A Modular and Systematic Approach to Key Establishment and Public-Key Encryption Based on LWE and Its Variants[*]

Principal Submitter: Yunlei Zhao
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai 201203, China.
Telephone: 86-13564812886
Email: ylzhao@fudan.edu.cn
Auxiliary submitters: Zhengzhong Jin, Boru Gong, Guangye Sui

Abstract

In this work, we abstract some key ingredients in previous key exchange protocols based on LWE and its variants, by introducing and formalizing the building tool, referred to as key consensus (KC) and its asymmetric variant AKC. KC and AKC allow two communicating parties to reach consensus from close values obtained by some secure information exchange. We then discover upper bounds on parameters for any KC and AKC. KC and AKC are fundamental to lattice based cryptography, in the sense that a list of cryptographic primitives based on LWE and its variants (including key exchange, public-key encryption, and more) can be modularly constructed from them. As a conceptual contribution, this much simplifies the design and analysis of these cryptosystems in the future.

We then design and analyze both general and highly practical KC and AKC schemes, which are referred to as OKCN and AKCN respectively for presentation simplicity. Based on KC and AKC, we present generic constructions of key exchange (KE), public-key encryption (PKE), and authenticated key exchange (AKE) from LWR, LWE, RLWE and MLWE. The generic constructions allow versatile instantiations with our OKCN and AKCN schemes, for which we elaborate on evaluating and choosing the concrete parameters in order to achieve a well-balanced performance among security, computational cost, bandwidth efficiency, error rate, and operation simplicity.

[*]Preliminary version appears at arXiv: https://arxiv.org/abs/1611.06150.

Contents

1 Introduction

Most public-key cryptosystems currently in use, based on the hardness of solving (elliptic curve) discrete logarithm or factoring large integers, will be broken, if large-scale quantum computers are ever built. The arrival of such quantum computers is now believed by many scientists to be merely a significant engineering challenge, and is estimated by engineers at IBM to be within the next two decades or so. Historically, it has taken almost two decades to deploy the modern public key cryptography infrastructure. Therefore, regardless of whether we can estimate the exact time of the arrival of the quantum computing era, we must begin now to prepare our information security systems to be able to resist quantum computing. In addition, for the content we want to protect over a period of 15 years or longer, it becomes necessary to switch to post-quantum cryptography today. This has been recognized not only by the cryptography research community, but also by standardization bodies and leading information companies. As noted in [ADPS16, AJS16], in the majority of contexts the most critical asymmetric primitive to upgrade to post-quantum security is ephemeral key exchange (KE).

Lattice-based cryptography is among the major mathematical approaches to achieving security resistant to quantum attacks. For cryptographic usage, compared with the classic hard lattice problems such as SVP and CVP, the learning with errors (LWE) problem is proven to be much more versatile [Reg09]. Nevertheless, LWE-based cryptosystems are usually less efficient, which was then resolved by the introduction of the ring-LWE (RLWE) problem [LPR10]. In recent years, large numbers of impressive works are developed from LWE and RLWE, with (ephemeral) key exchange and public-key encryption being the study focus of this work [JD12, Pei14, BCNS15, ADPS16, BCD$^+$16, Reg09, GPV08, LP11, LPR10, LPR13b, PG13]. For an excellent survey of lattice-based cryptography, the reader is referred to [Pei16].

Some celebrating progresses on achieving practical LWE- and RLWE-based ephemeral key exchange are made in recent years. The performance of RLWE-based key exchange is significantly improved with NewHope [ADPS16], which achieves 256-bit shared-key with error rate about 2^{-61}. The negligible error rate of NewHope is achieved by decoding the four-dimensional lattice $\tilde{D}_4$. Compared to LWE, the additional ring structure of RLWE helps to improve the efficiency of cryptosystems, but the concrete hardness of RLWE remains less clear. The work [BCD$^+$16] proposes a key exchange protocol Frodo only based on LWE, and demonstrates that LWE-based key exchange can be practical as well. Nevertheless, bandwidth of Frodo is relatively large, as Frodo uses about 22kB bandwidth for its recommended parameter set. In addition, Frodo has relatively large error rates, and cannot be directly used for PKE. Whether further improvements on LWE- and RLWE-based key exchange can be achieved remains an interesting question of practical significance.

One of the main technical contributions in the works [ADPS16, BCD$^+$16, PG13], among others, is the improvement and generalization of the key reconciliation mechanisms [Pei14, JD12].[1] But the key reconciliation mechanisms were only previously used and analyzed, for both KE and PKE, in a *non-black-box* way. This means, for new key reconciliation mechanisms developed in the future to be used for constructing lattice-based cryptosystems, we need to analyze the security from scratch. Also, for the various parameters involved in key reconciliation, the bounds on what could or couldn't be achieved are unclear.

[1]To our knowledge, the key reconciliation mechanism in [Pei14] is the first that fits our KC definition (the mechanism in [JD12] requires the distance be of special types).

1.1 Our Contributions

In this work, we abstract some key ingredients in previous LWE- and RLWE-based key exchange protocols, by introducing and formalizing the building tool, referred to as key consensus (KC) and its asymmetric variant AKC. KC and AKC allow two communicating parties to reach consensus from close values obtained by some secure information exchange, such as exchanging their LWE/RLWE samples. KC and AKC are fundamental to lattice based cryptography, in the sense that a list of cryptographic primitives based on LWE or RLWE (including authenticated key exchange, public-key encryption, and more) can be modularly constructed from them. As a conceptual contribution, this much simplifies the design and analysis of these cryptosystems in the future, by allowing for modular and black-box design and analysis with KC and AKC.

Abstracting KC and AKC also allows us to study and prove the inherent upper-bounds among the parameters. In particular, we discover the upper bounds on parameters for any KC and AKC. This allows us to understand what can or cannot be achieved with any KC and AKC, and guides our actual protocol design. These upper-bounds also guide parameter choosing for various trade-offs, and are insightful in comparing the performance of KC vs. AKC.

Guided by, and motivated for reaching, these proved upper-bounds, we then design and analyze both general and highly practical KC and AKC schemes, which are referred to as OKCN and AKCN respectively for presentation simplicity.

Based on KC and AKC, we present generic constructions of key exchange from LWR, LWE, RLWE and MLWE with delicate analysis of error rates. Then, for the instantiations of these generic constructions with our OKCN and AKCN schemes, we elaborate on evaluating and choosing the concrete parameters in order to achieve a well-balanced performance among security, computational efficiency, bandwidth efficiency, error rate, and operation simplicity.

- We propose the first construction of key exchange *merely* based on the LWR problem with concrete analysis and evaluation, to the best of our knowledge. In particular, we provide a delicate approach to calculating its error rate.

  Specifically, for the LWR-based KE protocol, the main difficulty here is the error probability analysis: the rounding operation in LWR brings new noises, yet these noises are *deterministic*, because they are completely determined by the public matrix $\mathbf{A}$ and the secret vector $\mathbf{S}$. In the formula calculating the error probability, the deterministic noises will multiply the secret $\mathbf{S}$. However they are correlated. This correlation prevents us from calculating the error probability efficiently. Note that, in the LWE-based KE, the noises are independent of $\mathbf{A}$ and the secret vector $\mathbf{S}$. So this is a new difficulty we encounter in LWR-based KE. Our contribution is to provide an analysis breaking the correlation, and design an algorithm to calculate the error probability numerically.

  A salient feature of LWR-based key exchange protocols is their bandwidth efficiency, for instance, about 16.19kB at the level of at least 128-bit quantum security (in the sense of resistance against the best known quantum attacks).

- When applied to LWE-based cryptosystems, OKCN can directly result in more practical or well-balanced schemes of key exchange. To further save bandwidth, we make a thorough analysis of the variant where some least significant bits of LWE samples are chopped off, which results in, for instance, 18.58kB bandwidth at the level of at least 128-bit quantum security. Chopping off some least bits of LWE samples can only improve the actual security guarantee in reality, but complicates the analysis of error rates.

	$\lvert\mathbf{K}\rvert$	bw.(kB)	err.	pq-sec
OKCN-LWR	256	16.19	2^{-30}	130
OKCN-LWE	256	18.58	2^{-39}	134
Frodo	256	22.57	$2^{-38.9}$	130

Table 1: Brief comparison between OKCN-LWE/LWR and Frodo. $\lvert\mathbf{K}\rvert$ refers to the size in bits of the shared key; "bw.(kB)" refers to bandwidth in kilo bytes; "err." refers to the error rate, and "pq-sec" refers to the best known quantum attack against the underlying lattice problem.

- When applied to RLWE-based cryptosystems, to the best of our knowledge, AKCN can lead to the most efficient KE protocols with shared-key of size of at least 512 bits (which may be prudent for ensuring 256-bit post-quantum security in reality). We first use the technique of NewHope to further lower the error rate, by decoding the four-dimensional lattice $\tilde{D}_4$, but at the price of achieving only 256-bit shared key.

  We then develop new approaches to lower the error rate of RLWE-based KE for achieving shared key of size at least 512 bits. Firstly, we make a key observation on RLWE-based key exchange, by proving that the errors in different positions in the shared-key are almost independent. Here, the main problem prevent us from using Central Limit Theorem (CLT) is that any two different coefficients of the product of two Gaussian polynomials are correlated. Note that the classical CLT requires the random variables to be independent. While some variants of CLT allow the random variables to be correlated (such as CLT extended to a stochastic process), but these variants cannot fit into our situation as far as we know. We are unaware of any existing techniques in dealing with this problem precisely. But we do need a result to deal with arbitrary two different coefficients, because when using error correcting code any two different coordinates may have errors. We prove that, even when coefficients are correlated, we can still have result similar to CLT. Though it is only an asymptotic result, it at least provides an indication or justification for the reasonability, and can play a fundamental basis for the approach to lower error rate of RLWE-based KE with error-correction codes. We note that, in some related (concurrent and subsequent) works on RLWE-based KE using error-correction codes, it is simply assumed the errors are independent without any argument. Also, for some related works on KE from RLWE and MLWE where some least significant bits of RLWE/MLWE samples are cut off, it is also similarly assumed that these least significant bits are uniform at random.

  Then, based upon this observation, we present a super simple and fast code, referred to as *single-error correction* (SEC) code, to correct at least one bit error. By equipping OKCN/AKCN with the SEC code, we achieve the simplest (up to now) RLWE-based key exchange, from both OKCN and AKCN, with negligible error rate for much longer shared-key size; for instance, OKCN-based implementation for 765-bit shared-key with bandwidth of 3136 bytes at error rate $2^{-68.4}$ and about 250-bit post-quantum security, and AKCN-based implementation for 765-bit shared-key with bandwidth of 3392 bytes at error rate $2^{-54.4}$ and about 258-bit post-quantum security.

  To further improve the bandwidth, error rate and post-quantum security simultaneously, we develop new lattice code in E_8, based on which we achieve AKCN-based KE for 512-bit shared-key with bandwidth of 3360 bytes at error rate $2^{-63.3}$ and about 262-bit post-quantum security. Note that sphere packing is optimal with the lattice E_8.

- Finally, when applying OKCN/AKCN to MLWE-based KE, they result in the (up-to-date) most efficient lattice-based key exchange protocols for 256-bit shared-key. Moreover, as noted in [BDK+17], MLWE-based implementations are very flexible and versatile, for instance, the recommended (resp., light) implementation of KE has bandwidth 1856 (resp., 1312) bytes at error rate $2^{50.1}$ (resp., $2^{-36.2}$) and 183-bit (resp., 116) post-quantum security.

  We then present applications to CCA-secure PKE, and to privacy-preserving authenticated key-exchange (AKE). In particularly, we design a new AKE scheme, referred to as *concealed non-malleable key-exchange* (CNKE).

	$\lvert\mathbf{K}\rvert$	bw.(B)	err.	pq-sec
OKCN-RLWE-SEC-1	765	3136	$2^{-68.4}$	250
OKCN-RLWE-SEC-2	765	3392	2^{-61}	258
NewHope	256	3872	2^{-61}	255
AKCN-RLWE-SEC-1	765	3264	$2^{-68.4}$	250
AKCN-RLWE-SEC-2	765	3520	2^{-61}	258
AKCN-RLWE-E8	512	3360	$2^{-63.3}$	262
NewHope-Simple	256	4000	2^{-61}	255

Table 2: Brief comparison between OKCN/AKCN-RLWE and NewHope.

	$\lvert\mathbf{K}\rvert$	bw.(B)	err.	pq-sec
OKCN-MLWE-KE	256	1856	$2^{-50.1}$	183
OKCN-MLWE-PKE-1	256	1952	$2^{-80.3}$	183
OKCN-MLWE-PKE-2	256	2048	$2^{-166.4}$	171
AKCN-MLWE-PKE (Kyber)	256	2272	$2^{-142.7}$	171

Table 3: Brief comparison between OKCN/AKCN-MLWE and Kyber.

All the main protocols developed in this work are implemented. The code and scripts, together with those for evaluating concrete security and error rates, for protocols based on LWE, LWR and RLWE are also available from Github `http://github.com/OKCN`. Besides theoretical analysis, much efforts in this work were also put on implementation and concrete evaluation.

1.2 Applications to KEM, PKE and AKE

KC-based KE protocol can be viewed as the equivalent of traditional Diffie-Hellman. It means that, as discussed in [Pei14], it can be transformed into an authenticated key exchange (AKE) protocol via the SIGMA mechanism [Kra03], and is well suitable to be integrated into more advanced protocols like IKE and TLS. As discussed in [Pei14], KC-based KE protocol can in turn be transformed into a CCA-secure key-encapsulation mechanism (KEM) via the FO-transformation and its variants [FO13, AGKS05, Pei14, TU16]. In this work, we also explicitly present CCA-secure KEM construction based on OKCN-MLWE, which is instantiated from [HHK17].

AKC-based KE protocol is actually a key transport protocol, which directly yields CPA-secure KEM (and CCA-secure KEM via the FO-transformation). A concrete MLWE-based CCA-secure KEM from our AKCN is presented in [BDK+17], by using a specific variant of the FO-transformation proposed in [HHK17]. Following the generic paradigm for achieving AKE from public-key encryption, a concrete AKE protocol based on KEM is also proposed in [HHK17]. In this work, we present a CCA-secure PKE scheme, with instantiation from AKCN-MLWE, by combining the techniques in [FO13, D02, TU16, HHK17], which aims for: (1) compatibility with existing standards; (2) flexibility when being used for AKE; (3) resistance to side-channel attacks.

We then design a new AKE scheem, referred to as concealed non-malleable key-exchange (CNKE). CNKE does not use signatures as the underlying authentication mechanism, and is carefully designed to enjoy the following advantages:

- Computational efficiency: By replacing CCA-secure KEM in existing constructions with an ephemeral key exchange/transport protocol, it is computationally more efficient, and is more applicable to client/server setting with low-power clients.
- Robust resistance to man-in-the-middle (MIM) malleating attacks, to secrecy exposure, and to side-channel attacks.
- Privacy protection: Client's identity information, as well as the components of the underlying ephemeral key exchange/transport protocol, are encrypted. Identity privacy is deemed to be an important privacy issue, and is mandated by some prominent standards like TLS1.3, EMV, etc. Concealing the components of the ephemeral key exchange/transport protocol not only strengthens security, but is also useful for privacy protection.
- Well compatibility with with TLS1.3. CNKE explicitly uses authenticated encryption (that is mandated by TLS1.3), and uses the Finish mechanism of TLS1.3 for mutual authentications.

For all the OKCN/AKCN-based KE protocols developed in this work, when they are used for KEM or PKE, the first-round message from the initiator corresponds to the public key.

1.3 Advantages and Disadvantages of OKCN vs. AKCN

Above all, with OKCN and AKCN, we provide a general framework for achieving key exchange and public-key encryption from lattice (specifically, LWE and its variants: LWR, RLWE, MLWE), *in a systemized and modular way.* Secondly, we provide a set of practical yet powerful tools for dealing with noise: OKCN, AKCN, single-error correction code and lattice code in E_8, which we suggest may play a basic role in the future design and analysis of cryptographic schemes from LWE and its variants. Also, to the best of our knowledge, AKC-based key exchange (actually, key transport) was firstly formalized in this work.

But cryptosystems based upon OKCN and AKCN have different performances and features in different settings.

- OKCN-based KE can be viewed as the equivalent of Diffie-Hellman in the lattice world, while AKCN-based KE is not. Specifically, with AKCN, the responder can predetermine and set the shared-key at its wish. But AKCN can be directly used for CPA-secure KEM.
- It is well recognized that monoculture is bad for security, and that AKE protocol via the SIGMA mechanism takes advantages over PKE-based AKE (e.g., symmetry, post-ID, privacy, modular and diversified deployments, etc). For instance, the first generation of IKE is based on PKE, but the second generation moves to SIGMA-based AKE.

- OKCN-based KE is more versatile, and is more appropriate for incorporating into the existing standards like IKE and TLS that are based on Diffie-Hellman via the SIGMA mechanism. In view that OKCN is better suitable for incorporating into IKE and TLS, it should be more desirable to employ the same OKCN mechanism for public-key encryption, for the sake of system simplicity and easy deployment.

- On the same parameters (q, m, g) as specified in Section 3 and 4 (which implies the same bandwidth), OKCN-based KE has lower error rate than AKCN-based KE. Or, on the same parameters (q, m, d) (which implies the same error rate), OKCN-based KE has smaller bandwidth than AKCN-based KE. This comparison is enabled by the upper-bounds on these parameters developed in Section 3 and 4.

- Similarly, on the same parameters (q, m, g) (which implies the same bandwidth), OKCN-based KEM has lower error rate than AKCN-based KEM. On the same parameters (q, m, d) (which implies the same error rate), the bandwidth of OKCN-based KEM is at least as good as that of AKCN-based KEM.[2]

- For KE of 256-bit shared-key, OKCN/AKCN-MLWE is the most efficient. But for KE with shared-key of size 512 bits or more (which might be necessary for ensuring 256-bit post-quantum security in reality), OKCN/AKCN-RLWE is the most efficient.

- Compared to RLWE and MLWE, the LWE and LWR problems have fewer algebraic structures that can be exploited by attacks. As noise sampling is relatively cumbersome for lattice-based cryptography, LWR-based KE may be more desirable in this sense.

1.4 On Novelty of OKCN and AKCN

To the best of our knowledge, the formulations of KC and AKC, their necessary properties for CPA-secure KEM, and their upper-bounds on the various parameters, are first (explicitly) presented in this work, which much simplify the future design and analysis of KE and PKE based on LWE and its variants. Indeed, both the design of OKCN and that of AKCN were guided by, and motivated for reaching, the upper-bounds for KC and AKC proved in this work.

To the best of our knowledge, OKCN is the first multi-bit reconciliation mechanism, and the first that can be instantiated to tightly match the upper-bound, which is the source for essentially outperforming Frodo.

The design of AKCN was guided by, and motivated for, the upper-bound for AKC proved in this work. In designing AKCN, we combine all existing optimizations in the literature in order to almost meet the proved upper-bound. AKCN is clearly a generalization of the basic reconciliation mechanisms proposed in [LPR10, LP11], and its design was also inspired by the design of our OKCN and the works [BPR12, PG13].[3] In particular, the reconciliation mechanisms proposed in [LPR10, LP11] correspond to the special case of AKCN when $g = q$ and $m = 2$. Note that, with AKCN, we use Equation 1 described in Section 2.4, which was explicitly proposed in [BPR12] for forming LWR samples and may also be derived implicitly from [Pei09].

[2]Specifically, as shown in this work, for KEM with 256-bit shared-key, the bandwidth of OKCN-based KEM is at least as good as that of AKCN-based KEM. But if the shared-key is of size 512 bits or more (e.g., to ensure 256-bit post-quantum security targeting the underlying shared-key in reality), OKCN-based KEM can have a smaller bandwidth.

[3]But AKCN and the underlying reconciliation mechanism of [PG13] could be viewed as incomparable in general.

Briefly speaking, the novelty of AKCN lies in two aspects: (1) the combination of the basic reconciliation mechanism from [LPR10, LP11] and the rounding technique from [BPR12, Pei09] in the Con procedure is first explicitly presented in this work, where the Rec procedure is less straightforward in this case; (2) multi-bit reconciliation with generalized m. Here, the bottom line is that the exact formula of AKCN, even for the special case of $m = 2$ (that is just the underlying reconciliation mechanism of CPA-secure Kyber [BDK+17]), couldn't be directly instantiated from any existing *single* work.

1.5 Recommended Algorithms

Our work can serve as a general framework for understanding and evaluating the various proposals for KEM schemes from LWE and its variants. With this submission, due to time limitation and intellectual property issues of implementation codes,[4] we only implemented the following five algorithms that are recommended for standardization considerations.

AKCN-SEC: It is an RLWE-based ephemeral KEM, which is specified in Section 8.3.3. AKCN-SEC uses AKCN equipped with the single-error correction (SEC) code. Its security lies in Category-5.

OKCN-SEC: It is an ephemeral KEM scheme based on RLWE, which is specified in Section 8.3.4. OKCN-SEC uses OKCN equipped with the SEC code. Its security lies in Category-5.

OKCN-MLWE: It is an OKCN-based ephemeral KEM from MLWE, which is specified in Section 9.1.1. Its security lies in Category-4.

AKCN-MLWE: It is an AKCN-based ephemeral KEM from MLWE, which is specified in Section 9.1.2. Its security lies in Category-4.

AKCN-MLWE-CCA: It is an AKCN-based CCA-secure public-key encryption from MLWE, which is specified in Section 9.4. Its security lies in Category-4.

1.6 Other Algorithms for Considerations

As KEM schemes from LWE and its variants are reduced to the corresponding KC and AKC mechanisms, and as our OKCN and AKCN are almost optimal, the following KEM algorithms are also of competitive performance or can serve as the benchmarks for evaluation and comparisons.

OKCN/AKCN-LWR: Specifically, the ephemeral KEM schemes based on OKCN and AKCN from LWR, as specified in Section 5. Their security lies in Category-3.

OKCN/AKCN-LWE: Specifically, the ephemeral KEM schemes based on OKCN and AKCN from LWE, as specified in Section 6. Their security lies in Category-3.

OKCN/AKCN-RLWE: Specifically, the ephemeral KEM schemes based on OKCN and AKCN from RLWE, as specified in Figure 10 and 11 in Section 8. Their security lies in Category-5.

[4]We understand that, according to the call-for-proposals of NIST, the submitters should have full intellectual properties for the implementation codes they submit.

All the above protocols are also implemented, and their codes and scripts are available from Github `http://github.com/OKCN`. As these implementations share some codes from Frodo and NewHope, we did not submit them to NIST.

In addition, the following two algorithms have special desirable features and performance. But they have not been implemented yet, due to time limitation. We will provide the implementations of them in the future.

AKCN-E8: It is an RLWE-based ephemeral KEM scheme specified in Section 8.4.1, which uses the developed lattice code in E_8 to reduce the error probability. It can have better performance compared to AKCN-SEC, at the price of increasing the complexity a little. Its security lies in Category-5.

CNKE: It is an authenticated key exchange protocol described in Section 9.5. CNKE is computationally efficient, and is well compatible with existing standards like TLS in the client/server setting. Its MLWE-based construction is described in Section 9.5.4, with security level lying in Category-4. The construction can also be straightforwardly adapted to those based on LWE, LWR and RLWE.

1.7 Concurrent and Subsequent Work

The work [CKLS16] proposes a CPA-secure PKE scheme, named Lizard, based both on a variant of LWE (referred to as spLWE) and on a variant of LWR (referred to as spLWR). Specifically, for Lizard, the public key is generated with spLWE sample, while the ciphertext is generated with spLWR sample. The underlying reconciliation mechanism of Lizard can be viewed as a special case of AKCN for $m|g|q$, where g (resp., m) in AKCN corresponds to p (resp., t) in [CKLS16]. Also, we do not know how to apply the analysis of Lizard to KE protocols merely based on LWR, as analyzed in Section 6 where both public key and ciphertext are generated merely from LWR samples.

To the best of our knowledge, AKC-based key exchange (actually, key transport) was firstly formalized in this work. In particular, AKCN4:1 is the first AKC-based variant of NewHope. Another AKC-based variant of NewHope, named NewHope-simple, was presented subsequently in a short note posted on 17 December 2016 [ADPS16b]. In comparison, NewHope-simple is still slightly inferior to AKCN4:1-RLWE in bandwidth expansion (specifically, 256 vs. 1024 bits).

Recently, a module lattice based CPA-secure KEM scheme, named Kyber, was introduced [BDK+17]. Though different notations and presentation methods are used in [BDK+17], it is easy to see that the underlying AKC mechanism of Kyber (specifically, Line 6 of Algorithm 2 in [BDK+17]) is just our AKCN scheme. Specifically, by letting $\sigma_1 = t^T r + e_2$, $m = 2$ and $g = 2^{d_v}$, the resultant instantiation of AKCN is actually the underlying AKC mechanism implicitly used in [BDK+17]. In particular, when setting $d_t = d_u = 13$ and $k = 1$ (corresponding to $t_1 = t_2 = 0$ and $l = 1$ in our case), Kyber is actually AKCN-RLWE that is explicitly specified in this work.

2 Preliminaries

A string or value α means a binary one, and $|\alpha|$ is its binary length. For any real number x, $\lfloor x \rfloor$ denotes the largest integer that less than or equal to x, and $\lfloor x \rceil = \lfloor x + 1/2 \rfloor$. For any positive integers a and b, denote by $\mathsf{lcm}(a, b)$ the least common multiple of them. For any $i, j \in \mathbb{Z}$ such that $i < j$, denote by $[i, j]$ the set of integers $\{i, i+1, \cdots, j-1, j\}$. For any positive integer t, we let $\mathbb{Z}_t$ denote $\mathbb{Z}/t\mathbb{Z}$. The elements of $\mathbb{Z}_t$ are represented, by default, as $[0, t-1]$. Nevertheless, sometimes, $\mathbb{Z}_t$ is explicitly specified to be represented as $[-\lfloor (t-1)/2 \rfloor, \lfloor t/2 \rfloor]$.

If S is a finite set then $|S|$ is its cardinality, and $x \leftarrow S$ is the operation of picking an element uniformly at random from $\mathcal{S}$. For two sets $A, B \subseteq \mathbb{Z}_q$, define $A + B \triangleq \{a + b | a \in A, b \in B\}$. For an addictive group $(G, +)$, an element $x \in G$ and a subset $S \subseteq G$, denote by $x + S$ the set containing $x + s$ for all $s \in S$. For a set S, denote by $\mathcal{U}(S)$ the uniform distribution over S. For any discrete random variable X over $\mathbb{R}$, denote $\mathsf{Supp}(X) = \{x \in \mathbb{R} \mid \Pr[X = x] > 0\}$.

We use standard notations and conventions below for writing probabilistic algorithms, experiments and interactive protocols. If $\mathcal{D}$ denotes a probability distribution, $x \leftarrow \mathcal{D}$ is the operation of picking an element according to $\mathcal{D}$. If α is neither an algorithm nor a set then $x \leftarrow \alpha$ is a simple assignment statement. If A is a probabilistic algorithm, then $A(x_1, x_2, \cdots; r)$ is the result of running A on inputs $x_1, x_2, \cdots$ and coins r. We let $y \leftarrow A(x_1, x_2, \cdots)$ denote the experiment of picking r at random and letting y be $A(x_1, x_2, \cdots; r)$. By $\Pr[R_1; \cdots; R_n : E]$ we denote the probability of event E, after the ordered execution of random processes $R_1, \cdots, R_n$.

A function $f(\lambda)$ is *negligible*, if for every $c > 0$ there exists an λ_c such that $f(\lambda) < 1/\lambda^c$ for all $\lambda > \lambda_c$. In in this work, for presentation simplicity, when dealing with concrete parameters we also informally say that a quantity lower than 2^{-60} is negligible.

2.1 Authenticated Encryption with Associated Data

The presentation in this section is verbatim from [Z16]. Briefly speaking, an *authenticated encryption with associated data* (AEAD) scheme transforms a message M and a public header information H (e.g., a packet header, an IP address) into a ciphertext C in such a way that C provides both privacy (of M) and authenticity (of C and H) [R02]. In practice, when AEAD is used within cryptographic systems, the associated data is usually implicitly determined from the context (e.g., the ciphertext of the CPA-secure KEM, the hash of the transcript of protocol run or some pre-determined states). For simplicity, we usually do not explicitly specify the associated data in this work. For all the protocols developed in this work that use AEAD, the associated data can be set to be empty without sacrificing provable security.

Let $\mathsf{SE} = (\mathsf{K}_{se}, \mathsf{Enc}, \mathsf{Dec})$ be a symmetric encryption scheme. The probabilistic polynomial-time algorithm K_{se} takes a security parameter κ as input and samples a key K from a finite and non-empty set $\mathcal{K} \bigcap \{0,1\}^\kappa$. For presentation simplicity, we assume $K \leftarrow \mathcal{K} = \{0,1\}^\kappa$. The polynomial-time encryption algorithm $\mathsf{Enc} : \mathcal{K} \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \cup \{\perp\}$ and the (deterministic) polynomial-time decryption algorithm $\mathsf{Dec} : \mathcal{K} \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \cup \{\perp\}$ satisfy: for any $K \leftarrow \mathcal{K}$, any associate data $\mathsf{H} \in \{0,1\}^*$ and any message $M \in \{0,1\}^*$, if $\mathsf{Enc}_K(H, M)$ outputs $C \neq \perp$, then $\mathsf{Dnc}_K(C)$ always outputs M. Here, we assume the ciphertext C bears the associate data H in plain.

Let $\mathcal{A}$ be an adversary. Table 4 describes a security game for AEAD. We define the advantage of $\mathcal{A}$ to be $\mathbf{Adv}^{\mathrm{aead}}_{\mathsf{SE}}(\mathcal{A}) = \left|2 \cdot \Pr[\mathrm{AEAD}^{\mathcal{A}}_{\mathsf{SE}} \ \textit{returns}\ \mathsf{true}] - 1\right|$. We say that the SE scheme is AEAD-secure, if for any sufficiently large κ the advantage of any probabilistic polynomial-time adversary is negligible.

```
main $\mathrm{AEAD}^{\mathcal{A}}_{\mathsf{SE}}$:
$K \leftarrow \mathcal{K}_{\mathrm{se}}$
$\sigma \leftarrow \{0,1\}$
$\sigma' = \mathcal{A}^{\mathbf{Enc},\mathbf{Dec}}$
Ret $(\sigma' = \sigma)$

procedure Enc$(\mathsf{H}, M_0, M_1)$:
If $|M_0| \neq |M_1|$, Ret $\perp$
$C_0 \leftarrow \mathsf{Enc}_K(\mathsf{H}, M_0)$
$C_1 \leftarrow \mathsf{Enc}_K(\mathsf{H}, M_1)$
If $C_0 = \perp$ or $C_1 = \perp$, Ret $\perp$
$\mathcal{C} \overset{\cup}{\leftarrow} C_\sigma$; Ret $C_\sigma$

procedure Dec$(C')$:
If $\sigma = 1 \wedge C' \notin \mathcal{C}$ then
    Ret $\mathsf{Dec}_K(C')$
Ret $\perp$
```

Table 4: AEAD security game

The above AEAD security is quite strong. In particular, it means that, after adaptively seeing a polynomial number of ciphertexts, an efficient adversary is infeasible to generate a new valid ciphertext in the sense its decryption is not "$\perp$". Also, for two independent keys $K, K' \leftarrow \mathcal{K}_{se}$ and any message M and any header information H, $\Pr[\mathsf{Dec}_{K'}(\mathsf{Enc}_K(\mathsf{H}, M)) \neq \perp]$ is negligible.

The AEAD security definition is based on that in [RS06, PRS11], with the following modifications: the length-hiding requirement is removed while header information integrity property is added. In this work, we assume users' identities and public-key information to be of equal length; otherwise, we need length-hiding AEAD as defined in [PRS11, KPW13]. Currently, the most popular AEAD scheme in use may be GCM-AES.

2.2 Key Encapsulation Mechanism (KEM)

We review the definition of KEM given in [D02, HHK17]. A key encapsulation mechanism $\mathsf{KEM} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ consists of three algorithms. On a security parameter κ, the key generation algorithm KeyGen outputs a key pair (pk, sk), where pk also defines a finite key space $\mathcal{K}$. The encapsulation algorithm Encaps, on input pk, outputs a tuple (K, c) where c is said to be an encapsulation of the key K which is contained in key space $\mathcal{K}$. The deterministic decapsulation algorithm Decaps, on input sk and an encapsulation c, outputs either a key $K := \mathsf{Decaps}(sk, c) \in \mathcal{K}$ or a special symbol $\perp \notin \mathcal{K}$ to indicate that c is not a valid encapsulation. We call KEM *δ-correct* if

$$\Pr[\mathsf{Decaps}(sk, c) \neq K | (pk, sk) \leftarrow \mathsf{KeyGen}(1^\kappa); (K, c) \leftarrow \mathsf{Encaps}(pk)] \leq \delta.$$

The security notion, indistinguishability under chosen ciphertext attacks (CCA), is defined w.r.t. Figure 1. For any PPT adversary $\mathcal{A}$, define its CCA-advantage as $Adv^{CCA}_{KEM}(\mathcal{A}) := |Pr[\mathbf{GAME}\ \mathsf{CCA}\ \text{outputs}\ 1]] - 1/2|$. We say the KEM scheme is CCA-secure, if for any sufficiently larger security parameter and any PPT adversary $\mathcal{A}$, $Adv^{CCA}_{KEM}(\mathcal{A})$ is negligible.

2.3 Public-Key Encryption (PKE)

We review the definition of PKE given in [FO13, HHK17]. A public-key encryption scheme is given by a triple of algorithms, $\mathsf{PKE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, where for every sufficiently large $\kappa \in \mathbb{N}$.

- KeyGen, the key-generation algorithm, is a probabilistic polynomial-time (in κ) algorithm which on input 1^κ outputs a pair of strings, (pk, sk), called the public and secret keys, respectively. This experiment is written as $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\kappa)$.

```
GAME IND-CCA
    $(pk, sk) \leftarrow \mathsf{Gen}$
    $b \xleftarrow{\$} \{0,1\}$
    $(K_0^*, c^*) \leftarrow \mathsf{Encaps}(pk)$
    $K_1^* \xleftarrow{\$} \mathcal{K}$
    $b' \leftarrow \mathsf{A}^{\textsc{Decaps}}(c^*, K_b^*)$
    return $[b' = b]$

DECAPS($c \neq c^*$)
    $K := \mathsf{Decaps}(sk, c)$
    return $K$
```

Figure 1: CCA game for KEM.

