dimension n for the hardness of the underlying Ring-LWE problems. Fortunately, we can reduce the big cost by further employing the rejection sampling technique [37]. Rejection sampling is a crucial technique in signature schemes to make the distribution of signatures independent of the signing key, and has been applied in many other lattice-based signature schemes [38–41].

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 large 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 in lattice-based cryptography. 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 large 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 [42] proposed the first password-based authenticated key exchange protocol that can be proven secure based on the LWE assumption. Ding *et al.* [33] elegantly constructed a passive-secure KE protocol on (Ring-)LWE by using a nice error-removing technique with a signal message. Like the standard DH protocol, the protocol in [33] could not provide authentication—it is not an AKE protocol—and is thus vulnerable to man-in-the-middle attacks. This motivates us to design an efficient AKE protocol on (ideal) lattices, especially an MQV-style one with implicit authentication.

Since the work of Katz et al. [42], there are four papers focusing on designing AKEs from lattices [34, 35, 43, 44]. At a high level, all of them are following generic transformations from key encapsulation mechanisms (KEM) to AKEs. Concretely, Fujioka *et*

Table 1. Comparison of lattice-based AKEs (CCA† means CCA-security with high min-entropy keys [43], and EUF-CMA means existential unforgeability under chosen message attacks)

Protocols	KEM/PKE	Signature	Message-pass	Model	RO?	Num. of R_q
FSXY12 [43]	CCA†	-	2-pass	CK	$\times$	$\gg 7$
FSXY13 [44]	OW-CCA	-	2-pass	CK	$\surd$	7
Peikert14 [34]	CPA	EUF-CMA	3-pass	SK-security	$\surd$	> 2 *
BCNS14 [35]	CPA	EUF-CMA	4-pass	ACCE	$\surd$	2 for KEM **
Ours	-	-	2-pass	BR with wPFS	$\surd$	2

* The actual number of ring elements depends on the choice of the concrete lattice-based signatures.

** Since the protocol uses traditional signatures to provide authentication, it does not contain any other ring elements.

al. [43] 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 of pseudorandom functions (PRF), then there is a secure AKE protocol in the standard model. Thus, by using existing lattice-based CCA-secure KEMs such as [45, 46], 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. Later, the paper [44] tried to get a practical AKE protocol by improving the efficiency of the generic framework in [43], and showed that one-way CCA-secure KEMs were enough to get AKEs in the random oracle model. The two protocols in [43,44] share some similarities such as having two-pass messages, and involving three encryptions (*i.e.*, two encryptions under each party's static public key and one encryption under an ephemeral public key). However, the use of the random oracle heuristic makes the protocol in [44] more efficient than that in [43]. Specifically, the protocol in [44] requires exchanging seven ring elements when instantiated with the CPA-secure encryption from Ring-LWE [29] by first transforming it into a CCA-secure one with the Fujisaki-Okamoto transformation.

Recently, Peikert [34] presented an efficient KEM based on Ring-LWE, which was then transformed into an AKE protocol by using the same structure as SIGMA [9]. Similar to the SIGMA protocol, the resulting protocol had three-pass messages and was proven SK-secure [47] in the random oracle model. For the computation overheads, Peikert's protocol involved one KEM, two signatures and two MACs. By treating the KEM in [34] as a DH-like KE protocol, Bos *et al.* [35] integrated it into the Transport Layer Security (TLS) protocol by directly using signatures to provide explicit authentication. Actually, the authors used traditional digital signatures such as RSA and ECDSA, and thus their protocol was not a pure post-quantum AKE. As for the security, the protocol in [35] was proven secure in the authenticated and confidential channel establishment (ACCE) security model [48] (which is based on the BR model, but has many differences to capture entity authentication and channel security).

Due to 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 all, and its security relies on the hardness of Ring-LWE in the random oracle model. To the best of our knowledge there is not a single post-quantum authenticated key exchange protocol (until this work) which directly relies on a quantum-hard computational problem and does not make use of explicit cryptographic primitives except hash functions.