- $\mathcal{E}$, the encryption algorithm, is a probabilistic polynomial-time (in κ) algorithm that takes public key pk and message M from the message space $\mathtt{MSP}$, draws coins r uniformly from coin space $\mathtt{COIN}$, and produces ciphertext $C := \mathcal{E}_{pk}(M; r)$. This experiment is written as $C \leftarrow \mathcal{E}_{pk}(x)$.
- $\mathcal{D}$, the decryption algorithm, is a deterministic polynomial-time (in κ) algorithm that takes secret key sk and ciphertext $C \in \{0,1\}^*$, and returns message $M := \mathcal{D}_{sk}(C)$ or a special symbol $\perp$ indicating decryption failure.

We require that an asymmetric encryption scheme should satisfy the following *correctness* condition:

We say a PKE scheme is δ-correct, if for every sufficiently large $\kappa \in \mathbb{N}$, every (pk, sk) generated by $\mathsf{KeyGen}(1^\kappa)$ and every $M \in \mathtt{MSP}$, we always have $\mathbf{E}[\max_{M \in \mathtt{MSP}} \Pr[\mathcal{D}_{sk}(\mathcal{E}_{pk}(M)) \neq M]] \leq \delta$.

Definition 2.1 (CCA-security)**.** *Let $\mathsf{PKE} = (\mathsf{KeyGen}, \mathcal{E}, \mathcal{D})$ be an asymmetric encryption scheme, and $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary for PKE. For $\kappa \in \mathbb{N}$, define the following CCA-advantage:*

$$\begin{aligned}\mathbf{Adv}_{\mathcal{A}}^{\mathsf{CCA}}(\kappa) = 2 \cdot \Pr[(pk, sk) \leftarrow \mathsf{KeyGen}(1^\kappa); (M_0, M_1, st) \leftarrow \mathcal{A}_1^{\mathcal{D}_{sk}}(pk);\\ b \leftarrow \{0,1\}; C^* \leftarrow \mathcal{E}_{pk}(M_b) : \mathcal{A}_2^{\mathcal{D}_{sk}}(C^*, st) = b] - 1.\end{aligned}$$

We say that the PKE scheme is CCA-secure, if for every sufficiently large security parameter κ, and PPT adversary $\mathcal{A}$, its CCA-advantage $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{CCA}}$ is negligible in κ.

2.4 The LWE, LWR, and RLWE problems

Given positive *continuous* $\sigma > 0$, define the real Gaussian function $\rho_\sigma(x) \triangleq \exp(-x^2/2\sigma^2)/\sqrt{2\pi\sigma^2}$ for $x \in \mathbb{R}$. Let $D_{\mathbb{Z},\sigma}$ denote the one-dimensional *discrete* Gaussian distribution over $\mathbb{Z}$, which is determined by its probability density function $D_{\mathbb{Z},\sigma}(x) \triangleq \rho_\sigma(x)/\rho_\sigma(\mathbb{Z}), x \in \mathbb{Z}$. Finally, let $D_{\mathbb{Z}^n,\sigma}$ denote the n-dimensional *spherical* discrete Gaussian distribution over $\mathbb{Z}^n$, where each coordinate is drawn *independently* from $D_{\mathbb{Z},\sigma}$.

Given positive integers n and q that are both polynomials in the security parameter λ, an integer vector $\mathbf{s} \in \mathbb{Z}_q^n$, and a probability distribution χ on $\mathbb{Z}_q$, let $A_{q,\mathbf{s},\chi}$ be the distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, and an error term $e \leftarrow \chi$, and outputting the pair $(\mathbf{a}, b = \mathbf{a}^T\mathbf{s} + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. The error distribution χ is typically taken to be the discrete Gaussian probability distribution $D_{\mathbb{Z},\sigma}$ defined previously; However, as suggested in [BCD+16] and as we shall see in Section 6.1, other alternative distributions of χ can be taken. Briefly speaking, the (decisional) *learning with errors* (LWE) assumption [Reg09] says that, for sufficiently large security parameter λ, no probabilistic polynomial-time (PT) algorithm can distinguish, with non-negligible probability, $A_{q,\mathbf{s},\chi}$ from the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$. This holds even if $\mathcal{A}$ sees polynomially many samples, and even if the secret vector $\mathbf{s}$ is drawn randomly from χ^n [ACPS09].

The LWR problem [BPR12] is a "derandomized" variant of the LWE problem. Let $\mathcal{D}$ be some distribution over $\mathbb{Z}_q^n$, and $\mathbf{s} \leftarrow \mathcal{D}$. For integers $q \geq p \geq 2$ and any $x \in \mathbb{Z}_q$, denote

$$\lfloor x \rceil_p = \lfloor \frac{p}{q} x \rceil. \tag{1}$$

Then, for positive integers n and $q \geq p \geq 2$, the LWR distribution $A_{n,q,p}(\mathbf{s})$ over $\mathbb{Z}_q^n \times \mathbb{Z}_p$ is obtained by sampling $\mathbf{a}$ from $\mathbb{Z}_q^n$ uniformly at random, and outputting $\left(\mathbf{a}, \left\lfloor \mathbf{a}^T\mathbf{s} \right\rceil_p\right) \in \mathbb{Z}_q^n \times \mathbb{Z}_p$. Briefly speaking, the (decisional) LWR assumption says that, for sufficiently large security parameter, no PPT algorithm $\mathcal{A}$ can distinguish, with non-negligible probability, the distribution $A_{n,q,p}(\mathbf{s})$ from the distribution $(\mathbf{a} \leftarrow \mathbb{Z}_q^n, \lfloor u \rceil_p)$ where $u \leftarrow \mathbb{Z}_q$. This holds even if $\mathcal{A}$ sees polynomially many samples. An efficient reduction from the LWE problem to the LWR problem, for super-polynomial large q, is provided in [BPR12]. Let B denote the bound for any component in the secret $\mathbf{s}$. It is recently shown that, when $q \geq 2mBp$ (equivalently, $m \leq q/2Bp$), the LWE problem can be reduced to the (decisional) LWR assumption with m independently random samples [BGM+16]. Moreover, the reduction from LWE to LWR is actually independent of the distribution of the secret $\mathbf{s}$.

For the positive integer m that is polynomial in the security parameter λ, let $n \triangleq \varphi(m)$ denote the toties of m, and $\mathcal{K} \triangleq \mathbb{Q}(\zeta_m)$ be the number field obtained by adjoining an abstract element ζ_m satisfying $\Phi_m(\zeta_m) = 0$, where $\Phi_m(x) \in \mathbb{Z}[x]$ is the m-th cyclotomies polynomial of degree n. Moreover, let $\mathcal{R} \triangleq \mathcal{O}_{\mathcal{K}}$ be the ring of integers in $\mathcal{K}$. Finally, given a positive prime $q = \text{poly}(\lambda)$ such that $q \equiv 1 \pmod{m}$, define the quotient ring $\mathcal{R}_q \triangleq \mathcal{R}/q\mathcal{R}$.

We briefly review the RLWE problem, and its hardness result [LPR10, LPR13b, DD12]. It suffices in this work to consider a *special* case of the RLWE problem defined in [LPR10]. Let $n \geq 16$ be a power-of-two and $q = \text{poly}(\lambda)$ be a positive prime such that $q \equiv 1 \pmod{2n}$. Given $\mathbf{s} \leftarrow \mathcal{R}_q$, a sample drawn from the RLWE distribution $A_{n,q,\sigma,\mathbf{s}}$ over $\mathcal{R}_q \times \mathcal{R}_q$ is generated by first choosing $\mathbf{a} \leftarrow \mathcal{R}_q, \mathbf{e} \leftarrow D_{\mathbb{Z}^n,\sigma}$, and then outputting $(\mathbf{a}, \mathbf{a} \cdot \mathbf{s} + \mathbf{e}) \in \mathcal{R}_q \times \mathcal{R}_q$. Roughly speaking, the (decisional) RLWE assumption says that, for sufficiently large security parameter λ, no PPT algorithm $\mathcal{A}$ can distinguish, with non-negligible probability, $A_{n,q,\sigma,\mathbf{s}}$ from the uniform distribution over $\mathcal{R}_q \times \mathcal{R}_q$. This holds even if $\mathcal{A}$ sees polynomially many samples, and even if the secret $\mathbf{s}$ is drawn randomly from the same distribution of the error polynomial $\mathbf{e}$ [DD12, ACPS09]. Moreover, as suggested in [ADPS16], alternative distributions for the error polynomials can be taken for the sake of efficiency while without essentially reducing security.

3 Key Consensus with Noise

$$\begin{array}{ccc}
\underline{\text{Alice}} & & \underline{\text{Bob}} \\
\sigma_1 \in \mathbb{Z}_q & \approx & \sigma_2 \in \mathbb{Z}_q \\
(k_1, v) \leftarrow \mathsf{Con}(\sigma_1, \mathsf{params}) & & \\
 & \xrightarrow{\quad v \quad} & \\
 & & k_2 \leftarrow \mathsf{Rec}(\sigma_2, v, \mathsf{params})
\end{array}$$

Figure 2: Brief depiction of KC, where $k_1, k_2 \in \mathbb{Z}_m$, $v \in \mathbb{Z}_g$ and $|\sigma_1 - \sigma_2|_q \le d$.

Before presenting the definition of key consensus (KC) scheme, we first introduce a new function $|\cdot|_t$ relative to arbitrary positive integer $t \ge 1$: $|x|_t = \min\{x \bmod t, t - x \bmod t\}, \quad \forall x \in \mathbb{Z}$, where the result of modular operation is represented in $\{0, ..., (t-1)\}$. For instance, $|-1|_t = \min\{-1 \bmod t, (t+1) \mod t\} = \min\{t-1, 1\} = 1$. In the following description, we use $|\sigma_1 - \sigma_2|_q$ to measure the distance between two elements $\sigma_1, \sigma_2 \in \mathbb{Z}_q$.

Definition 3.1. *A KC scheme KC = (params, Con, Rec), briefly depicted in Figure 2, is specified as follows.*

- $\mathsf{params} = (q, m, g, d, aux)$ *denotes the system parameters, where q, m, g, d are positive integers satisfying $2 \le m, g \le q, 0 \le d \le \lfloor \frac{q}{2} \rfloor$, and aux denotes some auxiliary values that are usually determined by (q, m, g, d) and could be set to be a special symbol $\emptyset$ indicating "empty".*
- $(k_1, v) \leftarrow \mathsf{Con}(\sigma_1, \mathsf{params})$*: On input of $(\sigma_1 \in \mathbb{Z}_q, \mathsf{params})$, the* probabilistic *polynomial-time conciliation algorithm Con outputs (k_1, v), where $k_1 \in \mathbb{Z}_m$ is the shared-key, and $v \in \mathbb{Z}_g$ is a hint signal that will be publicly delivered to the communicating peer to help the two parties reach consensus.*
- $k_2 \leftarrow \mathsf{Rec}(\sigma_2, v, \mathsf{params})$*: On input of $(\sigma_2 \in \mathbb{Z}_q, v, \mathsf{params})$, the* deterministic *polynomial-time reconciliation algorithm Rec outputs $k_2 \in \mathbb{Z}_m$.*

Correctness: *A KC scheme is* correct*, if it holds $k_1 = k_2$ for any $\sigma_1, \sigma_2 \in \mathbb{Z}_q$ such that $|\sigma_1 - \sigma_2|_q \le d$.*

Security: *A KC scheme is* secure*, if k_1 and v are independent, and k_1 is uniformly distributed over $\mathbb{Z}_m$, whenever $\sigma_1 \leftarrow \mathbb{Z}_q$. The probability is taken over the sampling of σ_1 and the random coins used by Con.*

3.1 Efficiency Upper Bound of KC

The following theorem reveals an upper bound on the parameters q (dominating security and efficiency), m (parameterizing range of consensus key), g (parameterizing bandwidth), and d (parameterizing error rate), which allows us to take balance on these parameters according to different priorities.

Theorem 3.1. *If KC = ($\mathsf{params}, \mathsf{Con}, \mathsf{Rec}$) is a* correct *and* secure *key consensus scheme, and* $\mathsf{params} = (q, m, g, d, aux)$*, then $2md \le q\left(1 - \frac{1}{g}\right)$.*

Before proceeding to prove Theorem 3.1, we first prove the following propositions.

Proposition 3.1. *Given* $\mathsf{params} = (q, m, g, d, aux)$ *for a* correct *and* secure *KC scheme. For any arbitrary fixed* $\sigma_1 \in \mathbb{Z}_q$*, if* $\mathsf{Con}(\sigma_1, \mathsf{params})$ *outputs* (k_1, v) *with positive probability, then the value* k_1 *is fixed w.r.t. the* (v, σ_1)*. That is, for any random coins* (r, r')*, if* $\mathsf{Con}(\sigma_1, \mathsf{params}, r) = (k_1, v)$ *and* $\mathsf{Con}(\sigma_1, \mathsf{params}, r') = (k_1', v)$*, then* $k_1 = k_1'$*.*

Proof. Let $\sigma_2 = \sigma_1$, then $|\sigma_1 - \sigma_2|_q = 0 \leq d$. Then, according to the *correctness* of KC, we have that $k_1 = k_2 = \mathsf{Rec}(\sigma_2, v) = \mathsf{Rec}(\sigma_1, v)$. However, as Rec is a deterministic algorithm, k_2 is fixed w.r.t. (σ_1, v). As a consequence, k_1 is also fixed w.r.t. (σ_1, v), no matter what randomness is used by Con. □ □

Proposition 3.2. *Given* $\mathsf{params} = (q, m, g, d, aux)$ *for a KC scheme, for any* $v \in \mathbb{Z}_g$*, let* S_v *be the set containing all* σ_1 *such that* $\mathsf{Con}(\sigma_1, \mathsf{params})$ *outputs* v *with positive probability. Specifically,*

$$S_v = \left\{\sigma_1 \in \mathbb{Z}_q \mid \Pr\left[(k_1, v') \leftarrow \mathsf{Con}(\sigma_1, \mathsf{params}) : v' = v\right] > 0\right\}.$$

Then, there exists $v_0 \in \mathbb{Z}_g$ *such that* $|S_{v_0}| \geq q/g$*.*

Proof. For each $\sigma_1 \in \mathbb{Z}_q$, we run $\mathsf{Con}(\sigma_1, \mathsf{params})$ and get a pair $(k_1, v) \in \mathbb{Z}_m \times \mathbb{Z}_g$ satisfying $\sigma_1 \in S_v$. Then, the proposition is clear by the pigeonhole principle. □ □

of Theorem 3.1. From Proposition 3.2, there exists a $v_0 \in \mathbb{Z}_g$ such that $|S_{v_0}| \geq q/g$. Note that, for any $\sigma_1 \in S_{v_0}$, $\mathsf{Con}(\sigma_1, \mathsf{params})$ outputs v_0 with positive probability.

For each $i \in \mathbb{Z}_m$, let K_i denote the set containing all σ_1 such that $\mathsf{Con}(\sigma_1, \mathsf{params})$ outputs $(k_1 = i, v = v_0)$ with positive probability. From Proposition 3.1, K_i's form a disjoint partition of S_{v_0}. From the independence between k_1 and v, and the uniform distribution of k_1, (as we assume the underlying KC is *secure*), we know $\Pr[k_1 = i \mid v = v_0] = \Pr[k_1 = i] > 0$, and so K_i is non-empty for each $i \in \mathbb{Z}_m$. Now, for each $i \in \mathbb{Z}_m$, denote by K_i' the set containing all $\sigma_2 \in \mathbb{Z}_q$ such that $\mathsf{Rec}(\sigma_2, v_0, \mathsf{params}) = i$. As Rec is deterministic, K_i''s are well-defined and are disjoint.

From the *correctness* of KC, for every $\sigma_1 \in K_i, |\sigma_2 - \sigma_1|_q \leq d$, we have $\sigma_2 \in K_i'$. That is, $K_i + [-d, d] \subseteq K_i'$.

We shall prove that $K_i + [-d, d]$ contains at least $|K_i| + 2d$ elements. If $K_i + [-d, d] = \mathbb{Z}_m$, then $m = 1$, which is a contradiction (we exclude the case of $m = 1$ in the definition of KC as it is a trivial case). If there exists an $x \in \mathbb{Z}_m$ such that $x \notin K_i + [-d, d]$, we can see $\mathbb{Z}_m$ as a segment starting from the point x by arranging its elements as $x, (x+1) \bmod m, (x+2) \bmod m, \ldots, (x+m-1) \bmod m$. Let l be the left most element in $K_i + [-d, d]$ on the segment, and r be the right most such element. Then $K_i + [-d, d]$ contains at least $|K_i|$ elements between l and r inclusively on the segment. Since $l + [-d, 0]$ and $r + [0, d]$ are subset of $K_i + [-d, d]$, and are not overlap (because $x \notin K_i + [-d, d]$), the set $K_i + [-d, d]$ contains at least $|K_i| + 2d$ elements.

Now we have $|K_i| + 2d \leq |K_i'|$. When we add up on both sides for all $i \in \mathbb{Z}_m$, then we derive $|S_{v_0}| + 2md \leq q$. By noticing that $|S_{v_0}| \geq q/g$, the theorem is established. □ □

3.2 Construction and Analysis of OKCN

The key consensus scheme, named OKCN, is presented in Algorithm 1.[5] An illustration diagram is given in Figure 3. Some explanations for implementation details are given below.

[5]Note that, for the general case of OKCN, the Con is probabilistic. When OKCN is used for key exchange or public-key encryption, the randomness can be derived from transcripts.

Algorithm 1 OKCN: Symmetric KC with Noise

```
1:  params $= (q, m, g, d, aux)$, $aux = \{q' = \mathsf{lcm}(q, m), \alpha = q'/q, \beta = q'/m\}$
2:  procedure Con$((\sigma_1, \mathsf{params}))$                          ▷ $\sigma_1 \in [0, q-1]$
3:      $e \leftarrow [-\lfloor(\alpha - 1)/2\rfloor, \lfloor \alpha/2 \rfloor]$
4:      $\sigma_A = (\alpha\sigma_1 + e) \bmod q'$
5:      $k_1 = \lfloor \sigma_A/\beta \rfloor \in \mathbb{Z}_m$
6:      $v' = \sigma_A \mod \beta$
7:      $v = \lfloor v'g/\beta \rfloor$                                      ▷ $v \in \mathbb{Z}_g$
8:      return $(k_1, v)$
9:  end procedure
10: procedure Rec$(\sigma_2, v, \mathsf{params})$                          ▷ $\sigma_2 \in [0, q-1]$
11:     $k_2 = \lfloor \alpha\sigma_2/\beta - (v + 1/2)/g \rceil \bmod m$
12:     return $k_2$
13: end procedure
```

Define $\sigma'_A = \alpha\sigma_1 + e$. Note that it always holds $\sigma'_A < q'$. However, in some rare cases, σ'_A could be a negative value; for example, for the case that $\sigma_1 = 0$ and $e \in [-\lfloor(\alpha-1)/2\rfloor, -1]$. Setting $\sigma_A = \sigma'_A \mod q'$, in line 4, is to ensure that σ_A is always a non-negative value in $\mathbb{Z}_{q'}$, which can be simply implemented as follows: if $\sigma'_A < 0$ then set $\sigma_A = \sigma'_A + q'$, otherwise set $\sigma_A = \sigma'_A$. Considering potential timing attacks, conditional statement judging whether σ'_A is negative or not can be avoided by a bitwise operation extracting the sign bit of σ'_A. In specific, suppose σ'_A is a 16-bit signed or unsigned integer, then one can code $\sigma_A = \sigma'_A + ((\sigma'_A >> 15)\&1) * q'$ in C language. The same techniques can also be applied to the calculation in line 11.

In lines 5 and 6, (k_1, v') can actually be calculated simultaneously by a single command *div* in assembly language. In line 11, the floating point arithmetic can be replaced by integer arithmetic. If m is small enough, such as 2 or 3, the slow complex integer division operation can be replaced by relative faster conditional statements.

The value $v + 1/2$, in line 11, estimates the exact value of $v'g/\beta$. Such an estimation can be more accurate, if one chooses to use the average value of all $v'g/\beta$'s such that $\lfloor v'g/\beta \rfloor = v$. Though such accuracy can improve the bound on correctness slightly, the formula calculating k_2 becomes more complicated.

The following fact is direct from the definition of $|\cdot|_t$.

Fact 3.1. *For any $x, y, t, l \in \mathbb{Z}$ where $t \geq 1$ and $l \geq 0$, if $|x - y|_q \leq l$, then there exists $\theta \in \mathbb{Z}$ and $\delta \in [-l, l]$ such that $x = y + \theta t + \delta$.*

Theorem 3.2. *Suppose that the system parameters satisfy $(2d+1)m < q\left(1 - \frac{1}{g}\right)$ where $m \geq 2$ and $g \geq 2$. Then, the* OKCN *scheme is* correct.

Proof. Suppose $|\sigma_1 - \sigma_2|_q \leq d$. By Fact 3.1, there exist $\theta \in \mathbb{Z}$ and $\delta \in [-d, d]$ such that $\sigma_2 = \sigma_1 + \theta q + \delta$. From line 4 and 6 in Algorithm 1, we know that there is a $\theta' \in \mathbb{Z}$, such that $\alpha\sigma_1 + e + \theta' q' = \sigma_A = k_1\beta + v'$. And from the definition of α, β, we have $\alpha/\beta = m/q$. Taking these into the formula of k_2 in Rec (line 11 in Algorithm 1), we have

$$k_2 = \lfloor \alpha\sigma_2/\beta - (v + 1/2)/g \rceil \bmod m \quad (2)$$

$$= \lfloor \alpha(\theta q + \sigma_1 + \delta)/\beta - (v + 1/2)/g \rceil \bmod m \quad (3)$$

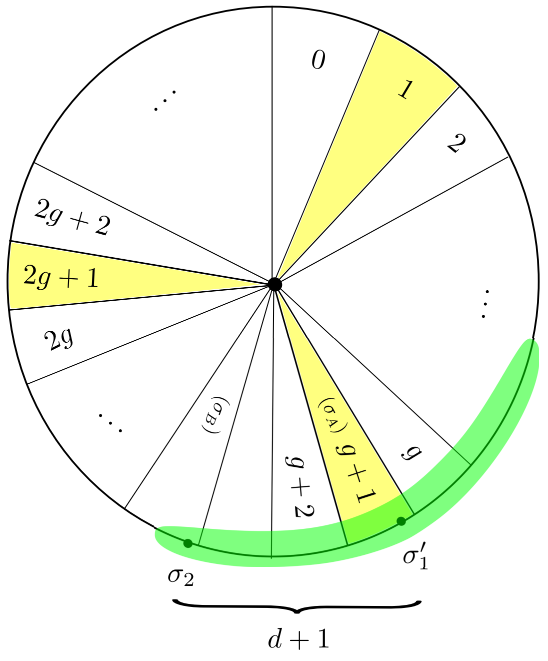

Figure 3: An illustration diagram of OKCN

$$= \left\lfloor m(\theta - \theta') + \frac{1}{\beta}(k_1\beta + v' - e) + \frac{\alpha\delta}{\beta} - \frac{1}{g}(v + 1/2) \right\rceil \bmod m \tag{4}$$

$$= \left\lfloor k_1 + \left(\frac{v'}{\beta} - \frac{v+1/2}{g}\right) - \frac{e}{\beta} + \frac{\alpha\delta}{\beta} \right\rceil \bmod m \tag{5}$$

Notice that $|v'/\beta - (v+1/2)/g| = |v'g - \beta(v+1/2)|/\beta g \leq 1/2g$. So

$$\left|\left(\frac{v'}{\beta} - \frac{v+1/2}{g}\right) - \frac{e}{\beta} + \frac{\alpha\delta}{\beta}\right| \leq \frac{1}{2g} + \frac{\alpha}{\beta}(d + 1/2).$$

From the assumed condition $(2d+1)m < q(1-\frac{1}{g})$, we get that the right-hand side is strictly smaller than $1/2$; Consequently, after the rounding, $k_2 = k_1$. □ □

Theorem 3.3. *OKCN is secure. Specifically, when $\sigma_1 \leftarrow \mathbb{Z}_q$, k_1 and v are independent, and k_1 is uniform over $\mathbb{Z}_m$, where the probability is taken over the sampling of σ_1 and the random coins used by* Con.

Proof. Recall that $q' = \mathsf{lcm}(q, m), \alpha = q'/q, \beta = q'/m$. We first demonstrate that σ_A is subject to uniform distribution over $\mathbb{Z}_{q'}$. Consider the map $f : \mathbb{Z}_q \times \mathbb{Z}_\alpha \to \mathbb{Z}_{q'}$; $f(\sigma, e) = (\alpha\sigma + e) \bmod q'$, where the elements in $\mathbb{Z}_q$ and $\mathbb{Z}_\alpha$ are represented in the same way as specified in Algorithm 1. It is easy to check that f is an one-to-one map. Since $\sigma_1 \leftarrow \mathbb{Z}_q$ and $e \leftarrow \mathbb{Z}_\alpha$ are subject to uniform distributions, and they are independent, $\sigma_A = (\alpha\sigma_1 + e) \bmod q' = f(\sigma_1, e)$ is also subject to uniform distribution over $\mathbb{Z}_{q'}$.

In the similar way, defining $f' : \mathbb{Z}_m \times \mathbb{Z}_\beta \to \mathbb{Z}_{q'}$ such that $f'(k_1, v') = \beta k_1 + v'$, then f' is obviously a one-to-one map. From line 6 of Algorithm 1, $f'(k_1, v') = \sigma_A$. As σ_A is distributed uniformly over $\mathbb{Z}_{q'}$, (k_1, v') is uniformly distributed over $\mathbb{Z}_m \times \mathbb{Z}_\beta$, and so k_1 and v' are independent. As v only depends on v', k_1 and v are independent. □ □

Algorithm 2 OKCN power 2

```
1: params : $q = 2^{\bar{q}}, g = 2^{\bar{g}}, m = 2^{\bar{m}}, d, aux = \{(\beta = q/m = 2^{\bar{q}-\bar{m}}, \gamma = \beta/g = 2^{\bar{q}-\bar{m}-\bar{g}})\}$
2: procedure CON($\sigma_1$, params)
3:     $k_1 = \lfloor \sigma_1/\beta \rfloor$
4:     $v = \lfloor (\sigma_1 \bmod \beta)/\gamma \rfloor$
5:     return $(k_1, v)$
6: end procedure
7: procedure REC($\sigma_2, v$, params)
8:     $k_2 = \lfloor \sigma_2/\beta - (v + 1/2)/g \rceil \bmod m$
9:     return $k_2$
10: end procedure
```

Algorithm 3 OKCN simple

```
1: params : $q = 2^{\bar{q}}, g = 2^{\bar{g}}, m = 2^{\bar{m}}, d$, where $\bar{g} + \bar{m} = \bar{q}$
2: procedure CON($\sigma_1$, params)
3:     $k_1 = \left\lfloor \frac{\sigma_1}{g} \right\rfloor$
4:     $v = \sigma_1 \bmod g$
5:     return $(k_1, v)$
6: end procedure
7: procedure REC($\sigma_2, v$, params)
8:     $k_2 = \left\lfloor \frac{\sigma_2 - v}{g} \right\rceil \bmod m$
9:     return $k_2$
10: end procedure
```

3.2.1 Special Parameters, and Performance Speeding-Up

The first and the second line of Con (line 3 and 4 in Algorithm 1) play the role in transforming a uniform distribution over $\mathbb{Z}_q$ to a uniform distribution over $\mathbb{Z}_{q'}$. If one chooses q, g, m to be power of 2, i.e., $q = 2^{\bar{q}}, g = 2^{\bar{g}}, m = 2^{\bar{m}}$ where $\bar{q}, \bar{g}, \bar{m} \in \mathbb{Z}$, then such transformation is not necessary, and the random noise e used in calculating σ_A in Algorithm 1 is avoided. In this case Con and Rec can be simplified to Algorithm 2. The following corollary is straightforward.

Corollary 3.1. *If q and m are power of 2, and d, g, m satisfy $2md < q\left(1 - \frac{1}{g}\right)$, then the KC scheme described in Algorithm 2 is both* correct *and* secure.

If we take $\bar{g} + \bar{m} = \bar{q}$, Algorithm 2 can be further simplified into the variant depicted in Algorithm 3, with the constraint on parameters is further relaxed.

Corollary 3.2. *If m, g are power of 2, $q = m \cdot g$, and $2md < q$, then the KC scheme described in Algorithm 3 is* correct *and* secure. *Notice that the constraint on parameters is further simplified to $2md < q$ in this case.*

The proof of Corollary 3.2 is given in Appendix C.

4 Asymmetric Key Consensus with Noise

$$\begin{array}{ccc} \underline{\text{Alice}} & & \underline{\text{Bob}} \\ \sigma_1 & \approx & \sigma_2 \\ k_1 \in \mathbb{Z}_m & & \\ v \leftarrow \mathsf{Con}(\sigma_1, k_1, \mathsf{params}) & & \\ & \xrightarrow{\quad v \quad} & \\ & & k_2 \leftarrow \mathsf{Rec}(\sigma_2, v, \mathsf{params}) \end{array}$$

Figure 4: Brief depiction of AKC

As we shall see, for OKCN-based key exchange both the initiator and the responder play a *symmetric* role in outputting the shared-key, in the sense that no one can pre-determine the session-key before the KE protocol run. Though OKCN is well desirable for (authenticated) key exchange, it is, however, not well suitable for *directly* achieving key transport and public-key encryption. This motivates us to introduce *asymmetric key consensus* (AKC), as specified below.

Definition 4.1. *An asymmetric key consensus scheme AKC* $= (\mathsf{params}, \mathsf{Con}, \mathsf{Rec})$ *is specified as follows:*

- $\mathsf{params} = (q, m, g, d, aux)$ *denotes the system parameters, where* q, $2 \leq m, g \leq q, 1 \leq d \leq \lfloor \frac{q}{2} \rfloor$ *are positive integers, and* aux *denotes some auxiliary values that are usually determined by* (q, m, g, d) *and could be set to be empty.*

- $v \leftarrow \mathsf{Con}(\sigma_1, k_1, \mathsf{params})$*: On input of* $(\sigma_1 \in \mathbb{Z}_q, k_1 \in \mathbb{Z}_m, \mathsf{params})$*, the* probabilistic *polynomial-time conciliation algorithm* Con *outputs the public hint signal* $v \in \mathbb{Z}_g$.

- $k_2 \leftarrow \mathsf{Rec}(\sigma_2, v, \mathsf{params})$*: On input of* $(\sigma_2, v, \mathsf{params})$*, the* deterministic *polynomial-time algorithm* Rec *outputs* $k_2 \in \mathbb{Z}_m$.

Correctness: *An AKC scheme is* correct*, if it holds* $k_1 = k_2$ *for any* $\sigma_1, \sigma_2 \in \mathbb{Z}_q$ *such that* $|\sigma_1 - \sigma_2|_q \leq d$.

Security: *An AKC scheme is* secure*, if* v *is independent of* k_1 *whenever* σ_1 *is uniformly distributed over* $\mathbb{Z}_q$*. Specifically, for arbitrary* $\tilde{v} \in \mathbb{Z}_g$ *and arbitrary* $\tilde{k}_1, \tilde{k}_1' \in \mathbb{Z}_m$*, it holds that* $\Pr[v = \tilde{v} | k_1 = \tilde{k}_1] = \Pr[v = \tilde{v} | k_1 = \tilde{k}_1']$*, where the probability is taken over* $\sigma_1 \leftarrow \mathbb{Z}_q$ *and the random coins used by* Con.

When AKC is used as a building tool for key transport, k_1 is taken uniformly at random from $\mathbb{Z}_m$. However, when AKC is used for public-key encryption, k_1 can be arbitrary value from the space of plaintext messages. In any case, k_1 can be generated offline, and can be input to the party Alice.

Theorem 4.1. *Let AKC be an asymmetric key consensus scheme with* $\mathsf{params} = (q, m, d, g, aux)$. *If AKC is* correct *and* secure*, then* $2md \leq q\left(1 - \frac{m}{g}\right)$.

Comparing the formula $2md \leq q(1-m/g)$ in Theorem 4.1 with the formula $2md \leq q(1-1/g)$ in Theorem 3.1, we see that the only difference is a factor m in g. This indicates that, on the same values of (q, m, d), an AKC scheme has to use a bigger bandwidth parameter g compared to KC.

Before proving Theorem 4.1, we first adjust Proposition 3.2 to the AKC setting, as following.

Proposition 4.1. *Given* $\mathsf{params} = (q, m, g, d, aux)$ *for an* correct *and* secure *AKC scheme, then there exists* $v_0 \in \mathbb{Z}_g$ *such that* $|S_{v_0}| \geq mq/g$.

Proof. If k_1 is taken uniformly at random from $\mathbb{Z}_m$, AKC can be considered as a special KC scheme by treating $k_1 \leftarrow \mathbb{Z}_m; v \leftarrow \mathsf{Con}(\sigma_1, k_1, \mathsf{params})$ as $(k_1, v) \leftarrow \mathsf{Con}(\sigma_1, \mathsf{params})$. Consequently, Proposition 3.1 holds for this case.

Denote $S'_v \overset{\triangle}{=} \{(\sigma_1, k_1) \in \mathbb{Z}_q \times \mathbb{Z}_m \mid \Pr[v' \leftarrow \mathsf{Con}(\sigma_1, k_1, \mathsf{params}) : v' = v] > 0\}$. Then, S_v defined in Proposition 3.2 equals to the set containing all the values of σ_1 appeared in $(\sigma_1, \cdot) \in S'_v$. We run $\mathsf{Con}(\sigma_1, k_1, \mathsf{params})$ for each pair of $(\sigma_1, k_1) \in \mathbb{Z}_q \times \mathbb{Z}_m$. By the pigeonhole principle, there must exist a $v_0 \in \mathbb{Z}_g$ such that $|S'_{v_0}| \geq qm/g$. For any two pairs (σ_1, k_1) and (σ'_1, k'_1) in S'_{v_0}, if $\sigma_1 = \sigma'_1$, from Proposition 3.1 we derive that $k_1 = k'_1$, and then $(\sigma_1, k_1) = (\sigma'_1, k'_1)$. Hence, if (σ_1, k_1) and (σ'_1, k'_1) are different, then $\sigma_1 \neq \sigma'_1$, and so $|S_{v_0}| = |S'_{v_0}| \geq mq/g$. □ □

Proof of Theorem 4.1. By viewing AKC, with $k_1 \leftarrow \mathbb{Z}_q$, as a special KC scheme, all the reasoning in the proof of Theorem 3.1 holds true now. At the end of the proof of Theorem 3.1, we derive $|S_{v_0}| + 2md \leq q$. By taking $|S_{v_0}| \geq mq/g$ according to Proposition 4.1, the proof is finished. □ □

4.1 Construction and Analysis of AKCN

Algorithm 4 AKCN: Asymmetric KC with Noise

```
1: params = $(q, m, g, d, aux)$, where $aux = \emptyset$.
2: procedure Con($\sigma_1, k_1$, params)                ▷ $\sigma_1 \in [0, q-1]$
3:     $v = \lfloor g(\sigma_1 + \lfloor k_1 q/m \rceil)/q \rceil \bmod g$
4:     return $v$
5: end procedure
6: procedure Rec($\sigma_2, v$, params)                 ▷ $\sigma_2 \in [0, q-1]$
7:     $k_2 = \lfloor m(v/g - \sigma_2/q) \rceil \bmod m$
8:     return $k_2$
9: end procedure
```

The AKCN scheme, referred to as *asymmetric key consensus with noise*, is depicted in Algorithm 4. We note that, in some sense, AKCN could be viewed as the generalization and optimization of the consensus mechanism proposed in [LPR10] for CPA-secure public-key encryption. For AKCN, we can opaline compute and store k_1 and $g\lfloor k_1 q/m \rceil$ in order to accelerate online performance.

Theorem 4.2. *Suppose the parameters of AKCN satisfy* $(2d+1)m < q\left(1 - \frac{m}{g}\right)$. *Then, the AKCN scheme described in Algorithm 4 is* correct.

Proof. From the formula generating v, we know that there exist $\varepsilon_1, \varepsilon_2 \in \mathbb{R}$ and $\theta \in \mathbb{Z}$, where $|\varepsilon_1| \leq 1/2$ and $|\varepsilon_2| \leq 1/2$, such that

$$v = \frac{g}{q}\left(\sigma_1 + \left(\frac{k_1 q}{m} + \varepsilon_1\right)\right) + \varepsilon_2 + \theta g$$

Taking this into the formula computing k_2 in Rec, we have

$$\begin{aligned} k_2 &= \lfloor m(v/g - \sigma_2/q) \rceil \bmod m \\ &= \left\lfloor m\left(\frac{1}{q}(\sigma_1 + k_1 q/m + \varepsilon_1) + \frac{\varepsilon_2}{g} + \theta - \frac{\sigma_2}{q}\right)\right\rceil \bmod m \\ &= \left\lfloor k_1 + \frac{m}{q}(\sigma_1 - \sigma_2) + \frac{m}{q}\varepsilon_1 + \frac{m}{g}\varepsilon_2 \right\rceil \bmod m \end{aligned}$$

By Fact 3.1 (page 18), there exist $\theta' \in \mathbb{Z}$ and $\delta \in [-d, d]$ such that $\sigma_1 = \sigma_2 + \theta' q + \delta$. Hence,

$$k_2 = \left\lfloor k_1 + \frac{m}{q}\delta + \frac{m}{q}\varepsilon_1 + \frac{m}{g}\varepsilon_2 \right\rceil \bmod m$$

Since $|m\delta/q + m\varepsilon_1/q + m\varepsilon_2/g| \leq md/q + m/2q + m/2g < 1/2$, $k_1 = k_2$. □ □

Theorem 4.3. *The AKCN scheme is* secure. *Specifically, v is independent of k_1 when $\sigma_1 \leftarrow \mathbb{Z}_q$.*

Proof. For arbitrary $\tilde{v} \in \mathbb{Z}_g$ and arbitrary $\tilde{k}_1, \tilde{k}_1' \in \mathbb{Z}_m$, we prove that $\Pr[v = \tilde{v} | k_1 = \tilde{k}_1] = \Pr[v = \tilde{v} | k_1 = \tilde{k}_1']$ when $\sigma_1 \leftarrow \mathbb{Z}_q$.

For any $(\tilde{k}, \tilde{v})$ in $\mathbb{Z}_m \times \mathbb{Z}_g$, the event $(v = \tilde{v} \mid k_1 = \tilde{k})$ is equivalent to the event that there exists $\sigma_1 \in \mathbb{Z}_q$ such that $\tilde{v} = \lfloor g(\sigma_1 + \lfloor \tilde{k}q/m \rceil)/q \rceil \bmod g$. Note that $\sigma_1 \in \mathbb{Z}_q$ satisfies $\tilde{v} = \lfloor g(\sigma_1 + \lfloor \tilde{k}q/m \rceil)/q \rceil \bmod g$, if and only if there exist $\varepsilon \in (-1/2, 1/2]$ and $\theta \in \mathbb{Z}$ such that $\tilde{v} = g(\sigma_1 + \lfloor \tilde{k}q/m \rceil)/q + \varepsilon - \theta g$. That is, $\sigma_1 = (q(\tilde{v} - \varepsilon)/g - \lfloor \tilde{k}q/m \rceil) \bmod q$, for some $\varepsilon \in (-1/2, 1/2]$. Let $\Sigma(\tilde{v}, \tilde{k}) = \{\sigma_1 \in \mathbb{Z}_q \mid \exists \varepsilon \in (-1/2, 1/2] \ s.t.\ \sigma_1 = (q(\tilde{v} - \varepsilon)/g - \lfloor \tilde{k}q/m \rceil) \bmod q\}$. Defining the map $\phi : \Sigma(\tilde{v}, 0) \to \Sigma(\tilde{v}, \tilde{k})$, by setting $\phi(x) = \left(x - \lfloor \tilde{k}q/m \rceil\right) \bmod q$. Then ϕ is obviously a one-to-one map. Hence, the cardinality of $\Sigma(\tilde{v}, \tilde{k})$ is irrelevant to $\tilde{k}$. Specifically, for arbitrary $\tilde{v} \in \mathbb{Z}_g$ and arbitrary $\tilde{k}_1, \tilde{k}_1' \in \mathbb{Z}_m$, it holds that $\left|\Sigma(\tilde{v}, \tilde{k}_1)\right| = \left|\Sigma(\tilde{v}, \tilde{k}_1')\right| = |\Sigma(\tilde{v}, 0)|$

Now, for arbitrary $\tilde{v} \in \mathbb{Z}_g$ and arbitrary $\tilde{k} \in \mathbb{Z}_m$, when $\sigma_1 \leftarrow \mathbb{Z}_q$ we have that $\Pr[v = \tilde{v} \mid k_1 = \tilde{k}] = \Pr\left[\sigma_1 \in \Sigma(\tilde{v}, \tilde{k}) \mid k_1 = \tilde{k}\right] = |\Sigma(\tilde{v}, \tilde{k})|/q = |\Sigma(\tilde{v}, 0)|/q$. The right-hand side only depends on $\tilde{v}$, and so v is independent of k_1. □ □

4.1.1 Simplified Variants of AKCN for Special Parameters

We consider the parameters $q = g = 2^{\bar{q}}, m = 2^{\bar{m}}$ for positive integers $\bar{q}, \bar{m}$. Then the two rounding operations in line 3 of Con (in Algorithm 4) can be directly eliminated, since only integers are involved in the computation. We have the following variant described in Algorithm 5. Note that, in Algorithm 5, the modular and multiplication/division operations can be implemented by simple bitwise operations.

For the protocol variant presented in Algorithm 5, its correctness and security can be proved with a relaxed constraint on the parameters of (q, d, m), as shown in the following corollary.

Corollary 4.1. *If q and m are power of 2, and d, m and q satisfy $2md < q$, then the AKCN scheme described in Algorithm 5 is* correct *and* secure.

Algorithm 5 AKCN power 2

```
1: params : $q = g = 2^{\bar{q}}, m = 2^{\bar{m}}, aux = \{G = q/m\}$
2: procedure Con($\sigma_1, k_1$, params)
3:     $v = (\sigma_1 + k_1 \cdot G) \bmod q$, where $k_1 \cdot G$ can be opaline computed
4:     return $v$
5: end procedure
6: procedure Rec($\sigma_2, v$, params)
7:     $k_2 = \lfloor (v - \sigma_2)/G \rceil \bmod m$
8:     return $k_2$
9: end procedure
```

Algorithm 6 AKCN simple

```
1: params = $(q, m, g, d, aux)$, where $q = 2^{\bar{q}}$, $g = 2^{\bar{g}}$, $m = 2^{\bar{m}}$, and $q = gm$ (i.e., $\bar{g} + \bar{m} = \bar{q}$)
2: procedure Con($\sigma_1, k_1$, params)                          ▷ $\sigma_1 \in [0, q-1]$
3:     $v = \lfloor (k_1 g + \sigma_1)/m \rceil \bmod g$              ▷ $k_1 g/m$ can be offline computed
4:     return $v$
5: end procedure
6: procedure Rec($\sigma_2, v$, params)                            ▷ $\sigma_2 \in [0, q-1]$
7:     $k_2 = \lfloor (mv - \sigma_2)/g \rceil \bmod m$
8:     return $k_2$
9: end procedure
```

Proof. For correctness, suppose $|\sigma_1 - \sigma_2|_q \le d$, then there exit $\delta \in [-d, d]$ and $\theta \in \mathbb{Z}$ such that $\sigma_2 = \sigma_1 + \theta q + \delta$. From the formula calculating v, there exists $\theta' \in \mathbb{Z}$ such that $v = \sigma_1 + k_1 2^{\bar{q}-\bar{m}} + \theta' q$. Taking these into the formula computing k_2, line 7 of Rec in Algorithm 5, we have

$$\begin{aligned} k_2 &= \lfloor (v - \sigma_1 - \delta - \theta q)/2^{\bar{q}-\bar{m}} \rceil \bmod m \\ &= \lfloor (k_1 2^{\bar{q}-\bar{m}} - \delta)/2^{\bar{q}-\bar{m}} \rceil \bmod m \\ &= \left(k_1 - \lfloor \delta/2^{\bar{q}-\bar{m}} \rceil\right) \bmod m \end{aligned}$$

If $2md < q$, then $|\delta/2^{\bar{q}-\bar{m}}| < 1/2$, so that $k_1 = k_2$.

For security, as a special case of the generic AKCN scheme in Algorithm 4, the security of the AKCN scheme in Algorithm 5 directly follows from that of Algorithm 4. □ □

Corollary 4.2. *If q, m and g all are power of 2 satisfying $q = mg$, and d, m and g satisfy $m + 2d < g$, then the AKCN-simple described in Algorithm 6 is* correct *and* secure.

Proof. For correctness, suppose $|\sigma_1 - \sigma_2|_q \le d$, then there exit $\delta \in [-d, d]$ and $\theta \in \mathbb{Z}$ such that $\sigma_2 = \sigma_1 + \theta q + \delta$. From the formula calculating v, there exist $\theta' \in \mathbb{Z}$ and $\varepsilon \in (-1/2, 1/2]$ such that $v = \sigma_1 2^{-\bar{m}} + k_1 2^{\bar{g}-\bar{m}} + \varepsilon + \theta' g$. Taking these into the formula computing k_2, line 7 of Rec in Algorithm 5, we have

$$k_2 = \lfloor k_1 + (m\varepsilon - \delta)/g \rceil \bmod m$$

If $m + 2d < g$, then $|k_1 + (m\varepsilon - \delta)/g| < 1/2$, so that $k_1 = k_2$.

As a special case of the AKCN scheme, the security of the AKCN-simple scheme in Algorithm 6 directly follows from that of Algorithm 4. □ □

Initiator

$\mathsf{seed} \leftarrow \{0,1\}^{\kappa}$

$\mathbf{A} = \mathsf{Gen}(\mathsf{seed}) \in \mathbb{Z}_q^{n \times n}$

$\mathbf{X}_1 \leftarrow \chi^{n \times l_A}$

$\mathbf{Y}_1 = \lfloor \mathbf{A}\mathbf{X}_1 \rceil_p$

$$\xrightarrow{\mathsf{seed}, \mathbf{Y}_1 \in \mathbb{Z}_p^{n \times l_A}}$$

Responder

$\mathbf{A} = \mathsf{Gen}(\mathsf{seed})$

$\mathbf{X}_2 \leftarrow \chi^{n \times l_B}$

$\mathbf{Y}_2 = \lfloor \mathbf{A}^T \mathbf{X}_2 \rceil_p$

$\boldsymbol{\epsilon} \leftarrow [-q/2p, q/2p - 1]^{n \times l_A}$

$\mathbf{\Sigma}_2 = \mathbf{Y}_1^T \mathbf{X}_2 + \lfloor \boldsymbol{\epsilon}^T \mathbf{X}_2 \rceil_p$

$(\mathbf{K}_2, \mathbf{V}) \leftarrow \mathsf{Con}(\mathbf{\Sigma}_2, \mathsf{params})$

$$\xleftarrow{\mathbf{Y}_2 \in \mathbb{Z}_p^{n \times l_B}, \mathbf{V} \in \mathbb{Z}_g^{l_A \times l_B}}$$

$\mathbf{\Sigma}_1 = \mathbf{X}_1^T \mathbf{Y}_2$

$\mathbf{K}_1 \leftarrow \mathsf{Rec}(\mathbf{\Sigma}_1, \mathbf{V}, \mathsf{params})$

Figure 5: LWR-based key exchange from KC, where $\mathbf{K}_1, \mathbf{K}_2 \in \mathbb{Z}_m^{l_A \times l_B}$ and $|\mathbf{K}_1| = |\mathbf{K}_2| = l_A l_B |m|$.

5 LWR-Based Key Exchange from KC and AKC

In this section, we present the applications of OKCN and AKCN to key exchange protocols based on LWR.[6] The LWR-based key exchange (KE) is depicted in Figure 5. Denote by $(n, l_A, l_B, q, p, KC, \chi)$ the system parameters, where $p|q$, and p and q are chosen to be power of 2. Let $KC = (\mathsf{params} = (p, m, g, d, aux), \mathsf{Con}, \mathsf{Rec})$ be a *correct* and *secure* key consensus scheme, χ be a small noise distribution over $\mathbb{Z}_q$, and Gen be a pseudo-random generator (PRG) generating the matrix $\mathbf{A}$ from a small seed. For presentation simplicity, we assume $\mathbf{A} \in \mathbb{Z}_q^{n \times n}$ to be square matrix. The length of the random seed, i.e., κ, is typically set to be 256.

The actual session-key is derived from $\mathbf{K}_1$ and $\mathbf{K}_2$ via some key derivation function KDF. For presentation simplicity, the functions Con and Rec are applied to matrices, meaning that they are applied to each of the coordinates respectively.

For presentation simplicity, we describe the LWR-based key exchange protocol from a KC scheme in Figure 5. But it can be trivially adapted to work on any *correct* and *secure* AKC scheme, which is also described in Figure 6. In this case, the responder user Bob simply chooses $\mathbf{K}_2 \in \mathbb{Z}_m^{l_A \times l_B}$ for PKE where $\mathbf{K}_2$ corresponds to the arbitrary plaintext message (or $\mathbf{K}_2 \leftarrow \mathbb{Z}_m^{l_A \times l_B}$ for KEM), and the output of $\mathsf{Con}(\mathbf{\Sigma}_2, \mathbf{K}_2, \mathsf{params})$ is simply defined to be $\mathbf{V}$. For presentation simplicity, in the following security definition and analysis we also simply assume that the output of the PRG Gen is truly random (which is simply assumed to be a random oracle in [ADPS16]).

[6]Note that AKCN-based KE is actually key transport. But for presentation simplicity, we do not make distinction between them in this work.

Initiator

$\mathsf{seed} \leftarrow \{0,1\}^{\kappa}$
$\mathbf{A} = \mathsf{Gen}(\mathsf{seed}) \in \mathbb{Z}_q^{n\times n}$
$\mathbf{X}_1 \leftarrow \chi^{n\times l_A}$
$\mathbf{Y}_1 = \lfloor \mathbf{A}\mathbf{X}_1 \rceil_p$

$$\xrightarrow{\mathsf{seed}, \mathbf{Y}_1 \in \mathbb{Z}_p^{n\times l_A}}$$

Responder

$\mathbf{K}_2 \in \mathbb{Z}_m^{l_A\times l_B}$
$\mathbf{A} = \mathsf{Gen}(\mathsf{seed})$
$\mathbf{X}_2 \leftarrow \chi^{n\times l_B}$
$\mathbf{Y}_2 = \lfloor \mathbf{A}^T\mathbf{X}_2 \rceil_p$
$\boldsymbol{\epsilon} \leftarrow [-q/2p, q/2p-1]^{n\times l_A}$
$\mathbf{\Sigma}_2 = \mathbf{Y}_1^T\mathbf{X}_2 + \lfloor \boldsymbol{\epsilon}^T\mathbf{X}_2 \rceil_p$
$\mathbf{V} \leftarrow \mathsf{Con}(\mathbf{\Sigma}_2, \mathbf{K}_2, \mathsf{params})$

$$\xleftarrow{\mathbf{Y}_2 \in \mathbb{Z}_p^{n\times l_B}, \mathbf{V} \in \mathbb{Z}_g^{l_A\times l_B}}$$

$\mathbf{\Sigma}_1 = \mathbf{X}_1^T\mathbf{Y}_2$
$\mathbf{K}_1 \leftarrow \mathsf{Rec}(\mathbf{\Sigma}_1, \mathbf{V}, \mathsf{params})$

Figure 6: LWR-based key exchange from AKC, where $\mathbf{K}_1, \mathbf{K}_2 \in \mathbb{Z}_m^{l_A\times l_B}$ and $|\mathbf{K}_1| = |\mathbf{K}_2| = l_A l_B |m|$.

5.1 Security Proof of LWR-Based Key Exchange

Definition 5.1. *A KC or AKC based key exchange protocol from LWR is* secure, *if for any sufficiently large security parameter λ and any PT adversary $\mathcal{A}$, $\left|\Pr[b'=b] - \frac{1}{2}\right|$ is negligible, as defined w.r.t. game G_0 specified in Algorithm 40.*[7]

Algorithm 7 Game G_0

```
 1: $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times n}$
 2: $\mathbf{X}_1 \leftarrow \chi^{n\times l_A}$
 3: $\mathbf{Y}_1 = \lfloor \mathbf{A}\mathbf{X}_1 \rceil_p$
 4: $\mathbf{X}_2 \leftarrow \chi^{n\times l_B}$
 5: $\boldsymbol{\varepsilon} \leftarrow \{-q/2p \ldots q/2p-1\}^{n\times l_A}$
 6: $\mathbf{Y}_2 = \lfloor \mathbf{A}^T\mathbf{X}_2 \rceil_p$
 7: $\mathbf{\Sigma}_2 = \lfloor (\frac{q}{p}\mathbf{Y}_1 + \boldsymbol{\varepsilon})^T\mathbf{X}_2 \rceil_p$    $\triangleright$ $\mathbf{\Sigma}_2 = \mathbf{Y}_1^T\mathbf{X}_2 + \lfloor \boldsymbol{\varepsilon}^T\mathbf{X}_2 \rceil_p = \lfloor (\frac{q}{p}\mathbf{Y}_1 + \boldsymbol{\varepsilon})^T\mathbf{X}_2 \rceil_p$
 8: $\left(\mathbf{K}_2^0, \mathbf{V}\right) \leftarrow \mathsf{Con}(\mathbf{\Sigma}_2, \mathsf{params})$
 9: $\mathbf{K}_2^1 \leftarrow \mathbb{Z}_m^{l_A\times l_B}$
10: $b \leftarrow \{0,1\}$
11: $b' \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{Y}_1, \mathbf{Y}_2, \mathbf{K}_2^b, \mathbf{V})$
```

Before starting to prove the security, we first recall some basic properties of the LWR assumption. The following lemma is derived by a hybrid argument, similar to that of LWE [PVW08, BCD+16].

[7]For presentation simplicity, we simply assume $\mathbf{K}_2^0 \leftarrow \mathbb{Z}_m^{l_A\times l_B}$ when the key exchange protocol is implemented with AKC. However, when the AKC-based protocol is interpreted as a public-key encryption scheme, $\mathbf{K}_2^0$ and $\mathbf{K}_2^1$ correspond to the plaintexts, which are taken independently at random from the same (arbitrary) distribution over $\mathbb{Z}_m^{l_A\times l_B}$.

Lemma 5.1 (LWR problem in the matrix form)**.** *For positive integer parameters $(\lambda, n, q \geq 2, l, t)$, where n, q, l, t all are polynomial in λ satisfying $p|q$, and a distribution χ over $\mathbb{Z}_q$, denote by $L_\chi^{(l,t)}$ the distribution over $\mathbb{Z}_q^{t\times n} \times \mathbb{Z}_p^{t\times l}$ generated by taking $\mathbf{A} \leftarrow \mathbb{Z}_q^{t\times n}, \mathbf{S} \leftarrow \chi^{n\times l}$ and outputting $(\mathbf{A}, \lfloor \mathbf{AS} \rceil_p)$. Then, under the assumption on indistinguishability between $A_{q,\mathbf{s},\chi}$ (with $\mathbf{s} \leftarrow \chi^n$) and $\mathcal{U}(\mathbb{Z}_q^n \times \mathbb{Z}_p)$ within t samples, no PT distinguisher $\mathcal{D}$ can distinguish, with non-negligible probability, between the distribution $L_\chi^{(l,t)}$ and $\mathcal{U}(\mathbb{Z}_q^{t\times n} \times \mathbb{Z}_p^{t\times l})$ for sufficiently large λ.*

Algorithm 8 Game G_0

```
1:  $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times n}$
2:  $\mathbf{X}_1 \leftarrow \chi^{n\times l_A}$
3:  $\mathbf{Y}_1 = \lfloor \mathbf{A}\mathbf{X}_1 \rceil_p$
4:  $\mathbf{X}_2 \leftarrow \chi^{n\times l_B}$
5:  $\boldsymbol{\varepsilon} \leftarrow \{-q/2p \dots q/2p - 1\}^{n\times l_A}$
6:  $\mathbf{Y}_2 = \lfloor \mathbf{A}^T\mathbf{X}_2 \rceil_p$
7:  $\boldsymbol{\Sigma}_2 = \lfloor (\frac{q}{p}\mathbf{Y}_1 + \boldsymbol{\varepsilon})^T \mathbf{X}_2 \rceil_p$
8:  $(\mathbf{K}_2^0, \mathbf{V}) \leftarrow \mathsf{Con}(\boldsymbol{\Sigma}_2, \mathsf{params})$
9:  $\mathbf{K}_2^1 \leftarrow \mathbb{Z}_m^{l_A\times l_B}$
10: $b \leftarrow \{0,1\}$
11: $b' \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{Y}_1, \mathbf{Y}_2, \mathbf{K}_2^b, \mathbf{V})$
```

Algorithm 9 Game G_1

```
1:  $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times n}$
2:  $\mathbf{X}_1 \leftarrow \chi^{n\times l_A}$
3:  $\mathbf{Y}_1 \leftarrow \mathbb{Z}_p^{n\times l_A}$
4:  $\mathbf{X}_2 \leftarrow \chi^{n\times l_B}$
5:  $\boldsymbol{\varepsilon} \leftarrow \{-q/2p \dots q/2p - 1\}^{n\times l_A}$
6:  $\mathbf{Y}_2 = \lfloor \mathbf{A}^T\mathbf{X}_2 \rceil_p$
7:  $\boldsymbol{\Sigma}_2 = \lfloor (\frac{q}{p}\mathbf{Y}_1 + \boldsymbol{\varepsilon})^T \mathbf{X}_2 \rceil_p$
8:  $(\mathbf{K}_2^0, \mathbf{V}) \leftarrow \mathsf{Con}(\boldsymbol{\Sigma}_2, \mathsf{params})$
9:  $\mathbf{K}_2^1 \leftarrow \mathbb{Z}_m^{l_A\times l_B}$
10: $b \leftarrow \{0,1\}$
11: $b' \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{Y}_1, \mathbf{Y}_2, \mathbf{K}_2^b, \mathbf{V})$
```

Algorithm 10 Distinguisher $\mathcal{D}$

```
1:  procedure $\mathcal{D}(\mathbf{A}, \mathbf{B})$        ▷ $\mathbf{A} \in \mathbb{Z}_q^{n\times n}, \mathbf{B} \in \mathbb{Z}_p^{n\times l_A}$
2:      $\mathbf{Y}_1 = \mathbf{B}$
3:      $\mathbf{X}_2 \leftarrow \chi^{n\times l_B}$
4:      $\boldsymbol{\varepsilon} \leftarrow \{-q/2p \dots q/2p - 1\}^{n\times l_A}$
5:      $\mathbf{Y}_2 = \lfloor \mathbf{A}^T\mathbf{X}_2 \rceil_p$
6:      $\boldsymbol{\Sigma}_2 = \lfloor (\frac{q}{p}\mathbf{Y}_1 + \boldsymbol{\varepsilon})^T \mathbf{X}_2 \rceil_p$
7:      $(\mathbf{K}_2^0, \mathbf{V}) \leftarrow \mathsf{Con}(\boldsymbol{\Sigma}_2, \mathsf{params})$
8:      $\mathbf{K}_2^1 \leftarrow \mathbb{Z}_m^{l_A\times l_B}$
9:      $b \leftarrow \{0,1\}$
10:     $b' \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{Y}_1, \mathbf{Y}_2, \mathbf{K}_2^b, \mathbf{V})$
11:     if $b' = b$ then
12:         return 1
13:     else
14:         return 0
15:     end if
16: end procedure
```

Theorem 5.1. *If* (params, Con, Rec) *is a* correct *and* secure *KC or AKC scheme, the key exchange protocol described in Figure 5 is* secure *under the (matrix form of) LWR assumption.*

Proof. The proof is analogous to that in [Pei14,BCD+16]. The general idea is that we construct a sequence of games: G_0, G_1 and G_2, where G_0 is the original game for defining security. In every move from game G_i to G_{i+1}, $0 \leq i \leq 1$, we change a little. All games G_i's share the same PT adversary $\mathcal{A}$, whose goal is to distinguish between the matrices chosen uniformly at random and the matrices generated in the actual key exchange protocol. Denote by T_i, $0 \leq i \leq 2$, the event that $b = b'$ in Game G_i. Our goal is to prove that $\Pr[T_0] < 1/2 + negl$, where $negl$ is a negligible function in λ. For ease of readability, we re-produce game G_0 below. For presentation simplicity, in the subsequent analysis, we always assume the underlying KC or AKC is *correct*. The proof can be trivially extended to the case that correctness holds with *overwhelming* probability (i.e., failure occurs with negligible probability).

Lemma 5.2. $|\Pr[T_0]-\Pr[T_1]| < negl$*, under the indistinguishability between* $L_\chi^{(l_A,n)}$ *and* $\mathcal{U}(\mathbb{Z}_q^{n\times n}\times\mathbb{Z}_p^{n\times l_A})$*.*

Proof. Construct a distinguisher $\mathcal{D}$, in Algorithm 10, who tries to distinguish $L_\chi^{(l_A,n)}$ from $\mathcal{U}(\mathbb{Z}_q^{n\times n}\times\mathbb{Z}_p^{n\times l_A})$.

If $(\mathbf{A},\mathbf{B})$ is subjected to $L_\chi^{(l_A,n)}$, then $\mathcal{D}$ perfectly simulates G_0. Hence, $\Pr\left[\mathcal{D}\left(L_\chi^{(l_A,n)}\right)=1\right]=\Pr[T_0]$. On the other hand, if $(\mathbf{A},\mathbf{B})$ is chosen uniformly at random from $\mathbb{Z}_q^{n\times n}\times\mathbb{Z}_p^{n\times l_A}$, which is denoted as $(\mathbf{A}^{\mathcal{U}},\mathbf{B}^{\mathcal{U}})$, then $\mathcal{D}$ perfectly simulates G_1. So $\Pr[\mathcal{D}(\mathbf{A}^{\mathcal{U}},\mathbf{B}^{\mathcal{U}})=1]=\Pr[T_1]$. Hence, $|\Pr[T_0]-\Pr[T_1]|=\left|\Pr[\mathcal{D}(L_\chi^{(l_A,n)})=1]-\Pr[\mathcal{D}(\mathbf{A}^{\mathcal{U}},\mathbf{B}^{\mathcal{U}})=1]\right|<negl$. □ □

Algorithm 11 Game G_1
```
1: $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times n}$
2: $\mathbf{X}_1, \mathbf{E}_1 \leftarrow \chi^{n\times l_A}$
3: $\mathbf{Y}_1 \leftarrow \mathbb{Z}_q^{n\times l_A}$
4: $\mathbf{X}_2 \leftarrow \chi^{n\times l_B}$
5: $\boldsymbol{\varepsilon} \leftarrow \{-q/2p \ldots q/2p-1\}^{n\times l_A}$
6: $\mathbf{Y}_2 = \lfloor \mathbf{A}^T\mathbf{X}_2 \rceil_p$
7: $\mathbf{\Sigma}_2 = \lfloor (\frac{q}{p}\mathbf{Y}_1+\boldsymbol{\varepsilon})^T\mathbf{X}_2 \rceil_p$
8: $\left(\mathbf{K}_2^0, \mathbf{V}\right) \leftarrow \mathsf{Con}(\mathbf{\Sigma}_2, \mathsf{params})$
9: $\mathbf{K}_2^1 \leftarrow \mathbb{Z}_m^{l_A\times l_B}$
10: $b \leftarrow \{0,1\}$
11: $b' \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{Y}_1, \mathbf{Y}_2, \mathbf{K}_2^b, \mathbf{V})$
```

Algorithm 12 Game G_2
```
1: $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times n}$
2: $\mathbf{X}_1, \mathbf{E}_1 \leftarrow \chi^{n\times l_A}$
3: $\mathbf{Y}_1 \leftarrow \mathbb{Z}_q^{n\times l_A}$
4: $\mathbf{X}_2 \leftarrow \chi^{n\times l_B}$
5: $\boldsymbol{\varepsilon} \leftarrow \{-q/2p \ldots q/2p-1\}^{n\times l_A}$
6: $\mathbf{Y}_2 \leftarrow \mathbb{Z}_p^{n\times l_B}$
7: $\mathbf{\Sigma}_2 \leftarrow \mathbb{Z}_p^{l_A\times l_B}$
8: $\left(\mathbf{K}_2^0, \mathbf{V}\right) \leftarrow \mathsf{Con}(\mathbf{\Sigma}_2, \mathsf{params})$
9: $\mathbf{K}_2^1 \leftarrow \mathbb{Z}_m^{l_A\times l_B}$
10: $b \leftarrow \{0,1\}$
11: $b' \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{Y}_1, \mathbf{Y}_2, \mathbf{K}_2^b, \mathbf{V})$
```

Lemma 5.3. $|\Pr[T_1]-\Pr[T_2]| < negl$*, under the indistinguishability between* $L_\chi^{(l_B,n+l_A)}$ *and* $\mathcal{U}(\mathbb{Z}_q^{(n+l_A)\times n}\times\mathbb{Z}_p^{(n+l_A)\times l_B})$*.*

Proof. As $\mathbf{Y}_1$ and $\boldsymbol{\varepsilon}$ are subjected to uniform distribution in G_1, $\frac{p}{q}\mathbf{Y}_1+\boldsymbol{\varepsilon}$ is subjected to uniform distribution over $\mathbb{Z}_q^{n\times l_A}$. Based on this observation, we construct the following distinguisher $\mathcal{D}'$ presented in Algorithm 13.

First observe that $\mathbf{Y}_1'=(\frac{q}{p}\mathbf{Y}_1+\boldsymbol{\varepsilon})\in\mathbb{Z}_q^{n\times l_A}$ follows the uniform distribution $\mathcal{U}(\mathbb{Z}_q^{n\times l_A})$, where $\mathbf{Y}_1\leftarrow\mathbb{Z}_q^{n\times l_A}$ and $\boldsymbol{\varepsilon}\leftarrow[-q/2p,q/2p-1]^{n\times l_A}$. If $(\mathbf{A}',\mathbf{B})$ is subject to $L_\chi^{(l_B,n+l_A)}$, $\mathbf{A}'\leftarrow\mathbb{Z}_q^{(n+l_A)\times n}$ corresponds to $\mathbf{A}\leftarrow\mathbb{Z}_q^{n\times n}$ and $\mathbf{Y}_1'=\frac{q}{p}\mathbf{Y}_1+\boldsymbol{\varepsilon}$ in G_1; And $\mathbf{S}\leftarrow\chi^{n\times l_B}$ in generating $(\mathbf{A}',\mathbf{B})$ corresponds to $\mathbf{X}_2\leftarrow\chi^{n\times l_B}$ in G_1. In this case, we re-write

$$\mathbf{B}=\lfloor\mathbf{A}'\mathbf{S}\rceil_p=\left\lfloor\begin{pmatrix}\mathbf{A}^T\\ \mathbf{Y}_1'^T\end{pmatrix}\mathbf{X}_2\right\rceil_p$$
$$=\begin{pmatrix}\lfloor\mathbf{A}^T\mathbf{X}_2\rceil_p\\ \lfloor\mathbf{Y}_1'^T\mathbf{X}_2\rceil_p\end{pmatrix}=\begin{pmatrix}\mathbf{Y}_2\\ \mathbf{\Sigma}_2\end{pmatrix}$$

Hence $\Pr\left[\mathcal{D}'\left(L_\chi^{(l_B,n+l_A)}\right)=1\right]=\Pr[T_1]$.

On the other hand, if $(\mathbf{A}', \mathbf{B})$ is subject to uniform distribution $\mathcal{U}(\mathbb{Z}_q^{(n+l_A)\times n} \times \mathbb{Z}_p^{(n+l_A)\times l_B})$, then $\mathbf{A}, \mathbf{Y}'_1, \mathbf{Y}_2, \mathbf{\Sigma}_2$ all are also uniformly random; So, the view of $\mathcal{D}'$ in this case is the same as that in game G_2. Hence, $\Pr[\mathcal{D}'(\mathbf{A}', \mathbf{B}) = 1] = \Pr[T_2]$ in this case. Then, $|\Pr[T_1] - \Pr[T_2]| = |\Pr[\mathcal{D}'(L_\chi^{(l_B, n+l_A)}) = 1] - \Pr[\mathcal{D}'(\mathcal{U}(\mathbb{Z}_q^{(n+l_A)\times n} \times \mathbb{Z}_p^{(n+l_A)\times l_B})) = 1]| < negl$. □ □

Lemma 5.4. *If the underlying KC or AKC is* secure, $\Pr[T2] = \frac{1}{2}$.