1.4 On the Quantum Hardness of our AKE Protocol

We call our AKE protocol post-quantum as our protocol relies merely on the Ring-LWE assumption, which is believed to hold even in presence of polynomial-time quantum computers. However, we emphasize that it does not mean necessarily that our scheme is quantum resistant. This may sound confusing and controversial in the beginning; that is why we clarify this issue in the following. While the underlying assumption may give the impression that our scheme is quantum secure, our security analysis makes use of rewinding the adversary, which is generally hard to apply to a quantum algorithm (exceptions can be found in [49, 50]). Moreover, our proof is done in the random oracle model. In [51], Boneh et al. introduced the quantum random oracle model, and show that proofs in this augmented model are more realistic when considering quantum adversaries. In fact, many well-known transformations proven secure in the classical random oracle model cannot be (easily) proven secure against quantum algorithms, such as the Fiat-Shamir transform [52, 53]. Moreover, it is not clear whether the security models for key exchange are appropriate when considering quantum adversaries. An update of security models (in general) may necessary when considering quantum adversaries (see [54, 55]). Therefore, we do not claim that our scheme is quantum resistant, but believe it is a big step forward.

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 [2, 31], 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 communication 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 as in [3] 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 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 [2], 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' secret key in advance and then passively eavesdrops the session sid^* (and thus $\widetilde{\mathrm{sid}}^*$). We remark that this seems to be stronger than what is needed for capturing wPFS [5], 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 $\rho_{\alpha,\mathbf{c}}(\mathbf{x}) = (\frac{1}{\sqrt{2\pi\alpha^2}})^m \exp\left(\frac{-\|\mathbf{x}-\mathbf{v}\|^2}{2\alpha^2}\right)$. For integer vectors $\mathbf{c} \in \mathbb{R}^n$, let

$\rho_{\alpha,\mathbf{c}}(\mathbb{Z}^m) = \sum_{\mathbf{x}\in\mathbb{Z}^m} \rho_{\alpha,\mathbf{c}}(\mathbf{x})$. Then, we define the discrete Gaussian distribution over $\mathbb{Z}^m$ as $D_{\mathbb{Z}^m,\alpha,\mathbf{c}}(\mathbf{x}) = \frac{\rho_{\alpha,\mathbf{c}}(\mathbf{x})}{\rho_{\alpha,\mathbf{c}}(\mathbb{Z}^m)}$, 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 ([56]). *Letting real $\alpha = \omega(\sqrt{\log m})$, constant $\eta > 1/\sqrt{2\pi}$, then we have that $\Pr_{\mathbf{x}\leftarrow_r D_{\mathbb{Z}^m,\alpha}}[\|\mathbf{x}\| > \eta \cdot \alpha\sqrt{m}] \leq \frac{1}{2}D^n$, where $D = \eta\sqrt{2\pi e} \cdot e^{-\pi\cdot\eta^2}$. 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 [37], which will be used in the security proof of our AKE protocol.

Theorem 1 (Rejection Sampling [37]). *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 $\frac{2^{-\omega(\log m)}}{M}$ from 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 (Euclidean) 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 $\|s \cdot t\|_\infty \leq n \cdot \|s\|_\infty \cdot \|t\|_\infty$.*

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 [29]. 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 $s \leftarrow_r R_q$. 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 Independent Vectors Problem (SIVP) over ideal lattices.

Proposition 1 (A special case of [29]). *Let n be a power of 2, α be a real number in $(0,1)$, and q be 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 [29, 57]. This variant is known as the *normal form*, and is preferable for controlling the size of the error term [58, 59]. The underlying Ring-LWE assumption also holds when scaling the error by a constant t relatively prime to q [58], *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 [58, 59]. 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 of the 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 [45].

3 Authenticated Key Exchange from Ring-LWE

We now introduce some notations. For an odd prime $q > 2$, take $\mathbb{Z}_q = \{-\frac{q-1}{2}, \ldots, \frac{q-1}{2}\}$ 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, $\mathsf{Mod}_2 : \mathbb{Z}_q \times \{0,1\} \to \{0,1\}$ as $\mathsf{Mod}_2(v,b) = (v + b \cdot \frac{q-1}{2}) \bmod q \bmod 2$.

In the following lemma, we show that given the bit $b = \mathsf{Cha}(v)$, and a value $w = v + 2e$ with sufficiently small e, one 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 two 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 (resp. Mod_2) to denote $\widetilde{\mathsf{Cha}}$ (resp. $\widetilde{\mathsf{Mod}_2}$). 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 "close" ring elements (by applying the Mod_2 function). Ideally, for a 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 an 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, and thus can be used to extract an (almost) uniformly distributed 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 $\{0,1\}$. Hence, 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 $\frac{1}{2|E|}$ 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 extractors can be used to obtain an almost uniformly distributed key from a biased bit-string with high min-entropy [60–64]. In practice, as recommended by NIST [65], 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 any positive $\gamma \in \mathbb{R}$, let $H_1 \colon \{0,1\}^* \to \chi_\gamma = D_{\mathbb{Z}^n,\gamma}$ be a hash function that always outputs invertible elements in R_q.[9] 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 [67]. 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).