Proof. Note that, in Game G_2, for any $1 \le i \le l_A$ and $1 \le j \le l_B$, $(\mathbf{K}_2^0[i,j], \mathbf{V}[i,j])$ only depends on $\mathbf{\Sigma}_2[i,j]$, and $\mathbf{\Sigma}_2$ is subject to uniform distribution. By the *security* of KC, we have that, for each pair (i,j), $\mathbf{K}_2^0[i,j]$ and $\mathbf{V}[i,j]$ are independent, and $\mathbf{K}_2^0[i,j]$ is uniform distributed. Hence, $\mathbf{K}_2^0$ and $\mathbf{V}$ are independent, and $\mathbf{K}_2^0$ is uniformly distributed, which implies that $\Pr[T_2] = 1/2$. □ □

This finishes the proof of Theorem 5.1. □ □

Algorithm 13 Distinguisher $\mathcal{D}'$

```
1: procedure $\mathcal{D}'(\mathbf{A}', \mathbf{B})$ where $\mathbf{A}' \in \mathbb{Z}_q^{(n+l_A)\times n}, \mathbf{B} \in \mathbb{Z}_p^{(n+l_A)\times l_B}$
2:     Denote $\mathbf{A}' = \begin{pmatrix} \mathbf{A}^T \\ \mathbf{Y}_1'^T \end{pmatrix}$    ▷ $\mathbf{A} \in \mathbb{Z}_q^{n\times n}, \mathbf{Y}_1'^T = (\frac{q}{p}\mathbf{Y}_1 + \boldsymbol{\varepsilon})^T \in \mathbb{Z}_q^{l_A\times n}$
3:     Denote $\mathbf{B} = \begin{pmatrix} \mathbf{Y}_2 \\ \mathbf{\Sigma}_2 \end{pmatrix}$    ▷ $\mathbf{Y}_2 \in \mathbb{Z}_p^{n\times l_B}, \mathbf{\Sigma}_2 \in \mathbb{Z}_p^{l_A\times l_B}$
4:     $(\mathbf{K}_2^0, \mathbf{V}) \leftarrow \mathsf{Con}(\mathbf{\Sigma}_2, \mathsf{params})$
5:     $\mathbf{K}_2^1 \leftarrow \mathbb{Z}_m^{l_A\times l_B}$
6:     $b \leftarrow \{0,1\}$
7:     $b' \leftarrow \mathcal{A}(\mathbf{A}, \lfloor \mathbf{Y}_1' \rceil_p, \mathbf{Y}_2, \mathbf{K}_2^b, \mathbf{V})$
8:     if $b' = b$ then
9:         return 1
10:    else
11:        return 0
12:    end if
13: end procedure
```

5.2 Analysis of Correctness and Error Rate

For any integer x, let $\{x\}_p$ denote $x - \frac{q}{p}\lfloor x \rceil_p$, where $\lfloor x \rceil_p = \lfloor \frac{p}{q} x \rceil$. Then, for any integer x, $\{x\}_p \in [-q/2p, q/2p - 1]$, hence $\{x\}_p$ can be naturally regarded as an element in $\mathbb{Z}_{q/p}$. In fact, $\{x\}_p$ is equal to $x \bmod q/p$, where the result is represented in $[-q/2p, q/2p-1]$. When the notation $\{\cdot\}_p$ is applied to a matrix, it means $\{\cdot\}_p$ applies to every element of the matrix respectively.

We have $\mathbf{\Sigma}_2 = \mathbf{Y}_1^T\mathbf{X}_2 + \lfloor \boldsymbol{\varepsilon}^T\mathbf{X}_2 \rceil_p = \lfloor \mathbf{A}\mathbf{X}_1 \rceil_p^T \mathbf{X}_2 + \lfloor \boldsymbol{\varepsilon}^T\mathbf{X}_2 \rceil_p = \frac{p}{q}(\mathbf{A}\mathbf{X}_1 - \{\mathbf{A}\mathbf{X}_1\}_p)^T\mathbf{X}_2 + \lfloor \boldsymbol{\varepsilon}^T\mathbf{X}_2 \rceil_p$. And $\mathbf{\Sigma}_1 = \mathbf{X}_1^T\mathbf{Y}_2 = \mathbf{X}_1^T\lfloor \mathbf{A}^T\mathbf{X}_2 \rceil_p = \frac{p}{q}(\mathbf{X}_1^T\mathbf{A}^T\mathbf{X}_2 - \mathbf{X}_1^T\{\mathbf{A}^T\mathbf{X}_2\}_p)$. Hence,

$$\begin{aligned}\mathbf{\Sigma}_2 - \mathbf{\Sigma}_1 &= \frac{p}{q}(\mathbf{X}_1^T\{\mathbf{A}^T\mathbf{X}_2\}_p - \{\mathbf{A}\mathbf{X}_1\}_p^T\mathbf{X}_2) + \lfloor \varepsilon^T\mathbf{X}_2 \rceil_p \mod p \\ &= \left\lfloor \frac{p}{q}(\mathbf{X}_1^T\{\mathbf{A}^T\mathbf{X}_2\}_p - \{\mathbf{A}\mathbf{X}_1\}_p^T\mathbf{X}_2 + \varepsilon^T\mathbf{X}_2) \right\rceil \mod p\end{aligned}$$

The general idea is that $\mathbf{X}_1, \mathbf{X}_2, \boldsymbol{\varepsilon}, \{\mathbf{A}^T\mathbf{X}_2\}_p$ and $\{\mathbf{A}\mathbf{X}_1\}_p$ are small enough, so that $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$ are close. If $|\boldsymbol{\Sigma}_1 - \boldsymbol{\Sigma}_2|_p \le d$, the *correctness* of the underlying KC guarantees $\mathbf{K}_1 = \mathbf{K}_2$. For given concrete parameters, we numerically derive the probability of $|\boldsymbol{\Sigma}_2 - \boldsymbol{\Sigma}_1|_p > d$ by numerically calculating the distribution of $\mathbf{X}_1^T\{\mathbf{A}^T\mathbf{X}_2\}_p - (\{\mathbf{A}\mathbf{X}_1\}_p^T\mathbf{X}_2 - \varepsilon^T\mathbf{X}_2)$ for the case of $l_A = l_B = 1$, then applying the *union bound*. The independency between variables indicated by the following Theorem 5.2 can greatly simplify the calculation.

Let $\mathsf{Inv}(\mathbf{X}_1, \mathbf{X}_2)$ denote the event that there exist invertible elements of ring $\mathbb{Z}_{q/p}$ in both vectors $\mathbf{X}_1$ and $\mathbf{X}_2$. $\mathsf{Inv}(\mathbf{X}_1, \mathbf{X}_2)$ happens with *overwhelming* probability in our application.

Lemma 5.5. *Consider the case of $l_A = l_B = 1$. For any $a \in \mathbb{Z}_{q/p}, \mathbf{x} \in \mathbb{Z}_{q/p}^n$, denote $S_{\mathbf{x},a} = \{\mathbf{y} \in \mathbb{Z}_{q/p}^n \mid \mathbf{x}^T\mathbf{y} \bmod (q/p) = a\}$. For any fixed $a \in \mathbb{Z}_{q/p}$, conditioned on $\mathsf{Inv}(\mathbf{X}_1, \mathbf{X}_2)$ and $\mathbf{X}_1^T\mathbf{A}^T\mathbf{X}_2 \bmod (q/p) = a$, the random vectors $\{\mathbf{A}^T\mathbf{X}_2\}_p$ and $\{\mathbf{A}\mathbf{X}_1\}_p$ are independent, and are subjected to uniform distribution over $S_{\mathbf{X}_1,a}, S_{\mathbf{X}_2,a}$ respectively.*

Proof. Under the condition of $\mathsf{Inv}(\mathbf{X}_1, \mathbf{X}_2)$, for any fixed $\mathbf{X}_1$ and $\mathbf{X}_2$, define the map $\phi_{\mathbf{X}_1,\mathbf{X}_2}$: $\mathbb{Z}_q^{n\times n} \to \mathbb{Z}_{q/p}^n \times \mathbb{Z}_{q/p}^n$, such that $\mathbf{A} \mapsto (\{\mathbf{A}\mathbf{X}_1\}_p, \{\mathbf{A}^T\mathbf{X}_2\}_p)$.

We shall prove that the image of $\phi_{\mathbf{X}_1,\mathbf{X}_2}$ is $S = \{(\mathbf{y}_1, \mathbf{y}_2) \in \mathbb{Z}_{q/p}^n \times \mathbb{Z}_{q/p}^n \mid \mathbf{X}_2^T\mathbf{y}_1 = \mathbf{X}_1^T\mathbf{y}_2 \bmod (q/p)\}$. Denote $\mathbf{X}_1 = (x_1, \mathbf{X}_1'^T)^T$ and $\mathbf{y}_2 = (y_2, \mathbf{y}_2'^T)^T$. Without loss of generality, we assume x_1 is invertible in the ring $\mathbb{Z}_{q/p}$. For any $(\mathbf{y}_1, \mathbf{y}_2) \in S$, we need to find an $\mathbf{A}$ such that $\phi_{\mathbf{X}_1,\mathbf{X}_2}(\mathbf{A}) = (\mathbf{y}_1, \mathbf{y}_2)$.

From the condition $\mathsf{Inv}(\mathbf{X}_1, \mathbf{X}_2)$, we know that there exists an $\mathbf{A}' \in \mathbb{Z}^{(n-1)\times n}$ such that $\{\mathbf{A}'\mathbf{X}_2\}_p = \mathbf{y}_2'$. Then, we let $\mathbf{a}_1 = x_1^{-1}(\mathbf{y}_1 - \mathbf{A}'^T\mathbf{X}_1') \bmod (q/p)$, and $\mathbf{A} = (\mathbf{a}_1, \mathbf{A}'^T)$. Now we check that $\phi_{\mathbf{X}_1,\mathbf{X}_2}(\mathbf{A}) = (\mathbf{y}_1, \mathbf{y}_2)$.

$$\{\mathbf{A}\mathbf{X}_1\}_p = \left\{\begin{pmatrix}\mathbf{a}_1 & \mathbf{A}'^T\end{pmatrix}\begin{pmatrix}x_1\\ \mathbf{x}_1'\end{pmatrix}\right\}_p = \{x_1\mathbf{a}_1 + \mathbf{A}'^T\mathbf{X}_1'\}_p = \mathbf{y}_1$$

$$\begin{aligned}\{\mathbf{A}^T\mathbf{X}_2\}_p &= \left\{\begin{pmatrix}\mathbf{a}_1^T\\ \mathbf{A}'\end{pmatrix}\mathbf{X}_2\right\}_p = \left\{\begin{pmatrix}\mathbf{a}_1^T\mathbf{X}_2\\ \mathbf{A}'\mathbf{X}_2\end{pmatrix}\right\}_p = \left\{\begin{pmatrix}x_1^{-1}(\mathbf{y}_1^T - \mathbf{X}_1'^T\mathbf{A})\mathbf{X}_2\\ \mathbf{A}'\mathbf{X}_2\end{pmatrix}\right\}_p\\ &= \left\{\begin{pmatrix}x_1^{-1}(\mathbf{X}_1^T\mathbf{y}_2 - \mathbf{X}_1'^T\mathbf{y}_2')\\ \mathbf{y}_2'\end{pmatrix}\right\}_p = \left\{\begin{pmatrix}y_2\\ \mathbf{y}_2'\end{pmatrix}\right\}_p = \mathbf{y}_2\end{aligned}$$

Hence, if we treat $\mathbb{Z}_q^{n\times n}$ and S as $\mathbb{Z}$-modules, then $\phi_{\mathbf{X}_1,\mathbf{X}_2} : \mathbb{Z}_q^{n\times n} \to S$ is a surjective homomorphism. Then, for any fixed $(\mathbf{X}_1, \mathbf{X}_2)$, $(\{\mathbf{A}\mathbf{X}_1\}_p, \{\mathbf{A}^T\mathbf{X}_2\}_p)$ is uniformly distributed over S. This completes the proof. □ □

Theorem 5.2. *Under the condition $\mathsf{Inv}(\mathbf{X}_1, \mathbf{X}_2)$, the following two distributions are identical:*

- $(a, \mathbf{X}_1, \mathbf{X}_2, \{\mathbf{A}\mathbf{X}_1\}_p, \{\mathbf{A}^T\mathbf{X}_2\}_p)$, *where* $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times n}$, $\mathbf{X}_1 \leftarrow \chi^n$, $\mathbf{X}_2 \leftarrow \chi^n$, *and* $a = \mathbf{X}_1^T\mathbf{A}^T\mathbf{X}_2 \bmod (q/p)$.
- $(a, \mathbf{X}_1, \mathbf{X}_2, \mathbf{y}_1, \mathbf{y}_2)$, *where* $a \leftarrow \mathbb{Z}_{q/p}, \mathbf{X}_1 \leftarrow \chi^n$, $\mathbf{X}_2 \leftarrow \chi^n$, $\mathbf{y}_1 \leftarrow S_{\mathbf{X}_2,a}$, *and* $\mathbf{y}_2 \leftarrow S_{\mathbf{X}_1,a}$.

Proof. For any $\tilde{a} \in \mathbb{Z}_{q/p}$, $\tilde{\mathbf{X}}_1, \tilde{\mathbf{X}}_2 \in \mathsf{Supp}(\chi^n)$, $\tilde{\mathbf{y}}_1, \tilde{\mathbf{y}}_2 \in \mathbb{Z}_{q/p}^n$, we have

$$\begin{aligned}&\Pr[a = \tilde{a}, \mathbf{X}_1 = \tilde{\mathbf{X}}_1, \mathbf{X}_2 = \tilde{\mathbf{X}}_2, \{\mathbf{A}\mathbf{X}_1\}_p = \tilde{\mathbf{y}}_1, \{\mathbf{A}^T\mathbf{X}_2\}_p = \tilde{\mathbf{y}}_2 \mid \mathsf{Inv}(\mathbf{X}_1, \mathbf{X}_2)]\\ =&\Pr[\{\mathbf{A}\mathbf{X}_1\}_p = \tilde{\mathbf{y}}_1, \{\mathbf{A}^T\mathbf{X}_2\}_p = \tilde{\mathbf{y}}_2 \mid a = \tilde{a}, \mathbf{X}_1 = \tilde{\mathbf{X}}_1, \mathbf{X}_2 = \tilde{\mathbf{X}}_2, \mathsf{Inv}(\mathbf{X}_1, \mathbf{X}_2)]\end{aligned}$$

$$\Pr[a = \tilde{a}, \mathbf{X}_1 = \tilde{\mathbf{X}}_1, \mathbf{X}_2 = \tilde{\mathbf{X}}_2 \mid \mathsf{Inv}(\mathbf{X}_1, \mathbf{X}_2)]$$

From Lemma 5.5, the first term equals to $\Pr[\mathbf{y}_1 \leftarrow S_{\tilde{\mathbf{X}}_2,\tilde{a}}; \mathbf{y}_2 \leftarrow S_{\tilde{\mathbf{X}}_1,\tilde{a}} : \mathbf{y}_1 = \tilde{\mathbf{y}}_1, \mathbf{y}_2 = \tilde{\mathbf{y}}_2 \mid a = \tilde{a}, \mathbf{X}_1 = \tilde{\mathbf{X}}_1, \mathbf{X}_2 = \tilde{\mathbf{X}}_2, \mathsf{Inv}(\mathbf{X}_1, \mathbf{X}_2)]$.

For the second term, we shall prove that a is independent of $(\mathbf{X}_1, \mathbf{X}_2)$, and is uniformly distributed over $\mathbb{Z}_{q/p}$. Under the condition of $\mathsf{Inv}(\mathbf{X}_1, \mathbf{X}_2)$, the map $\mathbb{Z}_q^{n\times n} \to \mathbb{Z}_{q/p}$, such that $\mathbf{A} \mapsto \mathbf{X}_1^T\mathbf{A}^T\mathbf{X}_2 \bmod (q/p)$, is a surjective homomorphism between the two $\mathbb{Z}$-modules. Then, $\Pr[a = \tilde{a} \mid \mathbf{X}_1 = \tilde{\mathbf{X}}_1, \mathbf{X}_2 = \tilde{\mathbf{X}}_2, \mathsf{Inv}(\mathbf{X}_1, \mathbf{X}_2)] = p/q$. Hence, under the condition of $\mathsf{Inv}(\mathbf{X}_1, \mathbf{X}_2)$, a is independent of $(\mathbf{X}_1, \mathbf{X}_2)$, and is distributed uniformly at random. So the two ways of sampling result in the same distribution. □ □

We design and implement the following algorithm to numerically calculate the distribution of $\mathbf{\Sigma}_2 - \mathbf{\Sigma}_1$ efficiently. For any $c_1, c_2 \in \mathbb{Z}_q, a \in \mathbb{Z}_{q/p}$, we numerically calculate $\Pr[\mathbf{X}_1^T\{\mathbf{A}^T\mathbf{X}_2\}_p = c_1]$ and $\Pr[\{\mathbf{A}\mathbf{X}_1\}_p^T\mathbf{X}_2 - \boldsymbol{\varepsilon}^T\mathbf{X}_2 = c_2, \mathbf{X}_1^T\mathbf{A}^T\mathbf{X}_2 \bmod (q/p) = a]$, then derive the distribution of $\mathbf{\Sigma}_2 - \mathbf{\Sigma}_1$.

As $\mathsf{Inv}(\mathbf{X}_1, \mathbf{X}_2)$ occurs with *overwhelming* probability, for any event E, we have $|\Pr[E] - \Pr[E|\mathsf{Inv}(\mathbf{X}_1, \mathbf{X}_2)]| < negl$. For simplicity, we ignore the effect of $\mathsf{Inv}(\mathbf{X}_1, \mathbf{X}_2)$ in the following calculations. By Theorem 5.2, $\Pr[\mathbf{X}_1^T\{\mathbf{A}^T\mathbf{X}_2\}_p = c_1] = \Pr[\mathbf{X}_1 \leftarrow \chi^n, \mathbf{y}_2 \leftarrow \mathbb{Z}_{q/p}^n; \mathbf{X}_1^T\mathbf{y}_2 = c_1]$. This probability can be numerically calculated by computer programs. The probability $\Pr[\{\mathbf{A}\mathbf{X}_1\}_p^T\mathbf{X}_2 - \boldsymbol{\varepsilon}^T\mathbf{X}_2 = c_2, \mathbf{X}_1^T\mathbf{A}^T\mathbf{X}_2 \bmod (q/p) = a]$ can also be calculated by the similar way. Then, for arbitrary $c \in \mathbb{Z}_q$,

$$\begin{aligned}
&\Pr[\mathbf{\Sigma}_1 - \mathbf{\Sigma}_2 = c] = \Pr[\mathbf{X}_1^T\{\mathbf{A}^T\mathbf{X}_2\}_p - \{\mathbf{A}\mathbf{X}_1\}_p^T\mathbf{X}_2 + \boldsymbol{\varepsilon}^T\mathbf{X}_2 = c] \\
&= \sum_{\substack{c_1 - c_2 = c \\ a \in \mathbb{Z}_{q/p}}} \Pr[\mathbf{X}_1^T\{\mathbf{A}^T\mathbf{X}_2\}_p = c_1, \{\mathbf{A}\mathbf{X}_1\}_p^T\mathbf{X}_2 - \boldsymbol{\varepsilon}^T\mathbf{X}_2 = c_2 | \mathbf{X}_1^T\mathbf{A}^T\mathbf{X}_2 \bmod (q/p) = a] \cdot \Pr[\mathbf{X}_1^T\mathbf{A}^T\mathbf{X}_2 \bmod (q/p) = a] \\
&= \sum_{\substack{c_1 - c_2 = c \\ a \in \mathbb{Z}_{q/p}}} \Pr[\mathbf{X}_1^T\{\mathbf{A}^T\mathbf{X}_2\}_p = c_1 | \mathbf{X}_1^T\mathbf{A}^T\mathbf{X}_2 \bmod (q/p) = a] \cdot \Pr[\{\mathbf{A}\mathbf{X}_1\}_p^T\mathbf{X}_2 - \boldsymbol{\varepsilon}^T\mathbf{X}_2 = c_2 | \mathbf{X}_1^T\mathbf{A}^T\mathbf{X}_2 \bmod (q/p) = a] \Pr[\mathbf{X}_1^T\mathbf{A}^T\mathbf{X}_2 \bmod (q/p) = a] \\
&= \sum_{\substack{a \in \mathbb{Z}_{q/p} \\ c_1 - c_2 = c}} \frac{\Pr[\mathbf{X}_1^T\{\mathbf{A}^T\mathbf{X}_2\}_p = c_1, c_1 \bmod (q/p) = a] \Pr[\{\mathbf{A}\mathbf{X}_1\}_p^T\mathbf{X}_2 - \boldsymbol{\varepsilon}^T\mathbf{X}_2 = c_2, \mathbf{X}_1^T\mathbf{A}^T\mathbf{X}_2 \bmod (q/p) = a]}{\Pr[\mathbf{X}_1^T\mathbf{A}^T\mathbf{X}_2 \bmod (q/p) = a]} \\
&= \sum_{\substack{a \in \mathbb{Z}_{q/p} \\ c_1 - c_2 = c \\ c_1 \bmod (q/p) = a}} \frac{\Pr[\mathbf{X}_1^T\{\mathbf{A}^T\mathbf{X}_2\}_p = c_1] \Pr[\{\mathbf{A}\mathbf{X}_1\}_p^T\mathbf{X}_2 - \boldsymbol{\varepsilon}^T\mathbf{X}_2 = c_2, \mathbf{X}_1^T\mathbf{A}^T\mathbf{X}_2 \bmod (q/p) = a]}{\Pr[\mathbf{X}_1^T\mathbf{A}^T\mathbf{X}_2 \bmod (q/p) = a]}
\end{aligned}$$

By Theorem 5.2, conditioned on $\mathsf{Inv}(\mathbf{X}_1, \mathbf{X}_2)$ and $\mathbf{X}_1^T\mathbf{A}^T\mathbf{X}_2 \bmod (q/p) = a$, $\mathbf{X}_1^T\{\mathbf{A}^T\mathbf{X}_2\}_p$ is independent of $\{\mathbf{A}\mathbf{X}_1\}_p^T\mathbf{X}_2 - \boldsymbol{\varepsilon}^T\mathbf{X}_2$, which implies the second equality. Our code and scripts are available from Github `http://github.com/OKCN`.

5.3 Parameter Selection and Evaluation

It is suggested in [ADPS16, BCD+16] that rounded Gaussian distribution can be replaced by discrete distribution that is very close to rounded Gaussian in the sense of Rényi divergence [BLL+15].

Definition 5.2 ([BLL+15]). *For two discrete distributions P, Q satisfying $\mathsf{Supp}(P) \subseteq \mathsf{Supp}(Q)$, their a-order Rényi divergence is $R_a(P||Q) = \left(\sum_{x\in\mathsf{Supp}(P)} \frac{P(x)^a}{Q(x)^{a-1}}\right)^{\frac{1}{a-1}}$.*

Lemma 5.6 ([BLL+15]). *Letting $a > 1$, P and Q are two discrete distributions satisfying* $\mathsf{Supp}(P) \subseteq \mathsf{Supp}(Q)$*, then we have*

Multiplicativity: *Let P and Q be two distributions of random variable (Y_1, Y_2). For $i \in \{1, 2\}$, let P_i and Q_i be the margin distribution of Y_i over P and Q respectively. If Y_1 and Y_2, under P and Q respectively, are independent, then $R_a(P||Q) = R_a(P_1||Q_1) \cdot R_a(P_2||Q_2)$.*

Probability Preservation: *Let $A \subseteq \mathsf{Supp}(Q)$ be an event, then* $Q(A) \geq P(A)^{\frac{a}{a-1}} / R_a(P||Q)$.

Note that, when the underlying key derivation function KDF is modelled as a random oracle (as in [BCD+16, ADPS16]), an attacker is considered to be successful only if it can recover the entire consensus bits. Denote by E the event that a PT attacker can successfully and entirely recover the bits of $\mathbf{K}_1 = \mathbf{K}_2$. By Lemma 5.6, we have that $\Pr_{\text{rounded Gaussian}}[E] > \Pr_{\text{discrete}}[E]^{a/(a-1)} / R_a^{n\cdot(l_A+l_B)+l_A\cdot l_B}(\chi||\bar{\phi})$, where $\bar{\phi}$ is the rounded Gaussian distribution, and χ is the discrete distribution.

5.3.1 Proposed Parameters

dist.	bits	var.	probability of 0	±1	±2	±3	±4	±5	±6	order	divergence
D_R	16	2.00	18110	14249	6938	2090	389	44	3	500.0	1.0000270
D_P	16	1.40	21456	15326	5580	1033	97	4	0	500.0	1.0000277

Table 5: Discrete distributions of every component in the LWR secret. We choose the standard variances large enough to prevent potential combinational attacks.

	n	q	p	l	m	g	distr.	bw.	err.	$\|\mathbf{K}\|$
Recommended	672	2^{15}	2^{12}	8	2^4	2^8	D_R	16.19	2^{-30}	256
Paranoid	832	2^{15}	2^{12}	8	2^4	2^8	D_P	20.03	2^{-34}	256

Table 6: Parameters for LWR-Based key exchange. "bw." refers to the bandwidth in kilo-bytes. "err." refers to the overall error rate that is calculated by the algorithm developed in Section 5.2. "$|\mathbf{K}|$" refers to the length of consensus bits.

5.3.2 Security Estimation

Similar to [ADPS16, BCD+16, CKLS16], we only consider the primal and dual attacks [CN11, SE94] adapted to the LWR problem, which are briefly reviewed in Appendix E. Recently, Albrecht showed new variants against LWE with small secret [A17]. But as noted in [A17], it does not violate the concrete security estimation of Frodo [BCD+16] and NewHope [ADPS16] as the security evaluation in these works are very conservative.

We aim at providing parameter sets for long term security, and estimate the concrete security *in a more conservative way* than [APS15] from the defender's point of view. We first consider the attacks of LWE whose secret and noise have different variances. Then, we treat the LWR problem as a special LWE problem whose noise is uniformly distributed over $[-q/2p, q/2p - 1]$. In our security estimation, we simply ignore the difference between the discrete distribution and the rounded Gaussian, on the following grounds: the dual attack and the primal attack only concern about the standard deviation, and the Rényi divergence between the two distributions is very small.

Scheme	Attack	m'	b	C	Q	P
Recommended	Primal	665	459	143	**131**	104
	Dual	633	456	142	**130**	103
Paranoid	Primal	768	584	180	164	**130**
	Dual	746	580	179	163	**129**

Table 7: Security estimation of the parameters described in Table 6. "C, Q, P" stand for "Classical, Quantum, Plausible" respectively. Numbers under these columns are the binary logarithm of running time of the corresponding attacks. Numbers under "m', b" are the best parameters for the attacks.

6 LWE-Based Key Exchange from KC and AKC

In this section, following the protocol structure in [Pei14, ADPS16, BCD$^+$16], we present the applications of OKCN and AKCN to key exchange protocols based on LWE.

Denote by $(\lambda, n, q, \chi, KC, l_A, l_B, t)$ the underlying parameters, where λ is the security parameter, $q \geq 2$, n, l_A and l_B are positive integers that are polynomial in λ (for protocol symmetry, l_A and l_B are usually set to be equal and are actually small constant). To save bandwidth, we chop off t least significant bits of $\mathbf{Y}_2$ before sending it to Alice. Of course, we can chop off some least significant bits from both $\mathbf{Y}_1$ and $\mathbf{Y}_2$. We mainly consider chopping off least significant bits from $\mathbf{Y}_2$, as we want to optimize the ciphertext size when LWE-based KE is used for public-key encryption.

Let $KC = (\mathsf{params}, \mathsf{Con}, \mathsf{Rec})$ be a *correct* and *secure* KC scheme, where params is set to be (q, g, m, d). The KC-based key exchange protocol from LWE is depicted in Figure 7, and the actual session-key is derived from $\mathbf{K}_1$ and $\mathbf{K}_2$ via some key derivation function KDF. There, for presentation simplicity, the Con and Rec functions are applied to matrices, meaning they are applied to each of the coordinates separately. Note that $2^t\mathbf{Y}_2'+2^{t-1}\mathbf{1}$ is an approximation of $\mathbf{Y}_2$, so we have $\mathbf{\Sigma}_1 \approx \mathbf{X}_1^T\mathbf{Y}_2 = \mathbf{X}_1^T\mathbf{A}^T\mathbf{X}_2 + \mathbf{X}_1^T\mathbf{E}_2$, $\mathbf{\Sigma}_2 = \mathbf{Y}_1^T\mathbf{X}_2 + \mathbf{E}_\sigma = \mathbf{X}_1^T\mathbf{A}^T\mathbf{X}_2 + \mathbf{E}_1^T\mathbf{X}_2 + \mathbf{E}_\sigma$.[8] As we choose $\mathbf{X}_1, \mathbf{X}_2, \mathbf{E}_1, \mathbf{E}_2, \mathbf{E}_\sigma$ according to a small noise distribution χ, the main part of $\mathbf{\Sigma}_1$ and that of $\mathbf{\Sigma}_2$ are the same $\mathbf{X}_1^T\mathbf{A}^T\mathbf{X}_2$. Hence, the corresponding coordinates of $\mathbf{\Sigma}_1$ and $\mathbf{\Sigma}_2$ are close in the sense of $|\cdot|_q$, from which some key consensus can be reached. The failure probability depends upon the number of bits we cut off t, the underlying distribution χ and the distance parameter d, which will be analyzed in detail in subsequent sections. In the following security definition and analysis, we simply assume that the output of the PRG Gen is truly random. For presentation simplicity, we have described the LWE-based key exchange protocol from a KC scheme. But it can be straightforwardly adapted to work on any correct and secure AKC scheme, which is also explicitly specified in Figure 8.

By a straightforward adaption (actually simplification) of the security proof of LWR-based key exchange protocol in Section 5.1, we have the following theorem. The detailed proof of Theorem 6.1 is presented in Appendix G.

[8] An alternative (equivalent) method is to set $\mathbf{Y}_2' = \lfloor \mathbf{Y}_2/2^t \rceil$, and in this case $\mathbf{\Sigma}_1 = \mathbf{X}_1^T 2^t \mathbf{Y}_2'$.

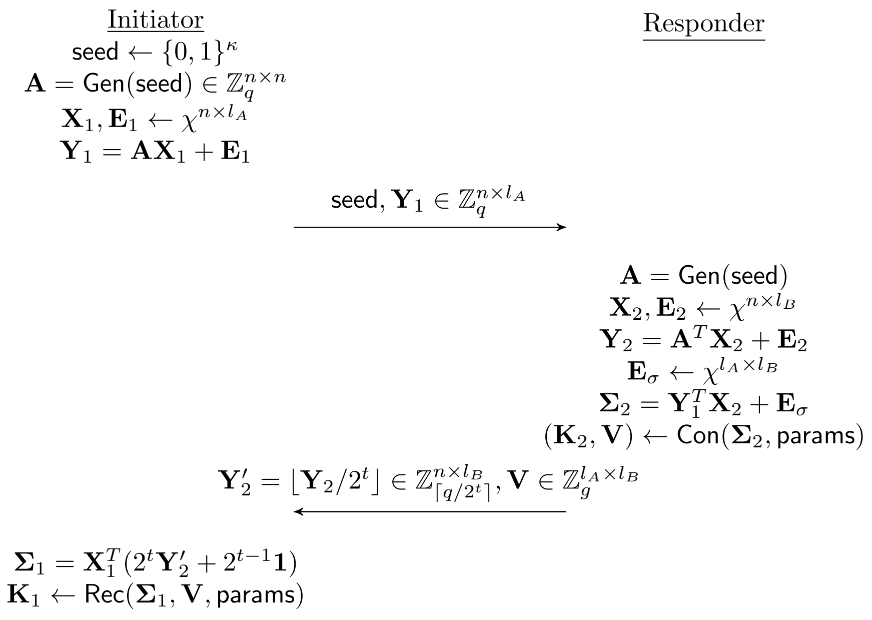
Initiator
Responder
$\mathsf{seed} \leftarrow \{0,1\}^\kappa$
$\mathbf{A} = \mathsf{Gen}(\mathsf{seed}) \in \mathbb{Z}_q^{n\times n}$
$\mathbf{X}_1, \mathbf{E}_1 \leftarrow \chi^{n\times l_A}$
$\mathbf{Y}_1 = \mathbf{A}\mathbf{X}_1 + \mathbf{E}_1$
$\mathsf{seed}, \mathbf{Y}_1 \in \mathbb{Z}_q^{n\times l_A}$
$\mathbf{A} = \mathsf{Gen}(\mathsf{seed})$
$\mathbf{X}_2, \mathbf{E}_2 \leftarrow \chi^{n\times l_B}$
$\mathbf{Y}_2 = \mathbf{A}^T\mathbf{X}_2 + \mathbf{E}_2$
$\mathbf{E}_\sigma \leftarrow \chi^{l_A\times l_B}$
$\mathbf{\Sigma}_2 = \mathbf{Y}_1^T\mathbf{X}_2 + \mathbf{E}_\sigma$
$(\mathbf{K}_2, \mathbf{V}) \leftarrow \mathsf{Con}(\mathbf{\Sigma}_2, \mathsf{params})$
$\mathbf{Y}_2' = \lfloor \mathbf{Y}_2/2^t \rfloor \in \mathbb{Z}_{\lceil q/2^t \rceil}^{n\times l_B}, \mathbf{V} \in \mathbb{Z}_g^{l_A\times l_B}$
$\mathbf{\Sigma}_1 = \mathbf{X}_1^T(2^t\mathbf{Y}_2' + 2^{t-1}\mathbf{1})$
$\mathbf{K}_1 \leftarrow \mathsf{Rec}(\mathbf{\Sigma}_1, \mathbf{V}, \mathsf{params})$

Figure 7: LWE-based key exchange from KC, where $\mathbf{K}_1, \mathbf{K}_2 \in \mathbb{Z}_m^{l_A\times l_B}$ and $|\mathbf{K}_1| = |\mathbf{K}_2| = l_A l_B |m|$. $\mathbf{1}$ refers to the matrix which every elements are 1.

Theorem 6.1. *If* ($\mathsf{params}, \mathsf{Con}, \mathsf{Rec}$) *is a* correct *and* secure *KC or AKC scheme, the key exchange protocol described in Figure 7 is* secure *under the (matrix form of) LWE assumption [PVW08, BCD+16].*

6.1 Noise Distributions and Correctness

For a *correct* KC with parameter d, if the distance of corresponding elements of $\mathbf{\Sigma}_1$ and $\mathbf{\Sigma}_2$ is less than d in the sense of $|\cdot|_q$, then the scheme depicted in Figure 7 is correct. Denote $\varepsilon(\mathbf{Y}_2) = 2^t\lfloor \mathbf{Y}_2/2^t \rfloor + 2^{t-1}\mathbf{1} - \mathbf{Y}_2$. Then

$$\begin{aligned}
\mathbf{\Sigma}_1 - \mathbf{\Sigma}_2 &= \mathbf{X}_1^T(2^t\mathbf{Y}_2' + 2^{t-1}\mathbf{1}) - \mathbf{Y}_1^T\mathbf{X}_2 - \mathbf{E}_\sigma \\
&= \mathbf{X}_1^T(\mathbf{Y}_2 + \varepsilon(\mathbf{Y}_2)) - \mathbf{Y}_1^T\mathbf{X}_2 - \mathbf{E}_\sigma \\
&= \mathbf{X}_1^T(\mathbf{A}^T\mathbf{X}_2 + \mathbf{E}_2 + \varepsilon(\mathbf{Y}_2)) - (\mathbf{A}\mathbf{X}_1 + \mathbf{E}_1)^T\mathbf{X}_2 - \mathbf{E}_\sigma \\
&= \mathbf{X}_1^T(\mathbf{E}_2 + \varepsilon(\mathbf{Y}_2)) - \mathbf{E}_1^T\mathbf{X}_2 - \mathbf{E}_\sigma
\end{aligned}$$

We consider each pair of elements in matrix $\mathbf{\Sigma}_1, \mathbf{\Sigma}_2$ separately, then derive the overall error rate by *union bound.* Now, we only need to consider the case $l_A = l_B = 1$. In this case, $\mathbf{X}_i, \mathbf{E}_i, \mathbf{Y}_i, (i = 1, 2)$ are column vectors in $\mathbb{Z}_q^n$, and $\mathbf{E}_\sigma \in \mathbb{Z}_q$.

If $\mathbf{Y}_2$ is independent of $(\mathbf{X}_2, \mathbf{E}_2)$, then we can directly calculate the distribution of $\boldsymbol{\sigma}_1 - \boldsymbol{\sigma}_2$. But now $\mathbf{Y}_2$ depends on $(\mathbf{X}_2, \mathbf{E}_2)$. To overcome this difficulty, we show that $\mathbf{Y}_2$ is independent of $(\mathbf{X}_2, \mathbf{E}_2)$ under a condition of $\mathbf{X}_2$ that happens with very high probability.

Theorem 6.2. *For any positive integer q, n, and a column vector $\mathbf{s} \in \mathbb{Z}_q^n$, let $\phi_\mathbf{s}$ denote the map $\mathbb{Z}_q^n \to \mathbb{Z}_q : \phi_\mathbf{s}(\mathbf{x}) = \mathbf{x}^T\mathbf{s}$. If there exits a coordinate of $\mathbf{s}$ which is not* zero divisor *in ring $\mathbb{Z}_q$, then map $\phi_\mathbf{s}$ is surjective.*

Proof. Let us assume one coordinate of $\mathbf{s}$, say s, has no *zero divisor* in ring $\mathbb{Z}_q$. Then the $\mathbb{Z}_q \to \mathbb{Z}_q$ map between the two $\mathbb{Z}_q$-modules deduced by s: $x \mapsto sx$, is injective, and thus surjective. Hence, $\phi_\mathbf{s}$ is surjective. □ □

$$
\begin{array}{ccc}
\underline{\text{Initiator}} & & \underline{\text{Responder}} \\
\mathsf{seed} \leftarrow \{0,1\}^{\kappa} & & \\
\mathbf{A} = \mathsf{Gen}(\mathsf{seed}) \in \mathbb{Z}_q^{n\times n} & & \\
\mathbf{X}_1, \mathbf{E}_1 \leftarrow \chi^{n\times l_A} & & \\
\mathbf{Y}_1 = \lfloor (\mathbf{A}\mathbf{X}_1 + \mathbf{E}_1)/2^{t_1} \rceil & & \\
& \xrightarrow{\mathsf{seed},\, \mathbf{Y}_1 \in \mathbb{Z}_{\lceil q/2^{t_1} \rceil}^{n\times l_A}} & \\
& & \mathbf{K}_2 \leftarrow \mathbb{Z}_m^{l_A\times l_B} \\
& & \mathbf{A} = \mathsf{Gen}(\mathsf{seed}) \\
& & \mathbf{X}_2, \mathbf{E}_2 \leftarrow \chi^{n\times l_B} \\
& & \mathbf{Y}_2 = \lfloor (\mathbf{A}^T\mathbf{X}_2 + \mathbf{E}_2)/2^{t_2} \rceil \\
& & \mathbf{E}_\sigma \leftarrow \chi^{l_A\times l_B} \\
& & \mathbf{\Sigma}_2 = 2^{t_1}\mathbf{Y}_1^T\mathbf{X}_2 + \mathbf{E}_\sigma \\
& & \mathbf{V} \leftarrow \mathsf{Con}(\mathbf{\Sigma}_2, \mathbf{K}_2, \mathsf{params}) \\
& \xleftarrow{\mathbf{Y}_2 \in \mathbb{Z}_{\lceil q/2^{t_2} \rceil}^{n\times l_B},\, \mathbf{V} \in \mathbb{Z}_g^{l_A\times l_B}} & \\
\mathbf{\Sigma}_1 = \mathbf{X}_1^T(2^{t_2}\mathbf{Y}_2) & & \\
\mathbf{K}_1 \leftarrow \mathsf{Rec}(\mathbf{\Sigma}_1, \mathbf{V}, \mathsf{params}) & &
\end{array}
$$

Figure 8: LWE-based key exchange from AKC, where $\mathbf{K}_1, \mathbf{K}_2 \in \mathbb{Z}_m^{l_A\times l_B}$ and $|\mathbf{K}_1| = |\mathbf{K}_2| = l_A l_B |m|$. $\mathbf{1}$ refers to the matrix which every elements are 1.

For a column vector $\mathbf{s}$ composed by random variables, denote by $F(\mathbf{s})$ the event that $\phi_{\mathbf{s}}$ is surjective. The following theorem gives a lower bound of probability of $F(\mathbf{s})$, where $\mathbf{s} \leftarrow \chi^n$. In our application, this lower bound is very close to 1.

Theorem 6.3. *Let p_0 be the probability that e is a* zero divisor *in ring $\mathbb{Z}_q$, where e is subject to χ. Then* $\Pr[\mathbf{s} \leftarrow \chi^n : F(\mathbf{s})] \geq 1 - p_0^n$

Proof. From Theorem 6.2, if $\phi_{\mathbf{s}}$ is not surjective, then all coordinates of $\mathbf{s}$ are *zero divisors*. Then $\Pr[\mathbf{s} \leftarrow \chi^n : \neg F(\mathbf{s})] \leq p_0^n$, and the proof is finished. □ □

Theorem 6.4. *If $\mathbf{s}, \mathbf{e} \leftarrow \chi^n, \mathbf{A} \leftarrow \mathbb{Z}_q^{n\times n}, \mathbf{y} = \mathbf{A}\mathbf{s} + \mathbf{e} \in \mathbb{Z}_q^n$, then under the condition $F(\mathbf{s})$, $\mathbf{y}$ is independent of $(\mathbf{s}, \mathbf{e})$, and is uniformly distributed over $\mathbb{Z}_q^n$.*

Proof. For all $\tilde{\mathbf{y}}, \tilde{\mathbf{s}}, \tilde{\mathbf{e}}$, $\Pr[\mathbf{y} = \tilde{\mathbf{y}} \mid \mathbf{s} = \tilde{\mathbf{s}}, \mathbf{e} = \tilde{\mathbf{e}}, F(\mathbf{s})] = \Pr[\mathbf{A}\tilde{\mathbf{s}} = \tilde{\mathbf{y}} - \tilde{\mathbf{e}} \mid \mathbf{s} = \tilde{\mathbf{s}}, \mathbf{e} = \tilde{\mathbf{e}}, F(\mathbf{s})]$. Let $\mathbf{A} = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n)^T, \tilde{\mathbf{y}} - \tilde{\mathbf{e}} = (c_1, c_2, \ldots, c_n)^T$, where $\mathbf{a}_i \in \mathbb{Z}_q^n$, and $c_i \in \mathbb{Z}_q$, for every $1 \leq i \leq n$. Since $\phi_{\mathbf{s}}$ is surjective, the number of possible choices of $\mathbf{a}_i$, satisfying $\mathbf{a}_i^T \cdot \tilde{\mathbf{s}} = c_i$, is $|\mathrm{Ker}\phi_{\mathbf{s}}| = q^{n-1}$. Hence, $\Pr[\mathbf{A}\tilde{\mathbf{s}} = \tilde{\mathbf{y}} - \tilde{\mathbf{e}} \mid \mathbf{s} = \tilde{\mathbf{s}}, \mathbf{e} = \tilde{\mathbf{e}}, F(\mathbf{s})] = (q^{n-1})^n/q^{n^2} = 1/q^n$. Since the right-hand side is the constant $1/q^n$, the distribution of $\mathbf{y}$ is uniform over $\mathbb{Z}_q^n$, and is irrelevant of $(\mathbf{s}, \mathbf{e})$. □ □

We now begin to analyze the error rate of the scheme presented in Figure 7.

Denote by E the event $|\mathbf{X}_1^T(\mathbf{E}_2 + \varepsilon(\mathbf{Y}_2)) - \mathbf{E}_1^T\mathbf{X}_2 - \mathbf{E}_\sigma|_q > d$. Then $\Pr[E] = \Pr[E|F(\mathbf{S})]\Pr[F(\mathbf{S})] + \Pr[E|\neg F(\mathbf{S})]\Pr[\neg F(\mathbf{S})]$. From Theorem 6.4, we replace $\mathbf{Y}_2 = \mathbf{A}^T\mathbf{X}_2 + \mathbf{E}_2$ in the event $E|F(\mathbf{S})$ with uniformly distributed $\mathbf{Y}_2$. Then,

$$
\begin{aligned}
\Pr[E] &= \Pr[\mathbf{Y}_2 \leftarrow \mathbb{Z}_q^n : E|F(\mathbf{S})]\Pr[F(\mathbf{S})] + \Pr[E|\neg F(\mathbf{S})]\Pr[\neg F(\mathbf{S})] \\
&= \Pr[\mathbf{Y}_2 \leftarrow \mathbb{Z}_q^n : E|F(\mathbf{S})]\Pr[F(\mathbf{S})] + \Pr[\mathbf{Y}_2 \leftarrow \mathbb{Z}_q^n : E|\neg F(\mathbf{S})]\Pr[\neg F(\mathbf{S})]
\end{aligned}
$$

$$+ \Pr[E|\neg F(\mathbf{S})] \Pr[\neg F(\mathbf{S})] - \Pr[\mathbf{Y}_2 \leftarrow \mathbb{Z}_q^n : E|\neg F(\mathbf{S})] \Pr[\neg F(\mathbf{S})]$$
$$= \Pr[\mathbf{Y}_2 \leftarrow \mathbb{Z}_q^n : E] + \varepsilon$$

where $|\varepsilon| \leq \Pr[\neg F(\mathbf{S})]$. In our application, p_0 is far from 1, and n is very large, by Theorem 6.3, ε is very small, so we simply ignore ε. If $\mathbf{Y}_2$ is uniformly distributed, then $\varepsilon(\mathbf{Y}_2)$ is a centered uniform distribution. Then, the distribution of $\mathbf{X}_1^T(\mathbf{E}_2 + \varepsilon(\mathbf{Y}_2)) - \mathbf{E}_1^T\mathbf{X}_2 - \mathbf{E}_\sigma$ can be directly computed by programs.

6.1.1 Discrete Distributions

As noted in [ADPS16, BCD+16], sampling from rounded Gaussian distribution (i.e., sampling from a discrete Gaussian distribution to a high precision) constitutes one of major efficiency bottleneck. In this work, for LWE-based key exchange, we use the following two classes of discrete distributions, which are specified in Table 8 and Table 9 respectively, where "bits" refers to the number of bits required to sample the distribution and "var." means the standard variation of the Gaussian distribution approximated. We remark that the discrete distributions specified in Table 9 are just those specified and used in [BCD+16] for the LWE-based Frodo scheme.

dist.	bits	var.	probability of						order	divergence
			0	±1	±2	±3	±4	±5		
D_1	8	1.10	94	62	17	2			15.0	1.0015832
D_2	12	0.90	1646	992	216	17			75.0	1.0003146
D_3	12	1.66	1238	929	393	94	12	1	30.0	1.0002034
D_4	16	1.66	19794	14865	6292	1499	200	15	500.0	1.0000274
D_5	16	1.30	22218	15490	5242	858	67	2	500.0	1.0000337

Table 8: Discrete distributions proposed in this work, and their Rényi divergences.

dist.	bits	var.	probability of							order	divergence
			0	±1	±2	±3	±4	±5	±6		
$\bar{D}_1$	8	1.25	88	61	20	3				25.0	1.0021674
$\bar{D}_2$	12	1.00	1570	990	248	24	1			40.0	1.0001925
$\bar{D}_3$	12	1.75	1206	919	406	104	15	1		100.0	1.0003011
$\bar{D}_4$	16	1.75	19304	14700	6490	1659	245	21	1	500.0	1.0000146

Table 9: Discrete distributions for Frodo [BCD+16], and their Rényi divergences

6.2 Instantiations, and Comparisons with Frodo

The comparisons, between the instantiations of our LWE-based KE protocol and Frodo, are summarized in the following tables 10, 11 and 12. Note that, for presentation simplicity, we take $l_A = l_B = l$ for the sets of parameters under consideration. Also, for space limitation, we use OKCN to denote OKCN-LWE in these tables. For "OKCN simple" proposed in Algorithm 3, it achieves a tight parameter constraint, specifically, $2md < q$. In comparison, the parameter constraint achieved by Frodo is $4md < q$. As we shall see, such a difference is one source that allows us to achieve better trade-offs among error rate, security, (computational and bandwidth) efficiency, and consensus range. In particular, it allows us to use q that is one bit shorter than that used in Frodo. Beyond saving bandwidth, employing a one-bit shorter q

	q	n	l	m	g		d		dist.	error rates		bw. (kB)	$\|A\|$ (kB)	$\|K\|$
					OKCN	Frodo	OKCN	Frodo		OKCN	Frodo			
Challenge	2^{10}	334	8	2^1	2^9	2	255	127	D_1	$2^{-47.9}$	$2^{-14.9}$	6.75	139.45	64
Classical	2^{11}	554	8	2^2	2^9	2	255	127	D_2	$2^{-39.4}$	$2^{-11.5}$	12.26	422.01	128
Recommended	2^{14}	718	8	2^4	2^{10}	2	511	255	D_3	$2^{-37.9}$	$2^{-10.2}$	20.18	902.17	256
Paranoid	2^{14}	818	8	2^4	2^{10}	2	511	255	D_4	$2^{-32.6}$	$2^{-8.6}$	22.98	1170.97	256
Paranoid-512	2^{12}	700	16	2^2	2^{10}	2	511	255	$\bar{D}_4$	$2^{-33.6}$	$2^{-8.3}$	33.92	735.00	512

Table 10: Parameters proposed for OKCN-LWE when $t = 0$ (i.e., without cutting off least significant bits). "distr." refers to the discrete distributions proposed in Table 8 and Table 9. "bw." means bandwidth in kilo-bytes (kB). "$|\mathbf{A}|$" refers to the size of the matrix. $|\mathbf{K}| = l^2 \log m$ denotes the length of consensus bits.

	q	n	l	m	g		d		dist.	error rates		bw. (kB)		$\|A\|$ (kB)	$\|K\|$
					OKCN	Frodo	OKCN	Frodo		OKCN	Frodo	OKCN	Frodo		
Challenge	2^{11}	352	8	2^1	2^2	2	383	255	$\bar{D}_1$	$2^{-80.1}$	$2^{-41.8}$	7.76	7.75	170.37	64
Classical	2^{12}	592	8	2^2	2^2	2	383	255	$\bar{D}_2$	$2^{-70.3}$	$2^{-36.2}$	14.22	14.22	525.70	128
Recommended	2^{15}	752	8	2^4	2^3	2	895	511	$\bar{D}_3$	$2^{-105.9}$	$2^{-38.9}$	22.58	22.57	1060.32	256
Paranoid	2^{15}	864	8	2^4	2^3	2	895	511	$\bar{D}_4$	$2^{-91.9}$	$2^{-33.8}$	25.94	25.93	1399.68	256

Table 11: Parameters of Frodo, and comparison with OKCN-LWE when $t = 0$. Here, "distr." refers to the discrete distributions specified in Table 9. Note that, on the parameters of Frodo, OKCN-LWE achieves significantly lower error rates.

also much improves the computational efficiency (as the matrix $\mathbf{A}$ becomes shorter, and consequently the cost of generating $\mathbf{A}$ and the related matrix operations are more efficient), and can render stronger security levels simultaneously. Here, we briefly highlight one performance comparison: OKCN-T2 (resp., Frodo-recommended) has 18.58kB (resp., 22.57kB) bandwidth, 887.15kB (resp., 1060.32kB) matrix $\mathbf{A}$, at least 134-bit (resp., 130-bit) quantum security, and error rate 2^{-39} (resp., $2^{-38.9}$).

	q	n	l	m	g	t	d	dist.	err.	bw. (kB)	$\|A\|$ (kB)	$\|K\|$
OKCN-T2	2^{14}	712	8	2^4	2^8	2	509	D_5	$2^{-39.0}$	18.58	887.15	256
OKCN-T1	2^{14}	712	8	2^4	2^8	1	509	D_5	$2^{-52.3}$	19.29	887.15	256

Table 12: Parameters proposed for OKCN-LWE with t least significant bits chopped off.

The error probabilities for OKCN-LWE are derived by computing $\Pr\left[\left|\mathbf{\Sigma}_1[i,j] - \mathbf{\Sigma}_2[i,j]\right|_q > d\right]$, for any $1 \leq i \leq l_A$ and $1 \leq j \leq l_B$, and then applying the union bound. The concrete failure probabilities are gotten by running the code slightly adjusted, actually simplified, from the open source code of Frodo. The simplified code are available from Github `http://github.com/OKCN`.

The concrete security levels are calculated by running the same code of Frodo. For comparison, the security levels of Frodo are presented in Appendix F.

6.2.1 Benchmark

The work [SM16] introduces the Open Quantum Safe Project. liboqs is one part of this project. liboqs provides the interface for adding new key exchange schemes, benchmark, and an easy way to integrate to OpenSSL.

Scheme	Attack	Rounded Gaussian					Post-reduction		
		m'	b	C	Q	P	C	Q	P
Classical	Primal	477	444	138	126	100	**132**	120	95
	Dual	502	439	137	125	99	**131**	119	94
Recommended	Primal	664	500	155	141	112	146	**133**	105
	Dual	661	496	154	140	111	145	**132**	104
Paranoid	Primal	765	586	180	164	130	179	163	**130**
	Dual	743	582	179	163	129	178	162	**129**
Paranoid-512	Primal	643	587	180	164	131	180	164	**130**
	Dual	681	581	179	163	129	178	162	**129**
OKCN-T2	Primal	638	480	149	136	108	148	**135**	107
	Dual	640	476	148	135	107	147	**134**	106

Table 13: Security estimation of the parameters described in Table 10 and Table 12. “Rounded Gaussian” refers to the ideal case that noises and errors follow the rounded Gaussian distribution. “Post-reduction” refers to the case of using discrete distributions as specified in Table 8.

We fork the liboqs on Github and add our OKCN-LWR-Recommended and OKCN-LWE-Recommended. Most of the source codes are modified from Frodo-Recommended provided in liboqs.

	time(us)	stdev	cycle	stdev	bw. (B)
LWE Frodo recommended					
Alice 0	1443.915	10.990	3313704	25236	11280
Bob	1940.616	12.809	4453734	29439	11288
Alice 1	170.109	3.655	390331	8317	-
LWR OKCN recommended					
Alice 0	1161.154	11.839	2664789	27129	9968
Bob	1722.525	12.401	3953182	28400	8224
Alice 1	133.984	3.980	307404	9065	-
LWE OKCN recommended					
Alice 0	1335.453	13.460	3064789	30871	9968
Bob	1753.240	14.293	4023632	32851	8608
Alice 1	146.162	3.528	335380	8035	-

Table 14: Benchmark of liboqs integrated with OKCN-LWE-Recommended. “time(us)” refers to mean time that spent on each iteration. “cycle” refers to mean number of cpu cycles. “stdev” refers to population standard deviation of time or cpu cycles. “bw. (B)” refers to bandwidth, counted in bytes.

We run benchmark of liboqs on Ubuntu Linux 16.04, GCC 5.4.0, Intel Core i7-4712MQ 2.30GHz, with hyperthreading and TurboBoost disabled, and the CPU frequency fixed to 2.30GHz (by following the instructions on `http://bench.cr.yp.to/supercop.html`). The benchmark result (Table 14) shows that OKCN-LWR-Recommended and OKCN-LWE-Recommended are faster than Frodo, and use smaller bandwidth.

7 Hybrid Construction of Key Exchange from LWE and LWR

By composing a CPA-secure symmetric-key encryption scheme, the LWE-based key exchange protocols presented Section 6 can be used to construct public-key encryption (PKE) schemes, by treating $(\mathbf{A}, \mathbf{Y}_1)$ (resp., $\mathbf{X}_1$) as the static public key (resp., secret key). Moreover, AKC-based key-exchange protocol can be directly used as a CPA-secure PKE scheme. To further improve the efficiency of the resultant PKE scheme, the observation here is we can generate the ephemeral $\mathbf{Y}_2$ in the ciphertext with LWR samples. This results in the following hybrid construction of key exchange from LWE and LWR in the public-key setting. For applications to PKE, we focus on the AKC-based protocol construction. Denote by $(n_A, n_B, l_A, l_B, q, p, KC, \chi)$ the system parameters, where $p|q$, and we choose p and q to be power of 2. The AKC-based protocol from LWE and LWR is presented in Figure 9. To further reduce the size of $\mathbf{Y}_1$ public key, some least significant bits can also be cut off from $\mathbf{Y}_1$.

The hybrid construction of key exchange from LWE and LWR is similar to the underlying protocol in Lizard [CKLS16]. The Lizard PKE scheme uses our AKCN as the underlying reconciliation mechanism, while our protocol is a general structure that can be implemented with either KC or AKC. In order to improve efficiency, Lizard [CKLS16] is based on the variants, referred to as spLWE and spLWR, of LWE and LWR with sparse secret. We aim at providing parameter sets for long term security, and estimate the concrete security in a more conservative way than [CKLS16] from the defender's point of view.

$$
\begin{array}{ccc}
\underline{\text{Initiator}} & & \underline{\text{Responder}} \\
\mathsf{seed} \leftarrow \{0,1\}^{\kappa} & & \\
\mathbf{A} \leftarrow \mathbb{Z}_q^{n_B \times n_A} & & \\
\mathsf{sk} = \mathbf{X}_1 \leftarrow \chi^{n_A \times l_A} & & \\
\mathbf{E}_1 \leftarrow \chi^{n_B \times l_A} & & \\
\mathbf{Y}_1 = \mathbf{A}\mathbf{X}_1 + \mathbf{E}_1 \in \mathbb{Z}_q^{n_B \times l_A} & & \\
& \xrightarrow{\mathsf{pk} = (\mathbf{A}, \mathbf{Y}_1)} & \\
& & \mathbf{K}_2 \leftarrow \mathbb{Z}_m^{l_A \times l_B} \\
& & \mathbf{A} = \mathsf{Gen}(\mathsf{seed}) \\
& & \mathbf{X}_2 \leftarrow \chi^{n_B \times l_B} \\
& & \mathbf{Y}_2 = \lfloor \mathbf{A}^T \mathbf{X}_2 \rceil_p \\
& & \mathbf{\Sigma}_2 = \lfloor \mathbf{Y}_1^T \mathbf{X}_2 \rceil_p \\
& & \mathbf{V} \leftarrow \mathsf{Con}(\mathbf{\Sigma}_2, \mathbf{K}_2, \mathsf{params}) \\
& \xleftarrow{\mathbf{Y}_2 \in \mathbb{Z}_p^{n_A \times l_B}, \mathbf{V} \in \mathbb{Z}_g^{l_A \times l_B}} & \\
\mathbf{\Sigma}_2 = \mathbf{X}_1^T \mathbf{Y}_2 \mod p & & \\
\mathbf{K}_1 \leftarrow \mathsf{Rec}(\mathbf{\Sigma}_1, \mathbf{V}, \mathsf{params}) & &
\end{array}
$$

Figure 9: AKC-based key exchange from LWE and LWR in the public-key setting, where $\mathsf{pk} = (\mathbf{A}, \mathbf{Y}_1)$ is fixed once and for all, $\mathbf{K}_1, \mathbf{K}_2 \in \mathbb{Z}_m^{l_A \times l_B}$ and $|\mathbf{K}_1| = |\mathbf{K}_2| = l_A l_B |m|$.

7.1 Security and Error Rate Analysis

The security proof is very similar to LWE-based and LWR-based key exchanges in previous sections, and is omitted here.

For the error probability, we have

$$\mathbf{\Sigma}_1 = \mathbf{X}_1^T\mathbf{Y}_2 = \frac{p}{q}\mathbf{X}_1^T\left(\mathbf{A}^T\mathbf{X}_2 - \{\mathbf{A}^T\mathbf{X}_2\}_p\right) = \frac{p}{q}\left(\mathbf{X}_1^T\mathbf{A}^T\mathbf{X}_2 - \mathbf{X}_1^T\{\mathbf{A}^T\mathbf{X}_2\}_p\right)$$

$$\mathbf{\Sigma}_2 = \left\lfloor \mathbf{Y}_1^T\mathbf{X}_2\right\rceil_p = \frac{p}{q}\left(\mathbf{Y}_1^T\mathbf{X}_2 - \{\mathbf{Y}_1^T\mathbf{X}_2\}_p\right) = \frac{p}{q}(\mathbf{X}_1^T\mathbf{A}^T\mathbf{X}_2 + \mathbf{E}_1^T\mathbf{X}_2 - \{\mathbf{Y}_1^T\mathbf{X}_2\}_p)$$

$$\mathbf{\Sigma}_2 - \mathbf{\Sigma}_1 = \frac{p}{q}\left(\mathbf{E}_1^T\mathbf{X}_2 + \mathbf{X}_1^T\{\mathbf{A}^T\mathbf{X}_2\}_p - \{\mathbf{E}_1^T\mathbf{X}_2 + \mathbf{X}_1^T\mathbf{A}^T\mathbf{X}_2\}_p\right) = \lfloor \mathbf{E}_1^T\mathbf{X}_2 + \mathbf{X}_1^T\{\mathbf{A}^T\mathbf{X}_2\}_p\rceil_p$$

We can see that the distribution of $\mathbf{\Sigma}_2 - \mathbf{\Sigma}_1$ can be derived from the distribution of $\mathbf{E}_1\mathbf{X}_2 + \mathbf{X}_1^T\{\mathbf{A}^T\mathbf{X}_2\}_p$. From Theorem 6.4, we know that for almost all (with *overwhelm* probability) given $\mathbf{X}_2$, the distribution of $\{\mathbf{A}^T\mathbf{X}_2\}_p$ is the uniform distribution over $[-q/2p, q/2p)^{n_A}$. The concrete error probability can then be derived numerically by computer programs. The codes and scripts are available on Github `http://github.com/OKCN`.

7.2 Parameter Selection

For simplicity, we use the Gaussian distribution of the same variance (denote as σ_s^2) for the noise $\mathbf{E}_1$, secrets $\mathbf{X}_1$ and $\mathbf{X}_2$. We consider the weighted dual attack and weighted primal attack in Section 5.3.

	σ_s^2	n_A	n_B	q	p	l	m	g	pk	cipher	err.	$\lvert\mathbf{K}\rvert$
Recommended	2.0	712	704	2^{15}	2^{12}	8	2^4	2^8	10.56	8.61	2^{-63}	256
Paranoid	2.0	864	832	2^{15}	2^{12}	8	2^4	2^8	12.24	10.43	2^{-52}	256

Table 15: Parameters for the hybrid construction of key exchange from LWE and LWR. "err." refers to the overall error probability. "$|\mathbf{K}|$" refers to the length of consensus bits. "pk" refers to the kilo-byte (kB) size of the public key $pk = (\mathbf{A}, \mathbf{Y}_1)$. "cipher" refers to the kB size of $(\mathbf{Y}_2, \mathbf{V})$.

Scheme	Attack	LWE					LWR				
		m'	b	C	Q	P	m'	b	C	Q	P
Recommended	Primal	699	464	144	**131**	105	664	487	151	**138**	109
	Dual	672	461	143	**131**	104	665	483	150	**137**	109
Paranoid	Primal	808	590	181	165	**131**	856	585	180	164	**130**
	Dual	789	583	179	163	**130**	765	579	178	162	**129**

Table 16: Security estimation of the parameters described in Table 15.

8 RLWE-Based Key Exchange from KC and AKC

Denote by $(\lambda, n, q, \sigma, KC)$ the system parameters, where λ is the security parameter, $q \geq 2$ is a positive prime number, σ parameterizes the discrete Gaussian distribution $D_{\mathbb{Z}^n,\sigma}$, n denotes the degree of polynomials in $\mathcal{R}_q$, and Gen a PRG generating $\mathbf{a} \in \mathcal{R}_q$ from a small seed. Let $KC = (\mathsf{params}, \mathsf{Con}, \mathsf{Rec})$ be a correct and secure KC scheme, where $\mathsf{params} = (q, g, m, d)$. In this section, we mainly consider $m = 2$. The KC-based key exchange protocol from RLWE is depicted in Figure 10, where the actual session-key is derived from $\mathbf{k}_1$ and $\mathbf{k}_2$ via some key derivation function KDF. As discussed in Section 5, a KC-based key exchange protocol can be trivially extended to work on any correct and secure AKC scheme, which is also presented in Figure 11, where $\mathbf{k}_2 \leftarrow \{0,1\}^n$ for KEM (rep., $\mathbf{k}_2 \in \{0,1\}^n$ corresponds to any plaintext for PKE). When used for PKE, $(\mathsf{seed}, \mathbf{y}_1)$ corresponds to the public key, and $\mathbf{x}_1$ corresponds to the secret key. In the protocol description, for presentation simplicity, the Con and Rec functions are applied to polynomials, meaning they are applied to each of the coefficients respectively. Also, for simplicity and symmetry, in the following analysis we assume the same number of tail bits are chopped off from both $\mathbf{y}_1$ and $\mathbf{y}_2$ by setting $t = t_1 = t_2 \geq 0$. In general, if we want to optimize the size of public key (resp., ciphertext), we can set $t_1 > t_2$ (resp., $t_1 < t_2$).

On parameters and implementations. The protocol described in Figure 10 works on any hard instantiation of the RLWE problem. But if n is power of 2, and prime q satisfies $q \bmod 2n = 1$, then number-theoretic transform (NTT) can be used to speed up polynomial multiplication. The performance can be further improved by using the Montgomery arithmetic and AVX2 instruction set [ADPS16], and by carefully optimizing performance-critical routines (in particular, NTT) in ARM assembly [AJS16,H14]. As in [ADPS16], the underlying noise distribution is the centered binomial distribution Ψ_η (rather than rounded Gaussian distribution with the standard deviation $\sigma = \sqrt{\eta/2}$), which is the sum of η independent centered binomial variables and can be rather trivially sampled in hardware and software with much better protection against timing attacks. We remark that the actual noise distribution is the composition of Ψ_η and the chopped bits determined by t. When estimating the post-quantum security levels, we usually just assume $t = 0$ (i.e., without considering the effect of t on the actual noise distribution); but sometimes we also take this value into account by approximately treating the standard deviation of the noise as $\sigma' = \sqrt{(2\sigma^2 + 2^{t-1})/2}$. This is based on the observation that no attacks known take advantage of the information of different noise distributions. The concrete values of post-quantum security are gotten by running the scripts provided by [ADPS16,BDK+17]. The parameters and performance of OKCN-RLWE and AKCN-RLWE are summarized in Table 17 and 18.

On security analysis. The security definition and proof of the RLWE-based key exchange protocol can be straightforwardly adapted from those for the KE protocol based on LWE or LWR. Similar analysis is also given in [BDK+17]. NewHope achieves 255-bit post-quantum security against the underlying lattice problem, but the actual use of its 256-bit shared key may provide essentially lower security guarantee (in view of the quadratic speedup by Grover's search algorithm and the possibility of more sophisticated quantum attacks against symmetric-key cryptography [KM10,KLL15]). In this sense, the 255-bit post-quantum security of NewHope is actually overshot in reality. For RLWE-based KE protocols, we aim for about 256-bit post-quantum security against both the underlying lattice problem and the shared key. This means that the shared key should be of at least 256 bits.

On error rate analysis. The error rate analysis is a special case of that for MLWE-based key exchange presented in Section 9. Note that the correctness of OKCN (resp., AKCN) requires that $(2d+1)m < q(1-\frac{1}{g})$ (resp., $(2d+1)m < q(1-\frac{1}{g})$); This means that on the same parameters (q, m, d), OKCN-RLWE with parameter g has the same error rate of AKCN-RLWE with parameter $g' = mg$. In this work, we set $m = 2$, and the concrete error rate values are gotten by running the scripts provided in [ADPS16, BDK+17].