[9] In practice, one can first use a hash function (*e.g.*, SHA-2) to obtain a uniformly random string, and then use it to sample from $D_{\mathbb{Z}^n,\gamma}$. The algorithm outputs a sample only if it is invertible in R_q, otherwise, it tries another sample and repeats. By Lemma 10 in [66], we can have a good probability to sample an invertible element in each trial for an appropriate choice of γ.

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. Go to step 4 with probability $\min\left(\frac{D_{\mathbb{Z}^{2n},\beta}(\mathbf{z})}{MD_{\mathbb{Z}^{2n},\beta,\mathbf{z}_1}(\mathbf{z})}, 1\right)$, where $\mathbf{z} \in \mathbb{Z}^{2n}$ is the coefficient vector of $\hat{r}_i$ concatenated with the coefficient vector of $\hat{f}_i$, and $\mathbf{z}_1 \in \mathbb{Z}^{2n}$ is the coefficient vector of $s_i c$ concatenated with the coefficient vector of $e_i c$; otherwise go back to step 1;
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 $\hat{f}_j = e_j d + f_j$;

$3'$. Go to step $4'$ with probability $\min\left(\frac{D_{\mathbb{Z}^{2n},\beta}(\mathbf{z})}{MD_{\mathbb{Z}^{2n},\beta,\mathbf{z}_1}(\mathbf{z})}, 1\right)$, where $\mathbf{z} \in \mathbb{Z}^{2n}$ is the coefficient vector of $\hat{r}_j$ concatenated with the coefficient vector of $\hat{f}_j$, and $\mathbf{z}_1 \in \mathbb{Z}^{2n}$ is the coefficient vector of $s_j d$ concatenated with the coefficient vector of $e_j d$; otherwise go back to step $1'$;

$4'$. Sample $g_j \leftarrow_r \chi_\beta$, 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:

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

Remark 2. Deploying our protocol practically in a large scale requires the support of a PKI with a trusted Certificate Authority (CA). In this setting, all the system parameters (such as a) will be generated by the CA like other PKI-based protocols.

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 is about $1 - \frac{1}{M}$ 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. Next, 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:

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

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 $q = 1 \mod 2n$, real $\beta = \omega(\alpha\gamma n\sqrt{n \log n})$ and let H_1, H_2 be random oracles. Then, if RLWE$_{q,\alpha}$ is hard, the proposed AKE is secure with respect to Definition 2.*

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 (*i.e.*, $0.97n > 2\kappa$) even conditioned on w_j by Lemma 4. Since H_2 is a random oracle, we have that sk_j is uniformly distributed over $\{0,1\}^\kappa$ as expected. Now, let us 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 3) $\hat{s}_j$ 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 [68] 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 (y_j, resp.) before seeing c (d, resp.), thus it cannot determine the value $p_i c + x_i$ or $p_j d + y_i$ in advance (but the simulator can set 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).

For space reason, we only give the security proof for **Type I** adversaries in Lemma 5, and defer the proofs for other types of adversaries to the full version.

Lemma 5. *Let n be a power of 2 satisfying $0.97n \geq 2\kappa$, prime $q > 203$ satisfying $q = 1 \mod 2n$, real $\beta = \omega(\alpha\gamma n\sqrt{n \log n})$. 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.*

*In particular, if there is a PPT **Type I** adversary $\mathcal{A}$ breaking our protocol with non-negligible advantage ϵ, then there is a PPT algorithm $\mathcal{B}$ solving RLWE$_{q,\alpha}$ with advantage at least $\frac{\epsilon}{m^2N^2} - \mathrm{negl}(\kappa)$.*

Proof. We prove this lemma via a sequence of games $G_{1,l}$ for $0 \leq l \leq 7$, where the first game $G_{1,0}$ is almost the same as the real one except that the simulator randomly guesses the test session at the beginning of the game and aborts the simulation if the guess is wrong, while the last game $G_{1,7}$ is a fake one with randomly and independently chosen session key for the test session (thus the adversary can only win the game with negligible advantage). The security is established by showing that any two consecutive games are computationally indistinguishable. **Bold fonts** 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 does not exist a tuple (in, out) in L_1, choose an invertible element $out \in \chi_\gamma$ at random, and add (in, out) into L_1. Then, return out to $\mathcal{A}$.