8.1 Combining AKCN with Lattice Code in $\tilde{D}_4$

When implemented with the same parameters proposed in [ADPS16] for NewHope, as shown in Table 17, OKCN-RLWE and AKCN-RLWE reach 1024 consensus bits, with a failure probability around 2^{-40}; Though it suffices, we suggest, for most applications of key exchange. In order for

Initiator

$\mathsf{seed} \leftarrow \{0,1\}^\kappa$

$\mathbf{a} = \mathsf{Gen}(\mathsf{seed}) \in \mathcal{R}_q$

$\mathbf{x}_1, \mathbf{e}_1 \leftarrow D_{\mathbb{Z}^n,\sigma}$

$\mathbf{y}_1 = \lfloor (\mathbf{a} \cdot \mathbf{x}_1 + \mathbf{e}_1)/2^{t_1} \rceil$

Responder

$$\xrightarrow{\mathsf{seed}, \mathbf{y}_1 \in \mathcal{R}_q}$$

$\mathbf{a} = \mathsf{Gen}(\mathsf{seed})$

$\mathbf{x}_2, \mathbf{e}_2 \leftarrow D_{\mathbb{Z}^n,\sigma}$

$\mathbf{y}_2 = \lfloor (\mathbf{a} \cdot \mathbf{x}_2 + \mathbf{e}_2)/2^{t_2} \rceil$

$\mathbf{e}_2' \leftarrow D_{\mathbb{Z}^n,\sigma}$

$\boldsymbol{\sigma}_2 = 2^{t_1}\mathbf{y}_1 \cdot \mathbf{x}_2 + \mathbf{e}_2' \in \mathcal{R}_q$

$(\mathbf{k}_2, \mathbf{v}) \leftarrow \mathsf{Con}(\boldsymbol{\sigma}_2, \mathsf{params})$

$$\xleftarrow{\mathbf{y}_2 \in \mathcal{R}_q, \mathbf{v} \in \mathcal{R}_g}$$

$\boldsymbol{\sigma}_1 = 2^{t_2}\mathbf{y}_2 \cdot \mathbf{x}_1 \in \mathcal{R}_q$

$\mathbf{k}_1 \leftarrow \mathsf{Rec}(\boldsymbol{\sigma}_1, \mathbf{v}, \mathsf{params})$

Figure 10: RLWE-based key exchange from KC, where $\mathbf{k}_1, \mathbf{k}_2 \in \mathcal{R}_q$. The protocol instantiated with OKCN specified in Algorithm 1 is referred to as OKCN-RLWE.

reaching a negligible error rate, particularly for achieving a CCA-secure PKE scheme, we need to further lower the error rate.

A straightforward approach to reducing the error rate is to use the technique of NewHope by encoding and decoding the four-dimensional lattice $\tilde{D}_4$.[9] With such an approach, the error rate can be lowered to about 2^{-61}, but the shared-key size is reduced from 1024 to 256. AKCN-RLWE equipped with this approach, referred to as AKCN-4:1, is presented and analyzed in Appendix H. We note that, in comparison with NewHope-simple proposed in the subsequent work [ADPS16b], AKCN-4:1 still has some performance advantage in bandwidth expansion; specifically expanding 256 bits by AKCN-4:1 vs. 1024 bits by NewHope-simple compared to that of NewHope.[10]

8.2 On the Independence of Errors in Different Positions

Another approach to reduce error rate is to employ error correction code (ECC). Unfortunately, in general, the ECC-based approach can be more inefficient and overburdened than NewHope's approach. In this work, we make a key observation on RLWE-based key exchange, by proving that the errors in different positions in the shared-key are independent when n is large. Based upon this observation, we present a super simple and fast code, referred to as *single-error correction* (SEC) code, to correct at least one bit error. By equipping OKCN/AKCN with the SEC code, we present the (up-to-date) simplest RLWE-based key exchange from both OKCN and AKCN, which can be used for CCA-secure public-key encryption (e.g., for achieving 765-bit shared-key with bandwidth 3392 bytes and error rate $2^{-73.2}$ at about 250-bit post-quantum security).

Suppose $f(x), g(x)$ are two polynomials of degree n, whose coefficients are drawn independently from Gaussian. Let $h(x) = f(x) \cdot g(x) \in \mathbb{R}[x]/(x^n + 1)$. We show that for every two

[9]Decoding the 24-dimensional Leech lattice is also recently considered in [Pop16], but is more complicated.

[10]The bandwidth expansion, for both AKCN-4:1 and NewHope-simple, can be further compressed but at the price of losing operation simplicity.

	g	d	$\lvert\mathbf{K}\rvert$	bw.(B)	per.	n_H	err.	pq-sec
OKCN-RLWE	2^4	2879	1024	4128	2^{-48}	-	2^{-38}	255
OKCN-RLWE	2^6	3023	1024	4384	2^{-52}	-	2^{-42}	255
AKCN-RLWE	2^4	2687	1024	4128	2^{-42}	-	2^{-32}	255
AKCN-RLWE	2^6	2975	1024	4384	2^{-51}	-	2^{-41}	255
OKCN-SEC	2^2	2303	765	3904	2^{-31}	4	$2^{-48.5}$	255
OKCN-SEC	2^3	2687	765	4032	2^{-42}	4	$2^{-70.5}$	255
OKCN-SEC	2^3	2687	837	4021	2^{-42}	5	$2^{-69.5}$	255
AKCN-SEC	2^4	2687	765	4128	2^{-42}	4	$2^{-70.5}$	255
AKCN-SEC	2^4	2687	837	4128	2^{-42}	5	$2^{-69.5}$	255
NewHope	2^2	-	256	3872	2^{-69}	-	2^{-61}	255
NewHope-Simple	2^2	-	256	4000	2^{-69}	-	2^{-61}	255
AKCN-4:1-RLWE	2^2	-	256	3904	2^{-69}	-	2^{-61}	255

Table 17: All schemes in this table use the same parameters proposed for NewHope [ADPS16]: $(q = 12289, n = 1024, m = 2^1, t = 0, \sigma = \sqrt{8}, \kappa = 256, \Psi_{16})$. $\lvert\mathbf{K}\rvert$ refers to the total binary length of consensus bits. bw. (B) refers to the bandwidth in bytes. err. refers to failure probability. "n_H" refers to the dimension of SEC code used. "per" refers to the per bit error rate before applying the SEC code. "err." refers to overall error rate. "pq-sec" refers to the best known post-quantum attacks targeting the underlying lattice problem.

	g	t	σ (σ')	$\lvert\mathbf{K}\rvert$(SEC)	bw.(pk,$cipher$)	err.(SEC)	pq-sec (t-sec)
OKCN-RLWE	2^4	2	$\sqrt{8}$ ($\sqrt{9}$)	1024(765)	3392 (1440,1952)	$2^{-28.1}$ (2^{-61})	255 (258)
$\sigma = \sqrt{8}$	2^3	2	$\sqrt{8}$ ($\sqrt{9}$)	1024(765)	3264 (1440,1824)	$2^{-24.8}$ ($2^{-54.4}$)	255 (258)
	2^3	1	$\sqrt{8}$ ($\sqrt{8.5}$)	1024(765)	3520 (1568,1952)	$2^{-33.4}$ ($2^{-71.6}$)	255 (257)
	2^4	1	$\sqrt{8}$ ($\sqrt{8.5}$)	1024(765)	3648 (1568,2080)	$2^{-37.8}$ ($2^{-80.4}$)	255 (257)
OKCN-RLWE	2^2	2	$\sqrt{6}$ ($\sqrt{7}$)	1024(765)	3136(1440,1696)	$2^{-31.8}$ ($2^{-68.4}$)	246 (250)
$\sigma = \sqrt{6}$	2^3	2	$\sqrt{6}$ ($\sqrt{7}$)	1024(765)	3264 (1440,1824)	$2^{-43.2}$ ($2^{-91.2}$)	246 (250)
	2^4	2	$\sqrt{6}$ ($\sqrt{7}$)	1024(765)	3392(1440,1952)	2^{-49} ($2^{-102.8}$)	246 (250)
	2^3	1	$\sqrt{6}$ ($\sqrt{6.5}$)	1024(765)	3520 (1568,1952)	$2^{-60.6}$ (2^{-126})	246 (248)
	2^4	1	$\sqrt{6}$ ($\sqrt{6.5}$)	1024(765)	3648 (1568,2080)	$2^{-68.9}$ ($2^{-142.6}$)	246 (248)
AKCN-RLWE	2^5	2	$\sqrt{8}$ ($\sqrt{9}$)	1024(765)	3520 (1440,2080)	$2^{-28.1}$ (2^{-61})	255 (258)
$\sigma = \sqrt{8}$	2^4	2	$\sqrt{8}$ ($\sqrt{9}$)	1024(765)	3392 (1440,1952)	$2^{-24.8}$ ($2^{-54.4}$)	255 (258)
	2^4	1	$\sqrt{8}$ ($\sqrt{8.5}$)	1024(765)	3648 (1568,2080)	$2^{-33.4}$ ($2^{-71.6}$)	255 (257)
	2^5	1	$\sqrt{8}$ ($\sqrt{8.5}$)	1024(765)	3776 (1568,2208)	$2^{-37.8}$ ($2^{-80.4}$)	255 (257)
AKCN-RLWE	2^3	2	$\sqrt{6}$ ($\sqrt{7}$)	1024(765)	3264(1440,1824)	$2^{-31.8}$ ($2^{-68.4}$)	246 (250)
$\sigma = \sqrt{6}$	2^4	2	$\sqrt{6}$ ($\sqrt{7}$)	1024(765)	3392(1440,1952)	$2^{-43.2}$ ($2^{-91.2}$)	246 (250)
	2^5	2	$\sqrt{6}$ ($\sqrt{7}$)	1024(765)	3520(1440,2080)	2^{-49} ($2^{-102.8}$)	246 (250)
	2^4	1	$\sqrt{6}$ ($\sqrt{6.5}$)	1024(765)	3648 (1568,2080)	$2^{-60.6}$ (2^{-126})	246 (248)
	2^5	1	$\sqrt{6}$ ($\sqrt{6.5}$)	1024(765)	3776 (1568,2208)	$2^{-68.9}$ ($2^{-142.6}$)	246 (248)

Table 18: Parameters for $\kappa = 256$, $q = 12289$, $n = 1024$, $m = 2$, $n_H = 4$. "$\lvert K\rvert$ (SEC)" refers to the key size (resp., key size with SEC); "bw.(pk,cipher)" refers to the bandwidth in bytes (including the size of $pk = (\mathbf{y}_1, \mathsf{seed})$ and $cipher = (\mathbf{y}_2, \mathbf{v})$); "err.(SEC)" refers to the error rate (resp., the error rate with SEC); "pq-sec" (resp., "t-sec") refers to the security against the best known quantum attacks against the underlying lattice problem without considering the effect of t (resp., by heuristically viewing the standard deviation of the noise as $\sigma' = \sqrt{(2\sigma^2 + 2^{t-1})/2}$).

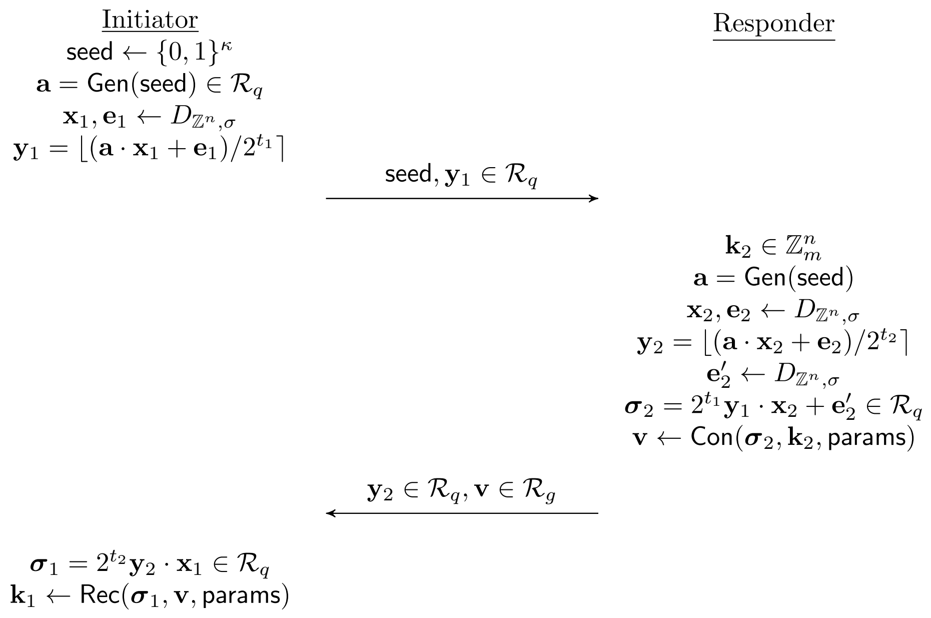
Initiator
seed $\leftarrow \{0,1\}^\kappa$
$\mathbf{a} = \mathsf{Gen}(\mathsf{seed}) \in \mathcal{R}_q$
$\mathbf{x}_1, \mathbf{e}_1 \leftarrow D_{\mathbb{Z}^n,\sigma}$
$\mathbf{y}_1 = \lfloor (\mathbf{a} \cdot \mathbf{x}_1 + \mathbf{e}_1)/2^{t_1} \rceil$
seed, $\mathbf{y}_1 \in \mathcal{R}_q$
Responder
$\mathbf{k}_2 \in \mathbb{Z}_m^n$
$\mathbf{a} = \mathsf{Gen}(\mathsf{seed})$
$\mathbf{x}_2, \mathbf{e}_2 \leftarrow D_{\mathbb{Z}^n,\sigma}$
$\mathbf{y}_2 = \lfloor (\mathbf{a} \cdot \mathbf{x}_2 + \mathbf{e}_2)/2^{t_2} \rceil$
$\mathbf{e}_2' \leftarrow D_{\mathbb{Z}^n,\sigma}$
$\boldsymbol{\sigma}_2 = 2^{t_1}\mathbf{y}_1 \cdot \mathbf{x}_2 + \mathbf{e}_2' \in \mathcal{R}_q$
$\mathbf{v} \leftarrow \mathsf{Con}(\boldsymbol{\sigma}_2, \mathbf{k}_2, \mathsf{params})$
$\mathbf{y}_2 \in \mathcal{R}_q, \mathbf{v} \in \mathcal{R}_g$
$\boldsymbol{\sigma}_1 = 2^{t_2}\mathbf{y}_2 \cdot \mathbf{x}_1 \in \mathcal{R}_q$
$\mathbf{k}_1 \leftarrow \mathsf{Rec}(\boldsymbol{\sigma}_1, \mathbf{v}, \mathsf{params})$

Figure 11: RLWE-based key exchange from AKC, where $\mathbf{k}_1, \mathbf{k}_2 \in \mathcal{R}_q$. The protocol instantiated with AKCN in Algorithm 4 is referred to as AKCN-RLWE.

different integers $0 \leq c_1, c_2 < n$, the joint distribution of $(h[c_1], h[c_2])$ will approach to the two-dimensional Gaussian when n tends to infinity. Hence, for the basic construction of RLWE-based key exchange from KC and AKC presented in Figure 10, it is reasonable to assume that the error rates of any two different positions are independent when n is sufficiently large.

For representation simplicity, for any polynomial f, let $f[i]$ denote the coefficient of x^i.

Lemma 8.1. *Suppose $f(x), g(x) \in \mathbb{R}[x]/(x^n+1)$ are two n-degree polynomials whose coefficients are drawn independently from $\mathcal{N}(0, \sigma^2)$. Let $h(x) = f(x) \cdot g(x) \in \mathbb{R}[x]/(x^n + 1)$, where $h(x)$ is represented as an n-degree polynomial. For any two different integers $0 \leq c_1, c_2 < n$, the characteristic function of the two-dimensional random vector $(h[c_1], h[c_2]) \in \mathbb{R}^2$ is*

$$\phi_{c_1,c_2}(t_1, t_2) = \mathbb{E}\left[e^{i(t_1 h[c_1]+t_2 h[c_2])}\right] = t_1\mathbf{f}^T\mathbf{A}_{c_1}\mathbf{g} + t_2\mathbf{f}^T\mathbf{A}_{c_2}\mathbf{g} \tag{6}$$

$$= \prod_{k=0}^{n-1}\left(1 + \sigma^4\left(t_1^2 + t_2^2 + 2t_1t_2\cos\left(\pi(c_1 - c_2)\frac{2k+1}{n}\right)\right)\right)^{-\frac{1}{2}} \tag{7}$$

Proof. One can observe that $t_1 h[c_1] + t_2 h[c_2]$ is equal to

$$t_1\left(\sum_{i+j=c_1} f[i]g[j] - \sum_{i+j=c_1+n} f[i]g[j]\right) + t_2\left(\sum_{i+j=c_2} f[i]g[j] - \sum_{i+j=c_2+n} f[i]g[j]\right)$$
$$= t_1\mathbf{f}^T\mathbf{A}_{c_1}\mathbf{g} + t_2\mathbf{f}^T\mathbf{A}_{c_2}\mathbf{g}. = \mathbf{f}^T(t_1\mathbf{A}_{c_1} + t_2\mathbf{A}_{c_2})\mathbf{g}$$

Where $\mathbf{f} = (f[0], f[1], \dots, f[n-1])^T$, $\mathbf{g} = (g[0], g[1], \dots, g[n-1])^T$, and the notations $\mathbf{A}_{c_1}, \mathbf{A}_{c_2}$ are defined by

$$\mathbf{A}_c = \begin{pmatrix} & & 1 & & & \\ & \iddots & & & & \\ 1 & & & & & \\ & & & & & -1 \\ & & & & \iddots & \\ & & & -1 & & \end{pmatrix}$$

The value 1 in the first row is in the c-th column.

As $t_1\mathbf{A}_{c_1} + t_2\mathbf{A}_{c_2}$ is symmetric, it can be orthogonally diagonalize as $\mathbf{P}^T\mathbf{\Lambda}\mathbf{P}$, where $\mathbf{P}$ is orthogonal, and $\mathbf{\Lambda}$ is diagonal. Hence, $\phi_{c_1,c_2}(t_1,t_2) = \mathbb{E}[\exp(i(\mathbf{Pf})^T\mathbf{\Lambda}(\mathbf{Pg}))]$. Since $\mathbf{P}$ is orthogonal, it keeps the normal distribution unchanged. Hence, $(\mathbf{Pf})^T\mathbf{\Lambda}(\mathbf{Pg})$ equals to the sum of n scaled products of two independent one-dimensional Gaussian.

Suppose $\lambda_1, \lambda_2, \dots, \lambda_n$ are the eigenvalues of $t_1\mathbf{A}_{c_1} + t_2\mathbf{A}_{c_2}$, and ϕ is the characteristic function of the product of two independent one-dimensional standard Gaussian. Then we have

$$\phi_{c_1,c_2}(t_1,t_2) = \prod_{k=0}^{n-1} \phi(\sigma^2\lambda_k) \tag{8}$$

From [Sim02], $\phi(t) = (1+t^2)^{-1/2}$. For λ_k, we further observe that

$$\begin{aligned}(t_1\mathbf{A}_{c_1} + t_2\mathbf{A}_{c_2})^2 &= (t_1^2+t_2^2)\mathbf{I} + t_1t_2(\mathbf{A}_{c_1}\mathbf{A}_{c_2} + \mathbf{A}_{c_2}\mathbf{A}_{c_1}) \\ &= (t_1^2+t_2^2)\mathbf{I} + t_1t_2(\mathbf{G}^{c_2-c_1} + \mathbf{G}^{c_1-c_2}),\end{aligned}$$

where

$$\mathbf{G} = \begin{pmatrix} & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \\ -1 & & & & \end{pmatrix}$$

The characteristic polynomial of $\mathbf{G}$ is $x^n + 1$. Hence, λ_k satisfies

$$\lambda_k^2 = t_1^2 + t_2^2 + 2t_1t_2\cos\left(\pi(c_1-c_2)\frac{2k+1}{n}\right)$$

By taking this into Equation 8, we derive the Equation 7. □ □

Theorem 8.1. *For any fixed integers $0 \le c_1, c_2 < n$, $c_1 \neq c_2$, when n tends to infinity, the distribution of* $\left(\frac{h[c_1]}{\sigma^2\sqrt{n}}, \frac{h[c_2]}{\sigma^2\sqrt{n}}\right)$ converges (in distribution) *to the two-dimensional normal distribution* $\mathcal{N}(\mathbf{0}, \mathbf{I}_2)$.

Proof. Let $\phi(t_1,t_2)$ denote the characteristic function of the random vector $\left(\frac{h[c_1]}{\sigma^2\sqrt{n}}, \frac{h[c_2]}{\sigma^2\sqrt{n}}\right)$. Then, for fixed t_1, t_2,

$$\ln(\phi(t_1,t_2)) = -\frac{1}{2}\sum_{k=0}^{n-1}\ln\left(1 + \frac{1}{n}\left(t_1^2+t_2^2+2t_1t_2\cos\left(\pi(c_1-c_2)\frac{2k+1}{n}\right)\right)\right) \tag{9}$$

$$= -\frac{1}{2}\sum_{k=0}^{n-1}\left[\frac{1}{n}\left(t_1^2 + t_2^2 + 2t_1t_2\cos\left(\pi(c_1 - c_2)\frac{2k+1}{n}\right)\right) + r_k\right] \tag{10}$$

$$= -\frac{1}{2}\left(t_1^2 + t_2^2\right) - \frac{1}{2}\sum_{k=0}^{n-1} r_k, \tag{11}$$

where r_k is the Lagrange remainders. So, $|r_k| \leq \lambda_k^4/2n^2$. Since $\lambda_k^2 \leq (|t_1| + |t_2|)^2$, we have $|r_k| \leq (|t_1| + |t_2|)^4/2n^2$.

When n tends to infinity, $\phi(t_1, t_2)$ converges pointwise to $\exp(-(t_1^2 + t_2^2)/2)$, which is the characteristic function of the two-dimensional normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I}_2)$. From Lévy's convergence theorem, we derive that the random vector $\left(\frac{h[c_1]}{\sigma^2\sqrt{n}}, \frac{h[c_2]}{\sigma^2\sqrt{n}}\right)$ *converges in distribution* to the normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I}_2)$. $\square$ $\square$

8.3 Reducing Error Rate with Single-Error Correction Code

Note that, for the basic protocol construction of RLWE-based key exchange from KC and AKC presented in Figure 10, it has already achieved per-bit error rate of about 2^{-42}. The observation here is that, by Theorem 8.1 on the independence of error in different positions when n is large, if we can correct one bit error the error rate will be greatly lowered. Towards this goal, we present an variant of the Hamming code, referred to as *single-error correction* (SEC) code, which can correct one-bit error in a very simple and fast way.

8.3.1 Single-Error Correction Code

All the arithmetic operations in this section are over $\mathbb{Z}_2$. For a positive integer n_H, denote $N_H = 2^{n_H}$, and define the matrix $\mathbf{H}$ as following, where for any i, $1 \leq i \leq N_H - 1$, the i-th column of $\mathbf{H}$ just corresponds to the binary presentation of i.

$$\mathbf{H}_{n_H \times (N_H - 1)} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & \cdots & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & \cdots & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & \cdots & 1 & 1 & 1 & 1 \\ & & & & & & & \cdots & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 1 & 1 & 1 \end{pmatrix}$$

For arbitrary $\mathbf{x} = (x_1, \ldots, x_{N_H-1}) \in \mathbb{Z}_2^{N_H-1}$, let $\mathbf{p}^T = \mathbf{H}\mathbf{x}^T$. It is easy to check that the j-th element of $\mathbf{p}$ is the exclusive-or of all x_i's satisfying the j-th least significant bit of i is 1, where $1 \leq j \leq n_H$ and $1 \leq i \leq N_H - 1$. Specifically, the first element of $\mathbf{p}$ is the exclusive-or of all x_i that the least significant bit of i is 1, and the second element of $\mathbf{p}$ is the exclusive-or of all x_i that the second least significant bit of i is 1, and so on. Denote $\mathbf{p} = (p_1, p_2, \ldots, p_{n_H})$. We can combine the bits in $\mathbf{p}$ into a binary number $\overline{\mathbf{p}} = 2^0 p_1 + 2^1 p_2 + \ldots 2^{n_H-1} p_{n_H}$. The construction of $\mathbf{H}$ directly leads to the following proposition.

Proposition 8.1. *If $\mathbf{p}^T = \mathbf{H}\mathbf{x}^T$, and the Hamming weight of $\mathbf{x}$ is 1, then $\overline{\mathbf{p}}$ is the subscript index of the only 1 in $\mathbf{x}$.*

Algorithm 14 $\mathsf{Encode}_{\mathcal{C}}(\mathbf{x} = (x_1, \ldots, x_{N_H-1}))$

```
1: $x_0 = \oplus_{i=1}^{N_H-1} x_i$
2: $\mathbf{p}^T = \mathbf{H}\mathbf{x}^T$
3: $\mathbf{c} = (x_0, \mathbf{x}, \mathbf{p})$
4: return $\mathbf{c}$
```

Algorithm 15 $\mathsf{Decode}_{\mathcal{C}}(x_0, \mathbf{x}, \mathbf{p})$

```
1: $p = \oplus_{i=0}^{N_H-1} x_i$
2: if $p = 1$ then
3:     $i = \overline{\mathbf{H}\mathbf{x}^T \oplus \mathbf{p}}$        ▷ bitwise exclusive-or
4:     $x_i = x_i \oplus 1$
5: end if
6: return $\mathbf{x} = (x_1, \ldots, x_{N_H-1})$
```

The single-error correction code $\mathcal{C}$ is defined by

$$\mathcal{C} = \left\{ (x_0, \mathbf{x}, \mathbf{p}) \in \mathbb{Z}_2 \times \mathbb{Z}_2^{N_H-1} \times \mathbb{Z}_2^{n_H} \mid x_0 = \oplus_{i=1}^{N_H-1} x_i, \mathbf{p}^T = \mathbf{H}\mathbf{x}^T \right\}$$

The encoding algorithm is straightforward and depicted in Algorithm 14.

We now show that $\mathcal{C}$ can correct one bit error. Suppose $\mathbf{x}$ is encoded into $\mathbf{c} = (x_0, \mathbf{x}, \mathbf{p})$. For some reasons, such as the noise in communication channel, the message $\mathbf{c}$ may be changed into $\mathbf{c}' = (x_0', \mathbf{x}', \mathbf{p}')$. We only need to consider the case that at most one bit error occurs. If x_0' equals to the parity bit of $\mathbf{x}'$, then no error occurs in x_0 and $\mathbf{x}$. Otherwise, there is one bit error in x_0' or $\mathbf{x}'$, but $\mathbf{p}' = \mathbf{p}$ (as we assume there exists at most one bit error that has already occurred in x_0' or $\mathbf{x}'$). We calculate $\mathbf{p}'' = \mathbf{H}\mathbf{x}'^T \oplus \mathbf{p}'^T$. In fact, $\mathbf{p}'' = \mathbf{H}\mathbf{x}'^T \oplus \mathbf{p}^T = \mathbf{H}(\mathbf{x}'^T \oplus \mathbf{x}^T)$. If the one-bit error occurs in $\mathbf{x}'$, by Proposition 8.1, $\overline{\mathbf{p}''}$ is the subscript index of the error bit. If the one-bit error occurs on x_0', then $\mathbf{x}' = \mathbf{x}$, and $\mathbf{p}'' = \mathbf{H}\mathbf{0} = \mathbf{0}$. Hence, $\overline{\mathbf{p}''}$ always equals to the subscript index of the error bit.

The decoding algorithm is depicted in Algorithm 15. Note that, according to the special form of $\mathbf{H}$, the matrix multiplication $\mathbf{H}\mathbf{x}^T$ in both encoding and decoding can be done with simple bit operations like bit shifts and bitwise exclusive-or (such an implementation is given in Appendix I). Moreover, for AKCN-SEC and OKCN-SEC, the calculations in Lines 2-4 in Algorithm 15 are executed only with probability around 2^{-40}, so the decoding is extremely fast.

8.3.2 AKC and KC with SEC code

Figure 12 depicts the AKC scheme equipped with the SEC code. Note that $\mathsf{Encode}_{\mathcal{C}}$ can be calculated off-line.

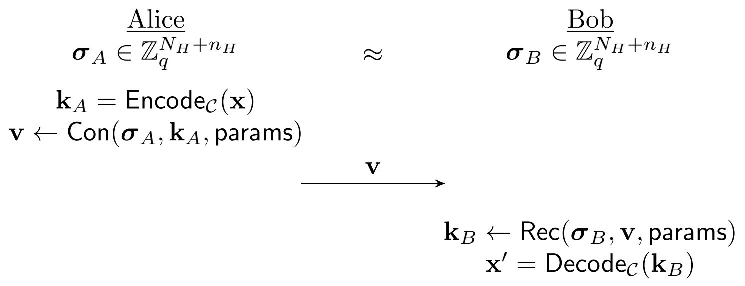
Alice
$\boldsymbol{\sigma}_A \in \mathbb{Z}_q^{N_H+n_H}$
$\approx$
Bob
$\boldsymbol{\sigma}_B \in \mathbb{Z}_q^{N_H+n_H}$
$\mathbf{k}_A = \mathsf{Encode}_{\mathcal{C}}(\mathbf{x})$
$\mathbf{v} \leftarrow \mathsf{Con}(\boldsymbol{\sigma}_A, \mathbf{k}_A, \mathsf{params})$
$\mathbf{v}$
$\mathbf{k}_B \leftarrow \mathsf{Rec}(\boldsymbol{\sigma}_B, \mathbf{v}, \mathsf{params})$
$\mathbf{x}' = \mathsf{Decode}_{\mathcal{C}}(\mathbf{k}_B)$

Figure 12: Depiction of AKC with SEC code, where $\mathbf{k}_A, \mathbf{k}_B \in \mathbb{Z}_2^{N_H+n_H}$, $|\mathbf{x}| = |\mathbf{x}'| = N_H - 1$. If the Hamming distance between $\mathbf{k}_A$ and $\mathbf{k}_B$ is at most 1, then $\mathbf{x} = \mathbf{x}'$.

$$\begin{array}{ccc}
\underline{\text{Alice}} & & \underline{\text{Bob}} \\
\boldsymbol{\sigma}_A \in \mathbb{Z}_q^{N_H+n_H} & \approx & \boldsymbol{\sigma}_B \in \mathbb{Z}_q^{N_H+n_H} \\
(\mathbf{k}_A, \mathbf{v}) \leftarrow \mathsf{Con}(\boldsymbol{\sigma}_A, \mathsf{params}) & & \\
\text{Denote } \mathbf{k}_A \text{ as } (x_0, \mathbf{x} = (x_1, \ldots, x_{N_H-1}), \mathbf{p}) & & \\
\mathbf{v}' = \mathsf{Encode}_{\mathcal{C}}(\mathbf{x}) \oplus \mathbf{k}_A & & \\
& \xrightarrow{\mathbf{v}, \mathbf{v}'} & \\
& & \mathbf{k}_B \leftarrow \mathsf{Rec}(\boldsymbol{\sigma}_B, \mathbf{v}, \mathsf{params}) \\
& & \mathbf{x}' = \mathsf{Decode}_{\mathcal{C}}(\mathbf{k}_B \oplus \mathbf{v}')
\end{array}$$

Figure 13: Depiction of application of SEC code to KC, where $\mathbf{k}_A, \mathbf{k}_B \in \mathbb{Z}_2^{N_H+n_H}$. If $\mathbf{k}_A$ and $\mathbf{k}_B$ have at most one different bit, then $\mathbf{x} = \mathbf{x}'$.

For KC equipped with the SEC code, we propose the algorithm depicted in Figure 13. Note that Alice only needs to send $n_H + 1$ bits of $\mathbf{v}'$, as the second to the N_H-th elements of $\mathbf{v}'$ are all zeros. Bob calculates $\mathbf{x}' = \mathsf{Decode}_{\mathcal{C}}(\mathbf{k}_B \oplus \mathbf{v}')$. In fact, $\mathbf{k}_B \oplus \mathbf{v}' = \mathsf{Encode}_{\mathcal{C}}(\mathbf{x}) \oplus (\mathbf{k}_A \oplus \mathbf{k}_B)$. Hence, if the Hamming distance between $\mathbf{k}_A$ and $\mathbf{k}_B$ is 1, then $\mathbf{x}' = \mathbf{x}$. To prove security of the algorithm in Figure 13, we need the following theorem.

Theorem 8.2. *Let $\mathcal{V} = \mathbb{Z}_2 \times \{\mathbf{0} \in \mathbb{Z}_2^{N_H-1}\} \times \mathbb{Z}_2^{n_H}$, then $\mathbb{Z}_2^{N_H+n_H} = \mathcal{C} \bigoplus \mathcal{V}$, where $\bigoplus$ denotes direct sum.*

Proof. For any $\mathbf{k}_A = (x_0, \mathbf{x} = (x_1, \ldots, x_{N_H-1}), \mathbf{p}) \in \mathbb{Z}_2^{N_H+n_H}$, let $\mathbf{c} = \mathsf{Encode}_{\mathcal{C}}(\mathbf{x})$ and $\mathbf{v}' = \mathsf{c} \oplus \mathbf{k}_A$. We have the decomposition $\mathbf{k}_A = \mathbf{c} \oplus \mathbf{v}'$, where $\mathbf{c} \in \mathcal{C}$ and $\mathbf{v}' \in \mathcal{V}$.

Next, we prove $\mathcal{V} \cap \mathcal{C} = \mathbf{0}$. If $\mathbf{k} = (x_0, \mathbf{x}, \mathbf{p}) \in \mathcal{V} \cap \mathcal{C}$, then $\mathbf{x} = 0$, which implies $x_0 = 0$ and $\mathbf{p}^T = \mathbf{H0} = \mathbf{0}$. Hence, $\mathbf{k} = \mathbf{0}$. □ □

When $\mathbf{k}_A$ is subjected to uniform distribution, then by Theorem 8.2, after the decomposition of $\mathbf{k}_A = \mathbf{c} \oplus \mathbf{v}'$ where $\mathbf{c} \in \mathcal{C}$ and $\mathbf{v}' \in \mathcal{V}$, $\mathbf{c}$ and $\mathbf{v}'$ are subjected to uniform distribution in $\mathcal{C}$ and $\mathcal{V}$ respectively. And $\mathbf{c}$ and $\mathbf{v}'$ are independent. As both $\mathbb{Z}_2^{N_H-1} \to \mathcal{C}$ and $\mathbf{x} \mapsto \mathsf{Encode}_{\mathcal{C}}(\mathbf{x})$ are one-to-one correspondence, we derive that $\mathbf{x}$ and $\mathbf{v}'$ are independent, and $\mathbf{x}$ is uniformly distributed. The parameters and performances for s OKCN-SEC and AKCN-SEC are summarized in Table 17 and 18.