– $H_2(in)$ queries: If there does not 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. Go to step 4 with probability $\min\left(\frac{D_{\mathbb{Z}^{2n},\beta}(\mathbf{z})}{MD_{\mathbb{Z}^{2n},\beta,\mathbf{z}_1}(\mathbf{z})}, 1\right)$, where $\mathbf{z} \in \mathbb{Z}^{2n}$ is the coefficient vector of $\hat{r}_i$ concatenated with the coefficient vector of $\hat{f}_i$, and $\mathbf{z}_1 \in \mathbb{Z}^{2n}$ is the coefficient vector of $s_i c$ concatenated with the coefficient vector of $e_i c$; otherwise go back to step 1;
  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′. Go to step 4′ with probability $\min\left(\frac{D_{\mathbb{Z}^{2n},\beta}(\mathbf{z})}{MD_{\mathbb{Z}^{2n},\beta,\mathbf{z}_1}(\mathbf{z})}, 1\right)$, where $\mathbf{z} \in \mathbb{Z}^{2n}$ is the coefficient vector of $\hat{r}_j$ concatenated with the coefficient vector of $\hat{f}_j$, and $\mathbf{z}_1 \in \mathbb{Z}^{2n}$ is the coefficient vector of $s_j d$ concatenated with the coefficient vector of $e_j d$; otherwise go back to step 1′;
  4′. Sample $g_j \leftarrow_r \chi_\beta$, 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 party i^* and the s_j^*-th session of party 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 $\frac{1}{m^2N^2}$.*

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 $G_{1,1}$. $\mathcal{S}$ 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** $d \leftarrow_r \chi_\gamma$, and compute $\hat{r}_j = s_j d + r_j$, $\hat{f}_j = e_j d + f_j$;
3′. Go to step 4′ with probability $\min\left(\frac{D_{\mathbb{Z}^{2n},\beta}(\mathbf{z})}{M D_{\mathbb{Z}^{2n},\beta,\mathbf{z}_1}(\mathbf{z})}, 1\right)$, where $\mathbf{z} \in \mathbb{Z}^{2n}$ is the coefficient vector of $\hat{r}_j$ concatenated with the coefficient vector of $\hat{f}_j$, and $\mathbf{z}_1 \in \mathbb{Z}^{2n}$ is the coefficient vector of $s_j d$ concatenated with the coefficient vector of $e_j d$; otherwise go back to step 1′;
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 $\mathrm{sk}_j = H_2(i, j, x_i, y_j, w_j, \sigma_j)$.

Let $F_{1,l}$ be 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 $(u, \mathbf{b}) \in R_q \times R_q^\ell$ 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 $\mathbf{b} = (b_0 = ur_0 + 2f_0, \ldots, b_{\ell-1} = ur_{\ell-1} + 2f_{\ell-1})$ for some randomly choose $r_{\ell'}, f_{\ell'} \leftarrow_r \chi_\beta$ where $\ell' \in \{0, 1, \ldots, \ell-1\}$, 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 **choose** $\mathbf{z} \leftarrow_r D_{\mathbb{Z}^{2n},\beta}$;
2′. **Parse 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′. **Go to step** 4′ **with probability** $1/M$**; otherwise go back to step** 1′;
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 claim 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** $c \leftarrow_r \chi_\gamma$, and compute $\hat{r}_i = s_i c + r_i$, $\hat{f}_i = e_i c + f_i$;
  3. Go to step 4 with probability $\min\left(\frac{D_{\mathbb{Z}^{2n},\beta}(\mathbf{z})}{M D_{\mathbb{Z}^{2n},\beta,\mathbf{z}_1}(\mathbf{z})}, 1\right)$, where $\mathbf{z} \in \mathbb{Z}^{2n}$ is the coefficient vector of $\hat{r}_i$ concatenated with the coefficient vector of $\hat{f}_i$, and $\mathbf{z}_1 \in \mathbb{Z}^{2n}$ is the coefficient vector of $s_i c$ concatenated with the coefficient vector of $e_i c$; otherwise go back to step 1;
  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 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. Sample an invertible element $c \leftarrow_r \chi_\gamma$, and **choose** $\mathbf{z} \leftarrow_r D_{\mathbb{Z}^{2n},\beta}$;
  2. **Parse 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. **Go to step 4 with probability** $1/M$**; otherwise go back to step 1**;
  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** $\mathrm{sk}_i = \mathrm{sk}_j$, where sk_j is the session key of 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 derives 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 **choose** $\hat{x}_i \leftarrow_r R_q$;
  2. **Define** $x_i = \hat{x}_i - p_i c$;
  3. Go to step 4 with probability $1/M$; otherwise go back to step 1;
  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. **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 [69], 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.*, $\mathrm{sk}_i = H_2(i, j, x_i, y_j, w_j, \sigma_i)$ 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 $\mathrm{RLWE}_{q,\beta}$ 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 **choose** $\hat{y}_j \leftarrow_r R_q$;
  2′. **Define** $y_j = \hat{y}_j - p_j d$;
  3′. Go to step 4′ with probability $1/M$; otherwise go back to step 1′;
  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 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 $\mathrm{RLWE}_{q,\beta}$ 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 $G_{1,6}$. 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. Go to step 4 with probability $1/M$; otherwise go back to step 1;
  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′. Go to step 4′ with probability $1/M$; otherwise go back to step 1′;
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 that in Game $G_{1,6}$ and Game $G_{1,7}$. 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 that in Game $G_{1,6}$ and Game $G_{1,7}$. 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 hold 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 that $\mathcal{A}$ will never make a H_2 query with σ_i.

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 does not make a query with σ_i exactly, the distribution of sk_i is uniform over $\{0,1\}^\kappa$ 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.

5 One-Pass Protocol from Ring-LWE

As MQV [20] and HMQV [5], our AKE protocol has a one-pass variant, which only consists of 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 here are the same as that in Section 3.

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. Go to step 4 with probability $\min\left(\frac{D_{\mathbb{Z}^{2n},\beta}(\mathbf{z})}{MD_{\mathbb{Z}^{2n},\beta,\mathbf{z}_1}(\mathbf{z})}, 1\right)$, where $\mathbf{z} \in \mathbb{Z}^{2n}$ is the coefficient vector of $\hat{r}_i$ concatenated with the coefficient vector of $\hat{f}_i$, and $\mathbf{z}_1 \in \mathbb{Z}^{2n}$ is the coefficient vector of $s_i c$ concatenated with the coefficient vector of $e_i c$; otherwise go back to step 1;
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 $g_j \leftarrow_r \chi_\alpha$, 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 not provide wPFS due to the lack of messages from the receiver j. Besides, the protocol cannot prevent a replay attack without additional measures like keeping a state or using synchronized time. However, we can prove its security in a weak model similar to [5] which avoids the (above) inherent insufficiencies for one-pass protocols. Since the proof is parallel to the two-pass 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-secure encryption scheme in the random oracle model by combining it with a CPA-secure symmetric-key encryption scheme 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 the correctness of our two-pass protocol, the error term must be bounded by $\|\widetilde{g}_i - \widetilde{g}_j\|_\infty < q/8$. Note that $\widetilde{g}_i = (e_j d + f_j)(s_i c + r_i) + d g_i$, and $\widetilde{g}_j = (e_i c + f_i)(s_j d + r_j) + c g_j$, 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 [35]). 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 $\|d g_i\|_\infty \leq 6\gamma\beta\sqrt{n}$ 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 rejection sampling in the protocol. Now, to choose an appropriate β we set $\eta = 1/2$ 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 $\beta \geq \tau \cdot 1/2\alpha\gamma n = 1/2\tau\alpha\gamma n$ 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 $16 \cdot 7\beta^2\sqrt{n} < q$ for correctness of the protocol in Section 3.

Though the Ring-LWE problem enjoys a worst-case connection to some hard problems (*e.g.,* SIVP [29]) 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 [70], which investigates the two most efficient ways to solve the underlying (Ring-)LWE

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	1.5	5.625	5.75
	I_2		75 bits	3.397	24	17.1	47	5.875	1.5	5.875	6.0
	II_1	2048	230 bits	3.397	12	17.1	47	11.75	3.0	11.75	12.0
	II_2		210 bits	3.397	36	18.7	50	12.50	3.0	12.50	12.75
One-pass	III_1	1024	160 bits	3.397	12	16.1	30	3.75	1.5	3.875	-
	III_2		140 bits	3.397	36	17.7	32	4.0	1.5	4.125	-
	IV_1	2048	360 bits	3.397	12	17.1	32	8.0	3.0	8.25	-
	IV_2		350 bits	3.397	36	18.7	33	8.25	3.0	8.5	-

problem, namely the embedding and decoding attacks. As opposed to [70], the decoding attack is more efficient against our instances because the Ring-LWE case with $m \geq 2n$ is close to the optimal attack dimension for the corresponding attacks. The decoding attack first uses a lattice reduction algorithm, such as BKZ [71] / BKZ 2.0 [72] and then applies a decoding algorithm like the ones in [73–75]. Finally, the closest vector is returned as the error polynomial, and the secret polynomial is recovered.