8.3.3 KEM Specification of AKCN-SEC in the Public-Key Settiing

We divide the n-bit string $\mathbf{k}_2$ into $\nu = \lfloor n/(N_H + n_H) \rfloor$ blocks, then apply our SEC code in each block. This means the actual shared-key is of size $\nu \cdot (N_H - 1)$ bits. Note that this approach can also correct more than one bit errors, if at most one bit error occurs in each block.

Suppose the per bit error rate of $\mathbf{k}_1$ and $\mathbf{k}_2$ is p, then under the assumption that the errors in different positions are independent, we can estimate that the overall heuristic error rate for the actual shared-key is no larger than $\lfloor \frac{n}{N_H+n_H} \rfloor C_{N_H+n_H}^2 p^2$.

Algorithm 16 $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}()$

```
1: $\mathsf{seed} \leftarrow \{0,1\}^\kappa$
2: $\mathbf{a} := \mathsf{Gen}(\mathsf{seed})$
3: $\mathbf{x}_1, \mathbf{e}_1 \leftarrow D_{\mathbb{Z}^n,\sigma}$
4: $\mathbf{y}_1 := \lfloor (\mathbf{a}\mathbf{x}_1 + \mathbf{e}_1)/2^{t_1} \rceil$
5: return $(\mathsf{pk} := (\mathsf{seed}, \mathbf{y}_1), \mathsf{sk} := \mathbf{x}_1)$
```

Algorithm 17 $(\mathsf{ct}, \mathsf{key}) \leftarrow \mathsf{Encaps}(\mathsf{pk})$

```
1: $\mathbf{x}_2, \mathbf{e}_2, \mathbf{e}'_2 \leftarrow D_{\mathbb{Z}^n,\sigma}$
2: $\mathbf{a} := \mathsf{Gen}(\mathsf{seed})$
3: $\mathbf{y}_2 := \lfloor (\mathbf{a}\mathbf{x}_2 + \mathbf{e}_2)/2^{t_2} \rceil$
4: $\sigma_2 := 2^{t_1}\mathbf{y}_1 \cdot \mathbf{x}_2 + \mathbf{e}'_2$
5: $\mathbf{k}'_2 \leftarrow \mathbb{Z}_2^{(N_H-1)\cdot\lfloor n/(N_H+n_H)\rfloor}$
6: $\mathbf{k}_2 := \mathsf{Encode}_{\mathcal{C}}(\mathbf{k}'_2)$
7: $\mathbf{v} \leftarrow \mathsf{Con}(\sigma_2, \mathbf{k}_2, \mathsf{params})$
8: return $(\mathsf{ct} := (\mathbf{y}_2, \mathbf{v}), \mathsf{key} := \mathbf{k}'_2)$
```

Algorithm 18 $\mathsf{key}' \leftarrow \mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$

```
1: $\sigma_1 := 2^{t_2}\mathbf{y}_2 \cdot \mathbf{x}_1$
2: $\mathbf{k}_1 := \mathsf{Rec}(\sigma_1, \mathbf{v}, \mathsf{params})$
3: $\mathbf{k}'_1 := \mathsf{Decode}_{\mathcal{C}}(\mathbf{k}_1)$
4: return $\mathsf{key}' := \mathbf{k}'_1$
```

Note: the above pseudo-codes describes how three algorithms work in general. Notice that $n \cdot (N_H - 1)/(N_H + n_H)$ may not be a positive integer in practice; in particular, it is *not* a positive integer in our software implementation. In this case, some coefficients in $\mathbf{v}, \sigma_1, \sigma_2$ will not contribute to the generation of the shared secret key.

8.3.4 KEM Specification of OKCN-SEC in the Public-Key Setting

Algorithm 19 $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}()$

```
1: $\mathsf{seed} \leftarrow \{0,1\}^\kappa$
2: $\mathbf{a} := \mathsf{Gen}(\mathsf{seed})$
3: $\mathbf{x}_1, \mathbf{e}_1 \leftarrow D_{\mathbb{Z}^n,\sigma}$
4: $\mathbf{y}_1 := \lfloor (\mathbf{a}\mathbf{x}_1 + \mathbf{e}_1)/2^{t_1} \rceil$
5: return $(\mathsf{pk} := (\mathsf{seed}, \mathbf{y}_1), \mathsf{sk} := \mathbf{x}_1)$
```

Algorithm 20 $(\mathsf{ct}, \mathsf{key}) \leftarrow \mathsf{Encaps}(\mathsf{pk})$

```
1: $\mathbf{x}_2, \mathbf{e}_2, \mathbf{e}'_2 \leftarrow D_{\mathbb{Z}^n,\sigma}$
2: $\mathbf{a} := \mathsf{Gen}(\mathsf{seed})$
3: $\mathbf{y}_2 := \lfloor (\mathbf{a}\mathbf{x}_2 + \mathbf{e}_2)/2^{t_2} \rceil$
4: $\sigma_2 := 2^{t_1}\mathbf{y}_1 \cdot \mathbf{x}_2 + \mathbf{e}'_2$
5: $(\mathbf{k}_2, \mathbf{v}) \leftarrow \mathsf{Con}(\sigma_2, \mathsf{params})$
6: parse the vector $\mathbf{k}_2$ into $\Delta := \lfloor n/(N_H + n_H) \rfloor$ blocks, say $\mathbf{k}_2^{(1)}, \cdots, \mathbf{k}_2^{(\Delta)}$, each of size $N_H + n_H$
7: parse every $\mathbf{k}_2^{(i)}$ into the form
       $\left( x_0^{(i)} \in \mathbb{Z}_2, \mathbf{x}^{(i)} \in \mathbb{Z}_2^{(N_H-1)}, \mathbf{p}^{(i)} \in \mathbb{Z}_2^{n_H} \right)$
8: $\mathbf{v}' := \left( \mathsf{Encode}_{\mathcal{C}}(\mathbf{x}^{(i)}) \oplus \mathbf{k}_1^{(i)} \right)_{i \in [\Delta]}$
9: return $\left( \mathsf{ct} := (\mathbf{y}_2, \mathbf{v}, \mathbf{v}'), \mathsf{key} := \left(\mathbf{x}^{(i)}\right)_{i \in [\Delta]} \right)$
```

Algorithm 21 $\mathsf{key}' \leftarrow \mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$

```
1: $\sigma_1 := 2^{t_2}\mathbf{y}_2 \cdot \mathbf{x}_1$
2: $\mathbf{k}_1 := \mathsf{Rec}(\sigma_1, \mathbf{v}, \mathsf{params})$
3: parse the vector $\mathbf{k}_1$ into $\Delta := \lfloor n/(N_H + n_H) \rfloor$ blocks, say $\mathbf{k}_1^{(1)}, \cdots, \mathbf{k}_1^{(\Delta)}$, each of size $N_H + n_H$
4: parse the vector $\mathbf{v}'$ into $\Delta$ blocks, say $\mathbf{v}'_1, \cdots, \mathbf{v}'_\Delta$, each of size $N_H + n_H$
5: return $\mathsf{key}' := \left( \mathsf{Decode}_{\mathcal{C}}(\mathbf{k}_2^{(i)} \oplus \mathbf{v}'_i) \right)_{i \in [\Delta]}$
```

Note: the above pseudo-codes describe how three algorithms works in general. Notice that $n/(N_H+n_H)$ may not be a positive integer in practice; in particular, it is *not* a positive integer in our software implementation. In this case, some coefficients in $\mathbf{v}, \sigma_1, \sigma_2, \mathbf{k}_1, \mathbf{k}_2$ will not contribute to the generation of the shared secret key.

8.4 Reducing Error Rate with Lattice Code in E_8

In this section, we further consider the approach to lower the error rate, and develop new lattice code in E_8. We divide the coefficients of the polynomial $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ into $\hat{n} = n/8$ groups, where each group is composed of 8 coefficients. In specific, denote $R = \mathbb{Z}[x]/(x^8+1), R_q = R/qR, K = \mathbb{Q}[x]/(x^8+1)$ and $K_{\mathbb{R}} = K \otimes \mathbb{R} \simeq \mathbb{R}[x]/(x^8+1)$. Then the polynomial $\boldsymbol{\sigma}_1$ can be represented as $\boldsymbol{\sigma}_1(x) = \sigma_0(x^{\hat{n}}) + \sigma_1(x^{\hat{n}})x + \cdots + \sigma_{\hat{n}-1}(x^{\hat{n}})x^{\hat{n}-1}$, where $\sigma_i(x) \in R_q$ for $i = 0, 1, \ldots \hat{n}$. $\boldsymbol{\sigma}_2$ can be divided in the same way. Then we only need to construct the reconciliation mechanism for each $\sigma_i(x)$, and finally combine the keys together. To do this, we need to first introduce the lattice E_8 and its encoding and decoding.

8.4.1 Combining AKCN with Lattice Code in E_8

We construct lattice E_8 from the Extended Hamming Code in dimension 8, which is denoted as H_8 for presentation simplicity. H_8 refers to the 4-dimension linear subspace of 8-dimension linear space $\mathbb{Z}_2^8$.

$$H_8 = \{\mathbf{c} \in \mathbb{Z}_2^8 \mid \mathbf{c} = \mathbf{z}\mathbf{H} \bmod 2, \mathbf{z} \in \mathbb{Z}^4\}$$

where

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

The encoding algorithm is straightforward: given a 4-bit string $\mathbf{k}_1$, calculate $\mathbf{k}_1\mathbf{H}$. This operation can be done efficiently by bitwise operations. We combine this encoding with AKCN (for the special case of $m = 2$), which is referred to as AKCN-E8-RLWE for presentation simplicity. The complete algorithm is shown in Algorithm 22. In this work, we focus on the combination of AKCN with encoding/decoding in E_8, and the extension to OKCN is straightforward.

Algorithm 22 AKCN-E8: Con with encoding in E_8

```
1: procedure Con($\boldsymbol{\sigma}_1 \in \mathbb{Z}_q^8, \mathbf{k}_1 \in \mathbb{Z}_2^4$, params)
2:     $\mathbf{v} = \left\lfloor \frac{g}{q}\left(\boldsymbol{\sigma}_1 + \frac{q-1}{2}(\mathbf{k}_1\mathbf{H} \bmod 2)\right)\right\rceil \bmod g$
3:     return $\mathbf{v}$
4: end procedure
```

The decoding algorithm finds the solution of the closest vector problem (CVP) for the lattice E_8. For any given $\mathbf{x} \in \mathbb{R}^8$, CVP asks which lattice point in E_8 is closest to $\mathbf{x}$. Based on the structure of E_8, we propose an efficient decoding algorithm. Let $C = \{(x_1, x_1, x_2, x_2, x_3, x_3, x_4, x_4) \in \mathbb{Z}_2^8 \mid x_1+x_2+x_3+x_4 = 0 \bmod 2\}$. In fact, C is spanned by the up most three rows of $\mathbf{H}$. Hence, $E_8 = C \cup (C + \mathbf{c})$, where $\mathbf{c} = (0, 1, 0, 1, 0, 1, 0, 1)$ is the last row of $\mathbf{H}$. For a given $\mathbf{x} \in \mathbb{R}^8$, to solve CVP of $\mathbf{x}$ in E_8, we solve CVP of $\mathbf{x}$ and $\mathbf{x} - \mathbf{c}$ in C, and then choose the one that has smaller distance.

Algorithm 23 AKCN-E8: Rec with decoding in E_8

```
1: procedure Rec($\boldsymbol{\sigma}_2 \in \mathbb{Z}_q^8, \mathbf{v} \in \mathbb{Z}_g^8$, params)
2:     $\mathbf{k}_2 = \mathsf{Decode}_{\mathsf{E_8}}\left(\left\lfloor \frac{q}{g}\mathbf{v}\right\rceil - \boldsymbol{\sigma}_2\right)$
3:     return $\mathbf{k}_2$
4: end procedure
```

Then we consider how to solve CVP in C. For an $\mathbf{x} \in \mathbb{R}^8$, we choose $(x_1, x_2, x_3, x_4) \in \mathbb{Z}_2^4$, such that $(x_1, x_1, x_2, x_2, x_3, x_3, x_4, x_4)$ is closest to $\mathbf{x}$. However, $x_1+x_2+x_3+x_4 \bmod 2$ may equal to 1. In such cases, we choose the 4-bit string (x_1', x_2', x_3', x_4') such that $(x_1', x_1', x_2', x_2', x_3', x_3', x_4', x_4')$ is secondly closest to $\mathbf{x}$. Note that (x_1', x_2', x_3', x_4') has at most one-bit difference from (x_1, x_2, x_3, x_4). The detailed algorithm is depicted in Algorithm 24. Considering potential timing attack, all the "if" conditional statements can be implemented by constant time bitwise operations. In practice, Decode_C^{00} and Decode_C^{01} are implemented as two subroutines.

For algorithm 24, in Decode_{E_8}, we calculate $\mathsf{cost}_{i,b}$, where $i = 0, 1, \ldots, 7, b \in \{0, 1\}$, which refer to the contribution to the total 2-norm when $x_i = b$. Decode_C^{00} solves the CVP in lattice C, and Decode_C^{01} solves the CVP in lattice $C + \mathbf{c}$. Then we choose the one that has smaller distance. $\mathsf{Decode}_C^{b_0b_1}$ calculates the $k_i, i = 0, 1, 2, 3$ such that $\frac{q-1}{2}(k_0 \oplus b_0, k_0 \oplus b_1, k_1 \oplus b_0, k_1 \oplus b_1, k_2 \oplus b_0, k_2 \oplus b_1, k_3 \oplus b_0, k_3 \oplus b_1)$ is closest to $\mathbf{x}$. We use min_d and min_i to find the second closest vector. Finally, we check the parity to decide which one should be returned.

The following theorem gives a condition of success of the encoding and decoding algorithm in Algorithm 22 and Algorithm 23. For simplicity, for any $\boldsymbol{\sigma} = (x_0, x_1, \ldots, x_7) \in \mathbb{Z}_q^8$, we define $\|\boldsymbol{\sigma}\|_{q,2}^2 = \sum_{i=0}^{7} |x_i|_q^2$.

Theorem 8.3. *If* $\|\boldsymbol{\sigma}_1 - \boldsymbol{\sigma}_2\|_{q,2} \leq (q-1)/2 - \sqrt{2}\left(\frac{q}{g}+1\right)$, *then* $\mathbf{k}_1$ *and* $\mathbf{k}_2$ *calculated by* Con *and* Rec *are equal.*

Algorithm 24 Decoding in E_8 and C

```
1: procedure $\mathsf{Decode}_{E_8}(\mathbf{x} \in \mathbb{Z}_q^8)$
2:     for $i = 0 \dots 7$ do
3:         $\mathsf{cost}_{i,0} = |x_i|_q^2$
4:         $\mathsf{cost}_{i,1} = |x_i + \frac{q-1}{2}|_q^2$
5:     end for
6:     $(\mathbf{k}^{00}, \mathsf{TotalCost}^{00}) \leftarrow \mathsf{Decode}_C^{00}(\mathsf{cost}_{i \in 0 \dots 7, b \in \{0,1\}})$
7:     $(\mathbf{k}^{01}, \mathsf{TotalCost}^{01}) \leftarrow \mathsf{Decode}_C^{01}(\mathsf{cost}_{i \in 0 \dots 7, b \in \{0,1\}})$
8:     if $\mathsf{TotalCost}^{00} < \mathsf{TotalCost}^{01}$ then
9:         $b = 0$
10:    else
11:        $b = 1$
12:    end if
13:    $(k_0, k_1, k_2, k_3) \leftarrow \mathbf{k}^{0b}$
14:    $\mathbf{k}_2 = (k_0, k_1 \oplus k_0, k_3, b)$
15:    return $\mathbf{k}_2$
16: end procedure
17: procedure $\mathsf{Decode}_C^{b_0 b_1}(\mathsf{cost}_{i \in 0 \dots 7, b \in \{0,1\}} \in \mathbb{Z}^{8 \times 2})$
18:    $min_d = +\infty$
19:    $min_i = 0$
20:    $\mathsf{TotalCost} = 0$
21:    for $j = 0 \dots 3$ do
22:        $c_0 \leftarrow \mathsf{cost}_{2j, b_0} + \mathsf{cost}_{2j+1, b_1}$
23:        $c_1 \leftarrow \mathsf{cost}_{2j, 1-b_0} + \mathsf{cost}_{2j+1, 1-b_1}$
24:        if $c_0 < c_1$ then
25:            $k_i \leftarrow 0$
26:        else
27:            $k_i \leftarrow 1$
28:        end if
29:        $\mathsf{TotalCost} \leftarrow \mathsf{TotalCost} + c_{k_i}$
30:        if $c_{1-k_i} - c_{k_i} < min_d$ then
31:            $min_d \leftarrow c_{1-k_i} - c_{k_i}$
32:            $min_i \leftarrow i$
33:        end if
34:    end for
35:    if $k_0 + k_1 + k_2 + k_3 \bmod 2 = 1$ then
36:        $k_{min_i} \leftarrow 1 - k_{min_i}$
37:        $\mathsf{TotalCost} \leftarrow \mathsf{TotalCost} + min_d$
38:    end if
39:    $\mathbf{k} = (k_0, k_1, k_2, k_3)$
40:    return $(\mathbf{k}, \mathsf{TotalCost})$
41: end procedure
```

Proof. The minimal Hamming distance of the Extended Hamming code H_8 is 4. Hence, the minimal distance in the lattice we used is $\frac{1}{2}\sqrt{\left(\frac{q-1}{2}\right)^2 \times 4} = (q-1)/2$.

We can find $\boldsymbol{\varepsilon}, \boldsymbol{\varepsilon}_1 \in [-1/2, 1/2]^8, \boldsymbol{\theta} \in \mathbb{Z}^8$ such that

$$\left\lfloor \frac{q}{g}\mathbf{v} \right\rceil - \boldsymbol{\sigma}_2 = \frac{q}{g}\mathbf{v} + \boldsymbol{\varepsilon} - \boldsymbol{\sigma}_2 = \frac{q}{g}\left(\frac{g}{q}\left(\boldsymbol{\sigma}_1 + \frac{q-1}{2}\mathbf{k}_1\mathbf{H}\right) + \boldsymbol{\varepsilon} + \boldsymbol{\theta}g\right) + \boldsymbol{\varepsilon}_1 - \boldsymbol{\sigma}_2$$
$$= (\boldsymbol{\sigma}_1 - \boldsymbol{\sigma}_2) + \frac{q-1}{2}\mathbf{k}_1\mathbf{H} + \frac{q}{g}\boldsymbol{\varepsilon} + \boldsymbol{\varepsilon}_1 + \boldsymbol{\theta}q$$

Hence, the bias from $\frac{q-1}{2}\mathbf{k}_1\mathbf{H}$ is no larger than $\|\boldsymbol{\sigma}_1 - \boldsymbol{\sigma}_2\|_{q,2} + \frac{q}{g}\|\boldsymbol{\varepsilon}\| + \sqrt{2} \le \|\boldsymbol{\sigma}_1 - \boldsymbol{\sigma}_2\|_{q,2} + \sqrt{2}\left(\frac{q}{g} + 1\right)$. If this value is less than the minimal distance $(q-1)/2$, the decoding will be correct, which implies $\mathbf{k}_1 = \mathbf{k}_2$. □

Parameters and implementation. The parameters and performance of AKCN-E8 are given in Table 20. We provide a script to calculate the concrete error rate. For AKCN-E8-256, the deviation in our parameter set ($\sigma = \sqrt{21}$) is quite large, which requires more many random bits to sample. However, the generation of random bits costs a lot of time. Frodo uses a table to generate a discrete distribution that is very close to the rounded Gaussian. However, in our parameter set for AKCN-E8-256, the table will be too large to sample efficiently. Hence, we propose the distribution $B^{a,b}$, where a and b are two integers.

Algorithm 25 Sample r from $B^{a,b}$

```
1: $r \leftarrow \sum_{i=1}^{a} \mathsf{getOneRandomBit}() + 2 * \sum_{i=1}^{b} \mathsf{getOneRandomBit}() - \left(\frac{a}{2} + b\right)$
```

The variation of r in Algorithm 25 is $\frac{a}{4} + b$, and the expect value of r is 0. By the *central limit theorem*, the distribution of r is close to a discrete Gaussian. In our implementation, we choose $a = 24, b = 15$, and the summation of the random bits are calculated by fast bit counting. Recall that the Renyi divergence increases as a increases. Hence, $B^{24,15}$ and rounded Gaussian of variance 21 are more close compared to Ψ_{16} and rounded Gaussian of variance 8. We use a larger a than NewHope so that the potential security decline can be smaller, although no attacks known make use of the information of different noise distributions.

	$\lvert K\rvert$	n	q	σ (σ')	g	t	pq-sec (t-sec)	err	pk (B)	cipher (B)	bw. (B)
AKCN-E8-256	256	512	12289	$\sqrt{21}$ ($\sqrt{22}$)	2^6	2	128 (129)	2^{-34}	800	1152	1952
AKCN-E8-512	512	1024	12289	$\sqrt{8}$ ($\sqrt{10}$)	2^4	3	255 (262)	$2^{-63.3}$	1440	1920	3360
$\sigma = \sqrt{8}$	512	1024	12289	$\sqrt{8}$ ($\sqrt{9}$)	2^4	2	255 (258)	2^{-98}	1568	2048	3616
	512	1024	12289	$\sqrt{8}$ ($\sqrt{10}$)	2^5	3	255 (262)	$2^{-81.6}$	1440	2048	3488
	512	1024	12289	$\sqrt{8}$ ($\sqrt{9}$)	2^5	2	255 (258)	$2^{-124.4}$	1568	2176	3744
	512	1024	12289	$\sqrt{8}$ ($\sqrt{9}$)	2^6	2	255 (258)	$2^{-138.7}$	1568	2304	3872
AKCN-E8-512	512	1024	12289	$\sqrt{6}$ ($\sqrt{10}$)	2^4	4	246 (262)	$2^{-35.6}$	1312	1792	3104
($\sigma = \sqrt{6}$)	512	1024	12289	$\sqrt{6}$ ($\sqrt{8}$)	2^4	3	246 (255)	$2^{-109.4}$	1440	1920	3360
	512	1024	12289	$\sqrt{6}$ ($\sqrt{10}$)	2^5	4	246 (262)	$2^{-47.2}$	1312	1920	3232
	512	1024	12289	$\sqrt{6}$ ($\sqrt{8}$)	2^3	3	246 (255)	$2^{-60.7}$	1440	1792	3232
	512	1024	12289	$\sqrt{6}$ ($\sqrt{8}$)	2^5	3	246 (255)	$2^{-138.4}$	1440	2048	3488

Table 19: Parameters for AKCN-E8-RLWE. "pk(B)" refers to the size of $(\mathbf{y}_1, \mathsf{seed})$ in bytes; "cipher(B)" refers to the size of $(\mathbf{y}_2, \mathbf{v})$. The underlying noise distribution is Ψ_η with $\sigma = \sqrt{\eta/2}$; "pq-sec" (resp., "t-sec") refers to the security against the best known quantum attacks against the underlying lattice problem without considering the effect of t (resp., by heuristically viewing the standard deviation of the noise as $\sigma' = \sqrt{(2\sigma^2 + 2^{t-1})/2}$).

8.5 On the Desirability of OKCN/AKCN-SEC and OKCN/AKCN-E8

Compared to NewHope, OKCN/AKCN-SEC and OKCN/AKCN-E8 are more desirable, on the following grounds:

- To our knowledge, OKCN/AKCN-SEC schemes are the simplest RLWE-based KE protocols *with error probability that can be viewed negligible in practice*, which are better suitable for hardware or software implementations than encoding and decoding the four-dimensional lattice $\tilde{D}_4$. Note that SEC can be implemented with simple bit operations. Moreover, with probability about $1-2^{-40}$, the decoding only involves the XOR operations in Line-1 of Algorithm 14, which is extremely simple and fast.

- AKCN-SEC can be directly transformed into a CPA-secure PKE scheme for encrypting 837-bit messages, while AKCN4:1-RLWE and NewHope-simple are for encrypting 256-bit messages.

- It is more desirable to have KE protocols that directly share or transport keys of larger size. On the one hand, it is commonly expected that, in the post-quantum era, symmetric-key cryptographic primitives like AES need larger key sizes, in view of the quadratic speedup by Grover's search algorithm and the possibility of more sophisticated quantum attacks [KM10, KLL15] against symmetric-key cryptography. Indeed, to our knowledge, the post-quantum security of NewHope is evaluated as a stand-alone protocol, without considering possible quantum attacks targeting the use of shared-key. On the other hand, in some more critical application areas than public commercial usage, larger key size actually has already been mandated nowadays. Note that for NewHope, AKCN4:1-RLWE, and NewHope-simple, if we want a 512-bit shared-key (which is necessary for ensuring 256-bit post-quantum security) they have to use a polynomial of degree 2048 which can be significantly less efficient.

- As clarified, the SEC approach fails only when there are more than one bit errors in some block, and is versatile in the sense: the smaller (resp., larger) is the block size n_H, the lower the error probability (resp., bandwidth expansion) will be.

- OKCN/AKCN-SEC and OKCN/AKCN-E8 are more versatile and flexible than NewHope, allowing more useful trade-offs among the parameters and performance.

- OKCN/AKCN-SEC vs. OKCN/AKCN-E8. OKCN/AKCN-SEC has larger key size and is simpler. In comparison, on the system parameters, OKCN/AKCN-E8 can have lower error rate, smaller bandwidth and stronger security simultaneously, but at the price of more complicated implementation. We may prefer OKCN/AKCN-SEC, from the viewpoint of system simplicity and easy implementation.

9 MLWE-Based Key Exchange from KC and AKC

Recall that $\mathcal{R} = \mathbb{Z}[X]/(X^n+1)$, and $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. Let l be a positive integer parameter. Let S_η denote a distribution on all elements $w \in \mathcal{R}$ such that $\|w\|_\infty \leq \eta$.[11] The Module-LWE (MLWE)

[11] A typical instantiation of S_η, as proposed in [BDK+17], is based on the following centered binomial distribution: sample $(a_1, \cdots, a_\eta, b_1, \cdots, b_\eta) \leftarrow \{0,1\}^{2\eta}$, and output $\sum_{i=1}^{\eta}(a_i - b_i)$.

Initiator

$\mathsf{seed} \leftarrow \{0,1\}^\kappa$
$\mathbf{A} = \mathsf{Gen}(\mathsf{seed}) \in \mathcal{R}_q^{l \times l}$
$\mathbf{X}_1, \mathbf{E}_1 \leftarrow S_\eta^{l \times 1}$
$\mathbf{Y}_1 = \lfloor (\mathbf{A}\mathbf{X}_1 + \mathbf{E}_1)/2^{t_1} \rceil$

Responder

$$\xrightarrow{\mathsf{seed}, \mathbf{Y}_1}$$

$\mathbf{A} = \mathsf{Gen}(\mathsf{seed})$
$\mathbf{X}_2, \mathbf{E}_2 \leftarrow S_\eta^{l \times 1}, \mathbf{E}_\sigma \leftarrow S_\eta$
$\mathbf{Y}_2 = \lfloor (\mathbf{A}^T\mathbf{X}_2 + \mathbf{E}_2)/2^{t_2} \rceil$
$\mathbf{\Sigma}_2 = 2^{t_1}\mathbf{Y}_1^T\mathbf{X}_2 + \mathbf{E}_\sigma$
$(\mathbf{K}_2, \mathbf{V}) \leftarrow \mathsf{Con}(\mathbf{\Sigma}_2, \mathsf{params})$

$$\xleftarrow{\mathbf{Y}_2, \mathbf{V}}$$

$\mathbf{\Sigma}_1 = \mathbf{X}_1^T(2^{t_2}\mathbf{Y}_2)$
$\mathbf{K}_1 = \mathsf{Rec}(\mathbf{\Sigma}_1, \mathbf{V}, \mathsf{params})$

Figure 14: Generic construction of OKCN-MLWE

problem is introduced in [LS15], which is a generalization of the RLWE problem. We make use of the definitions described in [BDK+17].

- **MLWE distribution.** The MLWE distribution is defined on $\mathcal{R}_q^l \times \mathcal{R}_q$ induced by pairs $(\mathbf{a}_i, \mathbf{b}_i)$, where $\mathbf{a}_i \leftarrow \mathcal{R}_q^l$ is uniform and $\mathbf{b} = \mathbf{a}_i^T\mathbf{s} + \mathbf{e}_i$ with $\mathbf{s} \leftarrow S_\eta^l$ common to all samples and $\mathbf{e}_i \leftarrow S_\eta$ fresh for every sample.

- **MLWE assumption.** The MLWE problem consists in recovering $\mathbf{s}$ from polynomially many samples chosen from the MLWE distribution. More precisely, for an algorithm A, we define
$$\mathbf{Adv}_{h,l,\eta}^{\text{mlwe}}(A) = \Pr\left[\mathbf{x} = \mathbf{s} : \begin{array}{l} \mathbf{A} \leftarrow \mathcal{R}_q^{h \times l}; (\mathbf{s}, \mathbf{e}) \leftarrow S_\eta^l \times S_\eta^h; \\ b \leftarrow \mathbf{A}\mathbf{s} + \mathbf{e}; \mathbf{x} \leftarrow A(\mathbf{A}, \mathbf{b}); \end{array}\right].$$
We say that the (t, ε) $\text{MLWE}_{h,l,\eta}$ hardness assumption holds if no algorithm A running in time at most t has an advantage greater than ε.

9.1 Generic Construction of MLWE-Based KE

Let $KC = (\mathsf{Con}, \mathsf{Rec}, \mathsf{params})$ be a *correct* and *secure* KC or AKC scheme with parameters $\mathsf{params} = (q, m, g, d)$, where $m = 2$ in this section. When Con and Rec are applied to a polynomial in $\mathcal{R}_q$, they are applied to each coefficients of the polynomial respectively. The generic construction of key exchange from MLWE is described in Figure 14 and Figure 15. When being cased into the public-key setting, its specification is given in Section 9.1.1. We remark that the underlying AKC (resp., KC) can be any one of AKCN, AKCN4:1, AKCN-SEC, AKCN-E8 (resp., OKCN, OKCN-SEC, OKCN-E8). Here, for simplicity and symmetry, we assume the same number of tail bits are chopped off from both Y_1 and Y_2 by setting $t = t_1 = t_2 \geq 0$.

The construction of MLWE-based KE is a direct adaptation of the LWE-based KE from KC/AKC presented in Figure 7 in Section 6. When $m = 2$ and g is power-of-two, the AKCN-based implementation is actually the CPA-secure Kyber scheme proposed in [BDK+17]. The

$$
\begin{array}{ccc}
\underline{\text{Initiator}} & & \underline{\text{Responder}} \\
\mathsf{seed} \leftarrow \{0,1\}^{\kappa} & & \\
\mathbf{A} = \mathsf{Gen}(\mathsf{seed}) \in \mathcal{R}_q^{l\times l} & & \\
\mathbf{X}_1, \mathbf{E}_1 \leftarrow S_\eta^{l\times 1} & & \\
\mathbf{Y}_1 = \lfloor (\mathbf{A}\mathbf{X}_1 + \mathbf{E}_1)/2^{t_1} \rceil & & \\
 & \xrightarrow{\mathsf{seed}, \mathbf{Y}_1} & \\
 & & \mathbf{K}_2 \leftarrow \mathbb{Z}_m^n \\
 & & \mathbf{A} = \mathsf{Gen}(\mathsf{seed}) \\
 & & \mathbf{X}_2, \mathbf{E}_2 \leftarrow S_\eta^{l\times 1}, \mathbf{E}_\sigma \leftarrow S_\eta \\
 & & \mathbf{Y}_2 = \left\lfloor (\mathbf{A}^T\mathbf{X}_2 + \mathbf{E}_2)/2^{t_2} \right\rceil \\
 & & \mathbf{\Sigma}_2 = 2^{t_1}\mathbf{Y}_1^T\mathbf{X}_2 + \mathbf{E}_\sigma \\
 & & \mathbf{V} \leftarrow \mathsf{Con}(\mathbf{\Sigma}_2, \mathbf{K}_2, \mathsf{params}) \\
 & \xleftarrow{\mathbf{Y}_2, \mathbf{V}} & \\
\mathbf{\Sigma}_1 = \mathbf{X}_1^T(2^{t_2}\mathbf{Y}_2) & & \\
\mathbf{K}_1 = \mathsf{Rec}(\mathbf{\Sigma}_1, \mathbf{V}, \mathsf{params}) & &
\end{array}
$$

Figure 15: Generic construction of AKCN-MLWE, where $m = 2$ in this work

parameters and performance of OKCN-MLWE and AKCN-MLWE are presented in Table 20. In practice, we prefer to use the parameter set OKCN-MLWE-PKE-1, which is also the parameter set used in our actual implementation. We note that, when $\eta = 2$, there may be potential combinational attacks, but the much larger η' voids such potential combinational attacks.

	$\lvert K\rvert$	n	q	η (η')	g	t	l	pq-sec (t-sec)	err	pk (B)	cipher (B)	bw. (B)
OKCN-MLWE-KE-light	256	256	7681	5 (13)	2^3	4	2	102 (116)	$2^{-36.2}$	608	704	1312
OKCN-MLWE-KE	256	256	7681	2 (10)	2^2	4	3	147 (183)	$2^{-50.1}$	896	960	1856
OKCN-MLWE-PKE-light	256	256	7681	5 (9)	2^3	3	2	102 (111)	$2^{-105.5}$	672	768	1440
OKCN-MLWE-PKE-1	256	256	7681	2 (10)	2^5	4	3	147 (183)	$2^{-80.3}$	896	1056	1952
OKCN-MLWE-PKE-2	256	256	7681	2 (6)	2^2	3	3	147 (171)	$2^{-166.4}$	992	1056	2048
AKCN-MLWE-PKE-light	256	256	7681	5 (9)	2^3	3	2	102 (111)	$2^{-105.5}$	672	800	1472
AKCN-MLWE-PKE-1	256	256	7681	2 (10)	2^6	4	3	147 (183)	$2^{-80.3}$	896	1088	1984
AKCN-MLWE-PKE-2	256	256	7681	2 (6)	2^3	3	3	147 (171)	$2^{-166.4}$	992	1088	2080
OKCN-MLWE-Alt1	256	256	7681	4	2^2	2	3	161 (171)	$2^{-142.7}$	1088	1152	2240
AKCN-MLWE-Alt1(Kyber)	256	256	7681	4	2^3	2	3	161 (171)	$2^{-142.7}$	1088	1184	2272
OKCN-MLWE-Alt2	256	256	7681	4	2^2	3	3	161	$2^{-71.9}$	992	1056	2048
OKCN-MLWE-Alt3	256	256	7681	4	2^4	3	3	161	2^{-109}	992	1120	2112
OKCN-MLWE-Alt3	256	256	7681	4	2^4	4	3	161	$2^{-34.5}$	896	1024	1920

Table 20: Parameters for OKCN/AKCN-MLWE. $\eta' = \eta + 2^{t-1}$; "pq-sec (t-sec)" refers to the best known quantum attack against the underlying lattice problem w.r.t. η (resp., η').