As recommended in [74, 76], 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.[10] In our experiment, we fix $\alpha = 3.397$ for a better performance of the Gaussian sampling algorithm in [39]. 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 in 1000 hash evaluations). In Table 2, 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[11]). 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

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

[11] 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 [77] to give a tighter choice of parameters for desired security levels.

We have implemented 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-thread programming techniques. The program is run on a Dell Optiplex 780 computer with Ubuntu 12.04 TLS 64-bit system, a 2.83GHz Intel Core 2 Quad CPU and 3.8GB RAM. We use an n-dimensional Fast Fourier Transform (FFT) for the multiplications of two ring elements [78, 79], and the CDT algorithm [80] as a tool for hashing to $D_{\mathbb{Z}^n,\gamma}$ and sampling from $D_{\mathbb{Z}^n,\alpha}$, but the DDLL algorithm [39] 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 average timings of each operation (in millisecond, ms) for 1000 executions. Since our protocols also allow some precomputations like sampling Gaussian distributions offline, the timings can be greatly reduced if this is considered in practice. Finally, we note that our implementation has not undergone any real optimization, and it can be much improved in practice.

7 Conclusions and Open Problems

In this paper, a two-pass AKE and its one-pass variant are proposed. Both protocols are carefully built upon on the algebraic structure of (Ring-)LWE problems and several recent developments in lattice-based cryptography, and are proven secure based on the hardness of Ring-LWE in the random oracle model. However, the literature shows that the use of random oracle is delicate for proving quantum resistance [51]. It is of great interest to investigate the quantum security of our protocol, or to design an efficient protocol without the random oracle heuristic (and the need of rewinding).

Acknowledgements. Jiang Zhang and Zhenfeng Zhang are supported by China's 973 program (No. 2013CB338003) and the National Natural Science Foundation of China (No. 61170278, 91118006). Jintai Ding is partially supported by the Charles Phelps Taft fund. Özgür Dagdelen is supported by the German Federal Ministry of Education and Research (BMBF) within EC-SPRIDE and by the DFG as part of project P1 within the CRC 1119 CROSSING. We would like to thank Johannes Buchmann, Lily Chen, Oded Regev, Adi Shamir, Tsuyoshi Takagi and Xiang Xie for useful discussions, and the anonymous reviewers of EUROCRYPT 2015 for helpful comments and suggestions.

References

1. Diffie, W., Hellman, M.: New directions in cryptography. Information Theory, IEEE Transactions on **22** (1976) 644 – 654
2. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: CRYPTO. Volume 773. (1994) 232–249
3. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: EUROCRYPT. (2001) 453–474
4. LaMacchia, B.A., Lauter, K.E., Mityagin, A.: Stronger security of authenticated key exchange. In: ProvSec. (2007) 1–16
5. Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In: CRYPTO. (2005) 546–566

6. Cremers, C., Feltz, M.: Beyond eck: Perfect forward secrecy under actor compromise and ephemeral-key reveal. In: ESORICS. (2012) 734–751
7. Harkins, D., Carrel, D., et al.: The internet key exchange (IKE). Technical report, RFC 2409, november (1998)
8. Kaufman, C., Hoffman, P., Nir, Y., Eronen, P.: Internet key exchange protocol version 2 (IKEv2). Technical report, RFC 5996, September (2010)
9. Krawczyk, H.: SIGMA: The 'SIGn-and-MAc' approach to authenticated Diffie-Hellman and its use in the IKE protocols. In: CRYPTO. (2003) 400–425
10. Freier, A.: The SSL protocol version 3.0. http://wp. netscape. com/eng/ssl3/draft302. txt (1996)
11. Dierks, T.: The transport layer security (TLS) protocol version 1.2. (2008)
12. Krawczyk, H., Paterson, K., Wee, H.: On the security of the TLS protocol: A systematic analysis. In: CRYPTO. (2013) 429–448
13. Mavrogiannopoulos, N., Vercauteren, F., Velichkov, V., Preneel, B.: A cross-protocol attack on the TLS protocol. In: CCS. (2012) 62–72
14. Giesen, F., Kohlar, F., Stebila, D.: On the security of TLS renegotiation. In: CCS. (2013) 387–398
15. Brzuska, C., Fischlin, M., Smart, N.P., Warinschi, B., Williams, S.C.: Less is more: relaxed yet composable security notions for key exchange. Int. J. Inf. Sec. **12** (2013) 267–297
16. Dagdelen, Ö., Fischlin, M.: Security analysis of the extended access control protocol for machine readable travel documents. In: ISC. (2010) 54–68
17. Dagdelen, Ö., Fischlin, M., Gagliardoni, T., Marson, G.A., Mittelbach, A., Onete, C.: A cryptographic analysis of OPACITY - (extended abstract). In: ESORICS. (2013) 345–362
18. Degabriele, J.P., Fehr, V., Fischlin, M., Gagliardoni, T., Günther, F., Marson, G.A., Mittelbach, A., Paterson, K.G.: Unpicking PLAID. In: Security Standardisation Research. (2014) 1–25
19. Matsumoto, T., Takashima, Y.: On seeking smart public-key-distribution systems. IEICE TRANSACTIONS (1976-1990) **69** (1986) 99–106
20. Menezes, A., Qu, M., Vanstone, S.: Some new key agreement protocols providing mutual implicit authentication. In: SAC. (1995) 22–32
21. ANS X9.42-2001: Public key cryptography for the financial services industry: Agreement of symmetric keys using discrete logarithm cryptography. (2001)
22. ANS X9.63-2001: Public key cryptography for the financial services industry: Key agreement and key transport using elliptic curve cryptography. (2001)
23. ISO/IEC: 11770-3:2008 information technology – security techniques – key management – part 3: Mechanisms using asymmetric techniques (2008)
24. IEEE 1363: IEEE std 1363-2000: Standard specifications for public key cryptography. IEEE, august 2000. (2000)
25. Barker, E., Chen, L., Roginsky, A., Smid, M.: Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography. NIST Special Publication **800** (2013) 56A
26. Yao, A.C.C., Zhao, Y.: OAKE: A new family of implicitly authenticated Diffie-Hellman protocols. In: CCS. (2013) 1113–1128
27. Shor, P.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing **26** (1997) 1484–1509
28. Chen, L.: 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`.
29. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: EUROCRYPT. (2010) 1–23

30. Ducas, L., Durmus, A.: Ring-LWE in polynomial rings. In: PKC. (2012) 34–51
31. Blake-Wilson, S., Johnson, D., Menezes, A.: Key agreement protocols and their security analysis. In: Proceedings of the 6th IMA International Conference on Cryptography and Coding, London, UK, UK, Springer-Verlag (1997) 30–45
32. Brzuska, C., Fischlin, M., Warinschi, B., Williams, S.C.: Composability of Bellare-Rogaway key exchange protocols. In: CCS. (2011) 51–62
33. Ding, J., Xie, X., Lin, X.: A simple provably secure key exchange scheme based on the learning with errors problem. Cryptology ePrint Archive, Report 2012/688 (2012)
34. Peikert, C.: Lattice cryptography for the internet. In: Post-Quantum Cryptography. (2014) 197–219
35. Bos, J.W., Costello, C., Naehrig, M., Stebila, D.: Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. Cryptology ePrint Archive, Report 2014/599 (2014)
36. Goldwasser, S., Kalai, Y.T., Peikert, C., Vaikuntanathan, V.: Robustness of the learning with errors assumption. In: Innovations in Computer Science. (2010) 230–240
37. Lyubashevsky, V.: Lattice signatures without trapdoors. In Pointcheval, D., Johansson, T., eds.: EUROCRYPT. (2012) 738–755
38. Güneysu, T., Lyubashevsky, V., Pöppelmann, T.: Practical lattice-based cryptography: A signature scheme for embedded systems. In: CHES. (2012) 530–547
39. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal Gaussians. In: CRYPTO. (2013) 40–56
40. Bai, S., Galbraith, S.D.: An improved compression technique for signatures based on learning with errors. In: CT-RSA. (2014) 28–47
41. Hoffstein, J., Pipher, J., Schanck, J.M., Silverman, J.H., Whyte, W.: Practical signatures from the partial fourier recovery problem. In: ACNS. (2014) 476–493
42. Katz, J., Vaikuntanathan, V.: Smooth projective hashing and password-based authenticated key exchange from lattices. In: ASIACRYPT. (2009) 636–652
43. Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Strongly secure authenticated key exchange from factoring, codes, and lattices. In: PKC. (2012) 467–484
44. Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Practical and post-quantum authenticated key exchange from one-way secure key encapsulation mechanism. In: ASIACCS. (2013) 83–94
45. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: STOC. (2008) 187–196
46. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In: STOC. (2009) 333–342
47. Canetti, R., Krawczyk, H.: Security analysis of IKEs signature-based key-exchange protocol. In: CRYPTO. (2002) 143–161
48. Jager, T., Kohlar, F., Schäge, S., Schwenk, J.: On the security of TLS-DHE in the standard model. In: CRYPTO. (2012) 273–293
49. Watrous, J.: Zero-knowledge against quantum attacks. SIAM J. Comput. **39** (2009) 25–58
50. Unruh, D.: Quantum proofs of knowledge. In: EUROCRYPT 2012. (2012) 135–152
51. Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C., Zhandry, M.: Random oracles in a quantum world. In: ASIACRYPT 2011. (2011) 41–69
52. Dagdelen, Ö., Fischlin, M., Gagliardoni, T.: The Fiat-Shamir transformation in a quantum world. In: ASIACRYPT 2013, Part II. (2013) 62–81
53. Ambainis, A., Rosmanis, A., Unruh, D.: Quantum attacks on classical proof systems (the hardness of quantum rewinding). In: FOCS 2014, IEEE (2014) 474–483
54. Boneh, D., Zhandry, M.: Secure signatures and chosen ciphertext security in a quantum computing world. In: CRYPTO. (2013) 361–379

55. Song, F.: A note on quantum security for post-quantum cryptography. In: Post-Quantum Cryptography. (2014) 246–265
56. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on gaussian measures. SIAM J. Comput. **37** (2007) 267–302
57. Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: CRYPTO. (2009) 595–618
58. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from Ring-LWE and security for key dependent messages. In: CRYPTO. (2011) 505–524
59. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: Fully homomorphic encryption without bootstrapping. Innovations in Theoretical Computer Science, ITCS (2012) 309–325
60. Chor, B., Goldreich, O.: Unbiased bits from sources of weak randomness and probabilistic communication complexity. In: FOCS. (1985) 429–442
61. Trevisan, L., Vadhan, S.: Extracting randomness from samplable distributions. In: FOCS. (2000) 32–42
62. Trevisan, L.: Extractors and pseudorandom generators. J. ACM **48** (2001) 860–879
63. Dodis, Y., Gennaro, R., Håstad, J., Krawczyk, H., Rabin, T.: Randomness extraction and key derivation using the CBC, Cascade and HMAC modes. In: CRYPTO. (2004) 494–510
64. Barak, B., Impagliazzo, R., Wigderson, A.: Extracting randomness using few independent sources. SIAM Journal on Computing **36** (2006) 1095–1118
65. Barker, E., Roginsky, A.: Recommendation for the entropy sources used for random bit generation. (Draft NIST Special Publication 800-90B, August 2012.)
66. Stehlé, D., Steinfeld, R.: Making NTRU as secure as worst-case problems over ideal lattices. In: EUROCRYPT. (2011) 27–47
67. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: CCS. (1993) 62–73
68. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: CCS. (2006) 390–399
69. Gennaro, R., Shoup, V.: A note on an encryption scheme of Kurosawa and Desmedt. Cryptology ePrint Archive, Report 2004/194 (2004)
70. Dagdelen, O., Bansarkhani, R.E., Göpfert, F., Güneysu, T., Oder, T., Pöppelmann, T., Sánchez, A.H., Schwabe, P.: High-speed signatures from standard lattices. In: LATINCRYPT. (2014)
71. Schnorr, C., Euchner, M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Mathematical Programming **66** (1994) 181–199
72. Chen, Y., Nguyen, P.: BKZ 2.0: Better lattice security estimates. In: ASIACRYPT. (2011) 1–20
73. Babai, L.: On Lovász' lattice reduction and the nearest lattice point problem. Combinatorica **6** (1986) 1–13
74. Lindner, R., Peikert, C.: Better key sizes (and attacks) for LWE-based encryption. In: CT-RSA. (2011) 319–339
75. Liu, M., Nguyen, P.: Solving bdd by enumeration: An update. In: CT-RSA. (2013) 293–309
76. Gentry, C., Halevi, S., Smart, N.: Homomorphic evaluation of the AES circuit. In: CRYPTO. (2012) 850–867
77. Lyubashevsky, V., Peikert, C., Regev, O.: A toolkit for Ring-LWE cryptography. In: EUROCRYPT. (2013) 35–54
78. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C., et al.: Introduction to algorithms. Volume 2. MIT press Cambridge (2001)
79. Lyubashevsky, V., Micciancio, D., Peikert, C., Rosen, A.: SWIFFT: A modest proposal for FFT hashing. In: FSE. (2008) 54–72
80. Peikert, C.: An efficient and parallel Gaussian sampler for lattices. In: CRYPTO. (2010) 80–97