9.1.1 KEM Specificiation of OKCN-MLWE in Public-Key Setting

When being casted into the public-key setting, the specification of OKCN-MLWE is given below. The adaption to the specification of AKCN-MLWE is straightforward, and is specified in Section 9.1.2. For presentation simplicity, we simply set the resultant shared-key to be $\mathbf{K} = \mathbf{K}_1 = \mathbf{K}_2$. In actual implementation of KEM, the shared-key is derived from $\mathbf{K}$ and the interaction transcript via some key derivation function (e.g., a cryptographic hash function or HMAC, etc).

Algorithm 26 $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$

```
1: $\mathsf{seed} \leftarrow \{0,1\}^\kappa$
2: $\mathbf{A} := \mathsf{Gen}(\mathsf{seed})$
3: $\mathbf{X}_1, \mathbf{E}_1 \leftarrow S_\eta^{l\times 1}$
4: $\mathbf{Y}_1 := \lfloor(\mathbf{A}\mathbf{X}_1 + \mathbf{E}_1)/2^{t_1}\rceil$
5: return $(\mathsf{pk} := (\mathsf{seed}, \mathbf{Y}_1), \mathsf{sk} := \mathbf{X}_1)$
```

Algorithm 27 $(\mathsf{ct},\mathsf{key}) \leftarrow \mathsf{Encaps}(\mathsf{pk})$

```
1: $\mathbf{X}_2, \mathbf{E}_2 \leftarrow S_\eta^{l\times 1}, \mathbf{E}_\sigma \leftarrow S_\eta$
2: $\mathbf{A} := \mathsf{Gen}(\mathsf{seed})$
3: $\mathbf{Y}_2 := \lfloor(\mathbf{A}^T\mathbf{X}_2 + \mathbf{E}_2)/2^{t_2}\rceil$
4: $\mathbf{\Sigma}_2 := 2^{t_1}\mathbf{Y}_1^T\mathbf{X}_2 + \mathbf{E}_\sigma$
5: $(\mathbf{K}_2, \mathbf{V}) \leftarrow \mathsf{Con}(\mathbf{\Sigma}_2, \mathsf{params})$
6: return $(\mathsf{ct} := (\mathbf{Y}_2, \mathbf{V}), \mathsf{key} := \mathbf{K}_2)$
```

Algorithm 28 $\mathsf{key}' \leftarrow \mathsf{Decaps}(\mathsf{sk},\mathsf{ct})$

```
1: $\mathbf{\Sigma}_1 := \mathbf{X}_1^T(2^{t_2}\mathbf{Y}_2)$
2: $\mathbf{K}_1 := \mathsf{Rec}(\mathbf{\Sigma}_1, \mathbf{V}, \mathsf{params})$
3: return $\mathsf{key}' := \mathbf{K}_1$
```

9.1.2 KEM Specificiation of AKCN-MLWE in Public-Key Setting

Algorithm 29 $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}()$

```
1: $\mathsf{seed} \leftarrow \{0,1\}^\kappa$
2: $\mathbf{A} := \mathsf{Gen}(\mathsf{seed})$
3: $\mathbf{X}_1, \mathbf{E}_1 \leftarrow S_\eta^{l\times 1}$
4: $\mathbf{Y}_1 := \lfloor(\mathbf{A}\mathbf{X}_1 + \mathbf{E}_1)/2^{t_1}\rceil$
5: return $(\mathsf{pk} := (\mathsf{seed}, \mathbf{Y}_1), \mathsf{sk} := \mathbf{X}_1)$
```

Algorithm 30 $(\mathsf{ct},\mathsf{key}) \leftarrow \mathsf{Encaps}(\mathsf{pk})$

```
1: $\mathbf{K}_2 \leftarrow \mathbb{Z}_m^n$
2: $\mathbf{X}_2, \mathbf{E}_2 \leftarrow S_\eta^{l\times 1}, \mathbf{E}_\sigma \leftarrow S_\eta$
3: $\mathbf{A} := \mathsf{Gen}(\mathsf{seed})$
4: $\mathbf{Y}_2 := \lfloor(\mathbf{A}^T\mathbf{X}_2 + \mathbf{E}_2)/2^{t_2}\rceil$
5: $\Sigma_2 := 2^{t_1}\mathbf{Y}_1^T\mathbf{X}_2 + \mathbf{E}_\sigma$
6: $\mathbf{V} \leftarrow \mathsf{Con}(\Sigma_2, \mathbf{K}_2, \mathsf{params})$
7: return $(\mathsf{ct} := (\mathbf{Y}_2, \mathbf{V}), \mathsf{key} := \mathbf{K}_2)$
```

Algorithm 31 $\mathsf{key}' \leftarrow \mathsf{Decaps}(\mathsf{sk},\mathsf{ct})$

```
1: $\Sigma_1 := \mathbf{X}_1^T(2^{t_2}\mathbf{Y}_2)$
2: $\mathbf{K}_1 := \mathsf{Rec}(\Sigma_1, \mathbf{V}, \mathsf{params})$
3: return $\mathsf{key}' := \mathbf{K}_1$
```

9.2 Error Rate Analysis and Parameter Selection

Denote $\boldsymbol{\varepsilon}_2 = \mathbf{A}^T\mathbf{X}_2 + \mathbf{E}_2 - 2^t\lfloor(\mathbf{A}^T\mathbf{X}_2 + \mathbf{E}_2)/2^t\rceil$, and $\boldsymbol{\varepsilon}_1 = \mathbf{A}\mathbf{X}_1 + \mathbf{E}_1 - 2^t\lfloor(\mathbf{A}\mathbf{X}_1 + \mathbf{E}_1)/2^t\rceil$. Then we have

$$\mathbf{\Sigma}_1 - \mathbf{\Sigma}_2 = \mathbf{X}_1^T(2^t\mathbf{Y}_2) - (2^t\mathbf{Y}_1^T\mathbf{X}_2 + \mathbf{E}_\sigma) \tag{12}$$

$$= 2^t\mathbf{X}_1^T\lfloor(\mathbf{A}^T\mathbf{X}_2 + \mathbf{E}_2)/2^t\rceil - 2^t\lfloor(\mathbf{A}\mathbf{X}_1 + \mathbf{E}_1)/2^t\rceil\mathbf{X}_2 + \mathbf{E}_\sigma \tag{13}$$

$$= \mathbf{X}_1^T(\mathbf{A}^T\mathbf{X}_2 + \mathbf{E}_2 - \boldsymbol{\varepsilon}_2) - ((\mathbf{A}\mathbf{X}_1 + \mathbf{E}_1 - \boldsymbol{\varepsilon}_1) + \boldsymbol{E}_\sigma) \tag{14}$$

$$= \mathbf{X}_1^T(\mathbf{E}_2 - \boldsymbol{\varepsilon}_2) + (\mathbf{E}_1 - \boldsymbol{\varepsilon}_1)^T\mathbf{X}_2 + \mathbf{E}_2 \tag{15}$$

From MLWE assumption, $(\mathbf{A}, \mathbf{A}^T\mathbf{X}_2 + \mathbf{E}_2)$ is indistinguishable with $(\mathbf{A}, \mathbf{U})$, where $\mathbf{U}$ is subjected to the uniform distribution, $\boldsymbol{\varepsilon}_i (i = 1, 2)$ should be closed to $\mathbf{U} - 2^t\lfloor\mathbf{U}/2^t\rceil$. We can roughly regard each coefficients of polynomials in $\mathbf{U} - 2^t\lfloor\mathbf{U}/2^t\rceil$ as uniform distribution over $[-2^{t-1}, 2^{t-1}]^n$.

Then we can calculate the standard deviation of each coefficients of polynomials in $\mathbf{\Sigma}_2 - \mathbf{\Sigma}_1$, denote it as s. We have

$$s^2 = 2nl\sigma^2\left(\sigma^2 + \frac{2^{2t}}{12}\right) + \sigma^2 \tag{16}$$

For AKCN-E8-MLWE, by the Central Limit Theorem, each coefficient of the polynomials in $\mathbf{\Sigma}_2 - \mathbf{\Sigma}_1$ is close to a Gaussian distribution. From Theorem 8.3, the AKCN-E8-MLWE scheme is correct with probability

$$\Pr\left[d' \leftarrow \chi^2(8) : \sqrt{d'} \le \left(\frac{q-1}{2} - \sqrt{2}\left(\frac{q}{g} + 1\right)\right)/s\right] \tag{17}$$

9.3 Implementation

In this section, we concentrate our introduction on the KEM implementation of the OKCN-MLWE scheme.[12] Almost all the implementations of our schemes with this submission follow the same designing principles. The suggested parameters for the implementation of OKCN-MLWE KEM scheme are:

$$n = 256, q = 6781, l = 3, g = 2^5, t = 4, m = 2, \eta = 2.$$

9.3.1 Generation of Noise Polynomials

In our implementation, each coefficient of the noise polynomial is drawn independently from the centered binomial distribution of parameter $\eta = 2$ (instead of the discrete Gaussian distribution), mostly due to the difficulty of implementing a discrete Gaussian sampler efficiently. To obtain such a noise polynomial, we first expand a seed of $31 + 1 = 32$ bytes into a uniform random array of length 128 bytes, and each byte is applied to generate two *indepenent* coefficients in the obvious manner. Note that although each coefficient of the noise polynomial corresponds to some integer in the interval $[q - \eta, q + \eta]$, we could compress each coefficient so that $\lceil\log(1 + 2\eta)\rceil = 3$ bits suffices to represent each coefficient of the noise polynomial. Such observation enables us to represent the secret key (i.e., the $\mathbf{X}_1$ in Algorithm 26) in a compact manner.

9.3.2 The Keys and Ciphertext

In our implementation of OKCN-MLWE KEM, every public key is a pair $(\mathbf{Y}_1, \mathsf{seed})$. Here, $\mathbf{Y}_1$ is a *truncated vector* consisting of $l = 3$ *truncated polynomial*, each of size

$$l \cdot n \cdot (\lceil 1 + \log q\rceil - t)/8 = 960$$

[12]Some parts of the implementations are inspired by [BDK+17, ADPS16].

bytes, whereas the seed to generate the public matrix $\mathbf{A} \in \mathcal{R}_q^{3\times 3}$ is of size $32-1=31$ bytes.

As noted before, a secret key is a small polynomial, which is of size $l \cdot n \cdot \lceil \log(1+2\eta) \rceil / 8 = 288$ bytes. Each ciphertext is a pair $(\mathbf{Y}_2, \mathbf{V})$. Here, $\mathbf{Y}_2$ is a truncated vector, and hence is of size 960 bytes. Conversely, the array $\mathbf{V}$ is of size $n \cdot g/8 = 160$ bytes.

9.3.3 Encoding/Decoding of Objects

In addition to the random seed, we need to convert *four* types of *numerical objects* into character strings in our implementation, i.e., the truncated polynomial (or more precisely, the truncated vector), the noise polynomial, the signal, and the final shared key. Note that although each could be seen as an n-dimensional "vector" in the space $\mathbb{Z}^n$, their coefficients are of different sizes in nature. Nevertheless, each vector could be divided into $n/8 = 32$ consecutive *blocks*, each with 8 coefficients, and each block could be encoded/decoded in the similar manner. Such observation enables us to define two general procedures, compress, decompress in the io.h file, which could handle all the *foregoing* objects and thus simplify our implementation *significantly*.

Take the signal vector $\mathbf{V} = (v_0, v_1, \cdots, v_{n-1})^T$ for instance. It consists of 32 blocks, and each coefficient $v_i = v_{i,0}v_{i,1}v_{v,2}\cdots v_{i,4} \in \left\{0, 1, \cdots, 2^5-1\right\}$. It is routine to see that each block could be *encoded* into five bytes, as the following indicates:

$$v_{i,0}v_{i+1,0}\cdots v_{i+7,0}, \quad v_{i,1}v_{i+1,1}\cdots v_{i+7,1}, \quad \cdots, \quad v_{i,4}v_{i+1,4}\cdots v_{i+7,4}.$$

The *decoding* procedure is defined in the appropriate manner.

It should be stressed that the coefficients of each noise polynomial fall into the interval $[q-\eta, q+\eta]$ during computation. Hence, we should *shift* these coefficients into an appropriate interval, and then encode each block into $\lceil \log(1+2\eta) \rceil = 3$ bytes; similarly, shifting operation is necessary after the decoding process.

9.3.4 NTT Technique

In our implementation, the NTT technique is applied to speeding up the polynomial multiplication operations. In particular, the negative wrapped convolution method [LMPR08] is used to avoid the use of the trivial doubling technique in $\mathbb{Z}_q/\langle x^n+1\rangle$. The Montgomery reduced algorithm [M85], i.e., the REDC algorithm, is also applied so as to make it more efficient, with $R = 2^{18}$.

Moreover, by setting the parameters in an appropriate manner, both the NTT transform and the inverse NTT transform could be conducted by invoking the same procedure, i.e., the Poly-NTT-transform procedure in polynomial.h file. This makes our implementation more compact and more readable.

9.4 Applications to CCA-Secure PKE

The transformation from AKCN-MLWE to CCA-secure KEM is specified in detail in [BDK+17]. A CCA-secure KEM from OKCN-MLWE, which is instantiated from [HHK17], is presented in Section J. In this section, we present new construction of CCA-secure PKE scheme from AKCN-MLWE. The extensions to schemes based on RLWE, LWE and LWR are straightforward.

For schemes based on MLWE, LWE and LWR, the security parameter κ is set to be 256.[13] Let $G : \{0,1\}^* \to \{0,1\}^\kappa \times \{0,1\}^{p_1(\kappa)}$, where p_1 is a positive polynomial, and $H : \{0,1\}^* \to$

[13]For schemes based on RLWE, we may suggest $\kappa = 512$.

$\{0,1\}^\kappa$ be two cryptographic hash functions (or any secure key derivation functions). Let $KDF\{0,1\}^* \to \mathcal{K}_{\mathsf{ae}}$ be a secure key derivation function, where $\mathcal{K}_{\mathsf{ae}}$ is the key space of AEAD. We write $(\mathbf{X}_2, \mathbf{E}_2, \mathbf{E}_\sigma) \leftarrow \mathsf{Sample}(1^\kappa; \mathsf{r}_1)$ to denote the process of sampling the noises: $\mathbf{X}_2, \mathbf{E}_2 \leftarrow S_\eta^{l\times 1}$ and $\mathbf{E}_\sigma \leftarrow S_\eta$, using randomness $\mathsf{r}_1 \in \{0,1\}^{p_1(\kappa)}$. Let $\mathsf{SE} = (\mathsf{K}_{se}, \mathsf{Enc}, \mathsf{Dec})$ be an AEAD scheme, as specified in Section 2.1, and $M \in \{0,1\}^*$ be the message to be encrypted. The key-generation algorithm, the encryption algorithm $\mathcal{E}$ and the decryption algorithm $\mathcal{D}$ of the PKE scheme from AKCN-MLWE are specified in Algorithm 32, Algorithm 33 and Algorithm 34 respectively.

Some comments are in order. The design of the PKE scheme combines techniques from [FO13, D02, TU16, HHK17], but with the following modifications and considerations.

- We explicitly use authenticated encryption (rather than any one-time CCA-secure symmetric-key encryption), which is mandated in (and is thus well compatible with) some prominent existing standards like TLS1.3. In actual implementations, we recommend to use AES-GCM, or Chacha for light-weight implementations.

- The value ϖ is sent in plain in [TU16, HHK17], which plays an essential role for provable security in the quantum random oracle model. In our design, it is encrypted with AE. On the one hand, we suggest it is more prudent to get it encrypted in reality. On the other hand, this value will also be used for other purposes in our design of AKE to be presented in the subsequent section. Though encrypting it with any (one-time) CCA-secure SE scheme might be problematic, encrypting it with AEAD can only strengthen the security in reality. Formal analysis will be conducted in a separate work.

- The underlying key for AEAD is set to be $H(\mathsf{S}, \mathsf{c}_1)$. This allows more flexible implementations (e.g., when the random coins for KEM are generated with part of plaintexts as input), and is well compatible with the use of the PKE scheme in more complex systems like AKE.

- When composing a CCA-secure KEM with AEAD, decrypting the AEAD ciphertext using a valid key or a fault key can have performance differences, which may cause potential side channel attacks. Our design tries to hide such performance differences.

Algorithm 32 $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$

```
1: $\mathsf{z} \leftarrow \{0,1\}^\kappa$
2: $\mathsf{seed} \leftarrow \{0,1\}^\kappa$
3: $\mathbf{A} := \mathsf{Gen}(\mathsf{seed})$
4: $\mathbf{X}_1, \mathbf{E}_1 \leftarrow S_\eta^{l\times 1}$
5: $\mathbf{Y}_1 := \lfloor (\mathbf{A}\mathbf{X}_1 + \mathbf{E}_1)/2^{t_1} \rceil$
6: return $(\mathsf{pk} := (\mathsf{seed}, \mathbf{Y}_1), \mathsf{sk} := \{\mathbf{X}_1, \mathsf{z}, \mathsf{pk}\})$
```

Algorithm 33 $\mathsf{ct} \leftarrow \mathcal{E}_{\mathsf{pk}}(M)$

```
1:  $\mathbf{A} := \mathsf{Gen}(\mathsf{seed})$[a]
2:  $\mathsf{S} \leftarrow \{0,1\}^{\kappa}$
3:  $(\varpi, \mathsf{r}_1) = G(\mathsf{pk}, \mathsf{S})$[b]
4:  $(\mathbf{X}_2, \mathbf{E}_2, \mathbf{E}_{\sigma}) \leftarrow \mathsf{Sample}(1^{\kappa}; \mathsf{r}_1)$
5:  $\mathbf{Y}_2 := \lfloor (\mathbf{A}^T\mathbf{X}_2 + \mathbf{E}_2)/2^{t_2} \rceil$
6:  $\mathbf{\Sigma}_2 := 2^{t_1}\mathbf{Y}_1^T\mathbf{X}_2 + \mathbf{E}_{\sigma}$
7:  $\mathbf{V} \leftarrow \mathsf{Con}(\mathbf{\Sigma}_2, \mathsf{S}, \mathsf{params})$
8:  $\mathbf{K} = KDF(\mathsf{S}, \mathsf{c}_1)$, where $\mathsf{c}_1 = (\mathbf{Y}_2, \mathbf{V})$
9:  $\mathsf{c}_2 = \mathsf{Enc}_{\mathbf{K}}(\mathsf{H}, \varpi || M)$[c]
10: return $\mathsf{ct} := (\mathsf{c}_1, \mathsf{c}_2)$
```

[a] $\mathbf{A}$ can be directly specified as part of pk and sk in place of seed.

[b] In practice, we may recommend the variant of $(\varpi, \mathsf{r}_1) = \hat{G}(\mathsf{pk}, \mathsf{S}, M')$, where M' is part of M.

[c] The associated data H contains a (possibly empty) subset of c_1 and some public values determined from the application context. For simplicity, we usually do not explicitly specify the associated data.

Algorithm 34 $\mathcal{D}_{\mathsf{sk}}(\mathsf{ct} = (\mathsf{c}_1, \mathsf{c}_2))$

```
1:  $\mathbf{A} := \mathsf{Gen}(\mathsf{seed})$
2:  $\mathbf{\Sigma}_1 := \mathbf{X}_1^T(2^{t_2}\mathbf{Y}_2)$
3:  $\tilde{\mathsf{S}} := \mathsf{Rec}(\mathbf{\Sigma}_1, \mathbf{V}, \mathsf{params})$
4:  $\mathbf{K}' = KDF(\tilde{\mathsf{S}}, \mathsf{c}_1)$
5:  $\bar{M} = \mathsf{Dec}_{\mathbf{K}'}(\mathsf{c}_2)$
6:  $\bar{S} = H(\mathsf{z}, \mathsf{c}_1)$
7:  if $\bar{M} = \perp$ then
8:      $\mathsf{S}' = \bar{S}$
9:  else
10:     $\mathsf{S}' = \tilde{\mathsf{S}}$[a]
11: end if
12: $(\varpi', \mathsf{r}_1') = G(\mathsf{pk}, \mathsf{S}')$
13: $(\mathbf{X}_2', \mathbf{E}_2', \mathbf{E}_\sigma') \leftarrow \mathsf{Sample}(1^\kappa; \mathsf{r}_1')$
14: $\mathbf{Y}_2' := \lfloor (\mathbf{A}^T\mathbf{X}_2' + \mathbf{E}_2')/2^{t_2} \rceil$
15: $\mathbf{\Sigma}_2' := 2^{t_1}\mathbf{Y}_1^T\mathbf{X}_2' + \mathbf{E}_\sigma'$ [b]
16: $\mathbf{V}' \leftarrow \mathsf{Con}(\mathbf{\Sigma}_2', \mathsf{S}', \mathsf{params})$
17: rephrase $\bar{M} = (\omega', M')$ if $\bar{M} \neq \perp$
18: if $(\mathbf{Y}_2 \neq \mathbf{Y}_2' \bigvee \mathbf{V} \neq \mathbf{V}' \bigvee \omega' \neq \varpi' \bigvee \bar{M} = \perp)$ then[c]
19:     return $\perp$
20: else
21:     return $M'$
22: end if
```

[a]This is to hide the performance difference between $\bar{M} = \perp$ and $\bar{M} \neq \perp$.

[b]We are unaware of any vulnerability without performing Step 15-16. If these steps are removed, the condition of $\mathbf{V} = \mathbf{V}'$ is also removed.

[c]The condition whether $\omega' = \varpi'$ can be checked just after Step 12. We refrain from doing so to be against potential side-channel attacks.

9.5 Applications to Privacy-Preserving AKE

In this section, we present a new construction for AKE, referred to as concealed non-malleable key-exchange (CNKE). We present its generic construction, clarify its design rationales, and finally give concrete instantiations from AKCN-MLWE. CNKE is carefully designed, with the following goals: (1) computational efficiency; (2) privacy protection; (3) well compatible with existing standards like TLS1.3; and (4) robust non-malleability.

9.5.1 Abstraction of Key-Exchange and Key-Transport

An ephemeral two-round key-exchange (KE) protocol Π consists of the following algorithms:

- Par: On a security parameter 1^κ, the probabilistic polynomial-time (PPT) procedure Par outputs the parameters params.[14] Denote by $\mathsf{params} \leftarrow \mathsf{Par}(1^\kappa)$.

[14]In reality, Par is implicit or provided by a higher-level protocol, when the KE protocol is used as a building block in a complex system.

- M_I: A PPT procedure used by the initiator player I to generate the first-round message and the associated secret state. Denote by $(\mathcal{M}_I, \mathcal{S}_I) \leftarrow \mathsf{M}_I(\mathsf{params})$, where $\mathcal{M}_I$ is the first-round message to be sent to the responder in plain, while $\mathcal{S}_I$ is some secret state kept by I in private.[15]

- M_R: A PPT procedure used by the responder player R to generate the second-round message, the associated secret state and the shared-key. Denote by $(\mathcal{M}_R, \mathcal{S}_R, K_R) \leftarrow \mathsf{M}_R(\mathsf{params}, \mathcal{M}_I)$, where $\mathcal{M}_R$ is the second-round message to be sent to the initiator in plain, K_R is the derived shared-key, and $\mathcal{S}_R$ is some secret state kept by R in private.[16]

- K_I: A polynomial-time procedure used by the initiator to derive the shared-key. Denote by $K_I = \mathsf{K}_I(\mathsf{params}, \mathsf{M}_R, \mathsf{M}_I, \mathsf{S}_I)$.[17] Without loss of generality, we assume $K_I, K_R \in \{0,1\}^\kappa$.

An ephemeral two-round key-transport (KT) protocol is identical to the above ephemeral KE protocol, except that: $(\mathcal{M}_R, \mathcal{S}_R) \leftarrow \mathsf{M}_R(\mathsf{params}, \mathcal{M}_I, K_R)$, where $\mathcal{M}_R$ is the second-round message to be sent to the initiator in plain, $K_R \leftarrow \{0,1\}^\kappa$ is the shared-key to be transported, and $\mathcal{S}_R$ is some secret state kept by R in private. Specifically, the shared-key K_R is set by the responder.

Definition 9.1. *A KE or KT protocol Π is sound, if it satisfies the following two conditions:*

Completeness *For any sufficiently large security parameter κ, it holds that $K_I = K_R$ with overwhelming probability. The probability is taken over the random coins used in Par, M_I, M_R and K_I.*

Security *Denote by Trans the execution transcript including $(\mathsf{params}, \mathsf{M}_I, \mathsf{M}_R)$. The following two distributions are computationally indistinguishable: $\{\mathsf{Trans}, K_R\}$ and $\{\mathsf{Trans}, \bar{K}\}$, where $\bar{K} \leftarrow \{0,1\}^\kappa$.*

Traditional Diffie-Hellman, and the various KE protocols based on OKCN, are instantiations of sound KE; while the protocols based on AKCN are instantiations of KT.

9.5.2 Basic Construction of CNKE

Let $(\mathsf{KenGen}, \mathcal{E}, \mathcal{D})$ be a CCA-secure PKE scheem,[18] and $\Pi = (\mathsf{Par}, \mathsf{M}_I, \mathsf{M}_R, \mathsf{K}_I)$ be a sound s (ephemeral) KE or KT protocol, where KeyGen and Par can have overlaps. Denote by I_A (resp., I_B) the identity information of the initiator (resp., responder), which consists of information like identity, public key, and certification, etc. Denote by $(\mathsf{PK}_A, \mathsf{SK}_A) \leftarrow \mathsf{KeyGen}(1^\kappa)$ (resp., $(\mathsf{PK}_A, \mathsf{SK}_A) \leftarrow \mathsf{KeyGen}(1^\kappa)$) the public key and secret key of I_A (resp., I_B). We assume the responder's identity information I_B is known to the initiator in advance (e.g, in the client/server setting). For instance, the responder's identity information can be sent to the initiator in the parameter negotiation stage prior to the protocol run. Let $KDF : \{0,1\}^* \to \{0,1\}^\kappa \times \{0,1\}^\kappa$ be a secure key derivation function, and $\mathsf{params} \leftarrow \mathsf{Par}(1^\kappa)$, which are assumed to have been negotiated between the communicating players before the protocol run. The basic version of the privacy-preserving AKE protocol works as follows, which is also depicted in Figure 16:

[15] In the public-key setting, $\mathcal{M}_I$ (resp., $\mathcal{S}_I$) corresponds to the public key (resp., secret key).

[16] Usually, the secret-state $\mathcal{S}_R$ is erased after the shared-key is derived.

[17] The procedure K_I is usually deterministic, but can also be probabilistic in general.

[18] In practice, we prefer to the hybrid construction of combining CCA-secure KEM and authenticated encryption for the CCA-secure PKE.

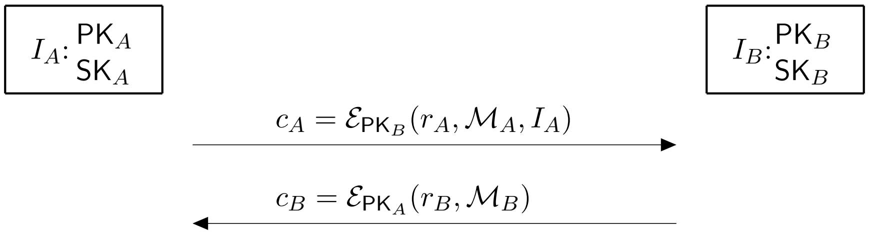
I_A: PK_A SK_A
I_B: PK_B SK_B
$c_A = \mathcal{E}_{\mathsf{PK}_B}(r_A, \mathcal{M}_A, I_A)$
$c_B = \mathcal{E}_{\mathsf{PK}_A}(r_B, \mathcal{M}_B)$

Figure 16: Brief depiction of CNKE, where $(K, K') = KDF(I_A, I_B, r_A, r_B, \mathcal{M}_A, \mathcal{M}_B, K_A, c_A, c_B)$

Round-1: The initiator I_A takes $r_A \leftarrow \{0,1\}^\kappa$, computes $(\mathcal{M}_A, \mathcal{S}_A) \leftarrow \mathsf{M}_I(\mathsf{params})$, and $c_A = \mathcal{E}_{\mathsf{PK}_B}(r_A, \mathcal{M}_A, I_A)$.[19] It sends c_A to I_B, and keeps $(r_A, \mathcal{S}_A)$ in private.

Round-2: The responder I_B computes $(r_A, \mathcal{M}_A, r_A) = \mathcal{D}_{\mathsf{SK}_B}(c_A)$,[20] $(\mathcal{M}_B, \mathcal{S}_B, K_B) \leftarrow \mathsf{M}_R(\mathsf{params}, \mathcal{M}_A)$ if Π is KE (or, takes $K_B \leftarrow \{0,1\}^\kappa$, and computes $(\mathcal{M}_B, \mathcal{S}_B) \leftarrow \mathsf{M}_R(\mathsf{params}, \mathcal{M}_A, K_B)$ if Π is KT), takes $r_B \leftarrow \{0,1\}^\kappa$, and computes $c_B = \mathcal{E}_{\mathsf{PK}_A}(r_A, \mathcal{M}_B, aux_B)$ where aux_B can be an empty string.[21] Then, it computes $(K, K') = KDF(I_A, I_B, r_A, r_B, \mathcal{M}_A, \mathcal{M}_B, K_B, aux_K)$, where K serves as the session-key while K' can be used for mutual authentication within the protocol run, and aux_K is some (possibly empty) auxiliary information.[22] Finally, it sends c_B to I_A, and keeps (K, K') in private but erases all the other secret states.

Initiator key derivation: I_A computes $(r_B, \mathcal{M}_B, aux_B) = \mathcal{D}_{\mathsf{SK}_A}(c_B)$,[23] $K_A = \mathsf{K}_I(\mathsf{params}, \mathcal{M}_B, \mathcal{M}_A, \mathcal{S}_A)$, and $(K, K') = KDF(I_A, I_B, r_A, r_B, \mathcal{M}_A, \mathcal{M}_B, K_A, aux_K)$. The session-key is set to be K.

Some comments about the above basic construction are in order. Firstly, identity privacy is now considered to be an important privacy to be protected, and is mandated in some standards like TLS1.3, EMV, etc. Secondly, by using ephemeral KE or KT protocol, it is computationally more efficient, and is better suitable for implementations by low-power clients. Thirdly, for efficiency considerations, we can only encrypt some parts of each of $\{\mathcal{M}_A, \mathcal{M}_B, I_A\}$, as long as the rest sent in plain does not breach ID-privacy or recover $\mathcal{M}_A$ or $\mathcal{M}_B$ with non-negligible probability. Finally, a more robust variant (but unnecessary for provable security) is to derive the session-key K from $(I_A, I_B, r_A, r_B, \mathcal{M}_A, \mathcal{M}_B, K_A = K_B)$ and the whole session transcript.

The construction, when using OKCN-MLWE as the underlying ephemeral KE, is illustrated in Figure 17. Note that if the underlying CCA-secure PKE is implemented with CCA-secure KEM or PKE from OKCN/AKCN-MLWE, seed can be part of public key or system parameters, and is no need to be encrypted in c_A.

9.5.3 Design Rationale of CNKE, and the Actual Design

We demonstrate the design rationales of CNKE with some concrete attacks, which lead us to the actual design and implementation.

If I_A is not encrypted, and suppose (I_A, I_B) are not put into the input of KDF, there is unknown key share (UKS) attack. The UKS attack works as follows: a man-in-the-middle (MIM)

[19]In actual implementation, some parts of $\mathcal{M}_A$ could be part of public key or system parameters, which are not necessary to be encrypted.
[20]In case the decryption outputs "$\perp$", it aborts.
[21]Depending on the application scenarios, aux_B can include the responder's identity information I_B.
[22]In practice, we may suggest aux_B includes (c_A, c_B).
[23]In case the decryption outputs "$\perp$", it aborts.

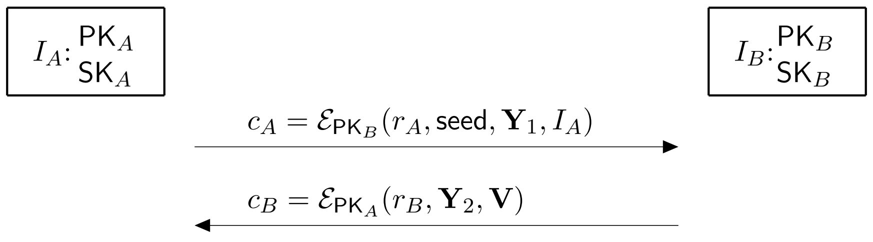
I_A: PK_A SK_A
I_B: PK_B SK_B
$c_A = \mathcal{E}_{\mathsf{PK}_B}(r_A, \mathsf{seed}, \mathbf{Y}_1, I_A)$
$c_B = \mathcal{E}_{\mathsf{PK}_A}(r_B, \mathbf{Y}_2, \mathbf{V})$

Figure 17: Basic Structure of CNKE from OKCN/AKCN-MLWE

adversary I_C intercepts (I_A, c_A), and changes it to (I_C, c_A) that is sent to I_B. After receiving c_B encrypted with the public key of I_C by I_B, it decrypts c_B and re-encrypts the plaintext into c'_B with the public key of I_A. With such an attack, I_A considers it is communicating with I_B, while I_B thinks it is communicating with I_C, but these two (unmatched) sessions are of the same session-key. As users' identity information is not put into the input of KDF for Kyber.AKE [BDK+17], this attack also demonstrates that the design of Kyber.AKE is specific to its CCA-secure KEM mechanism. For implementation generality and security robustness, we may suggest to put users' identity information into the input of KDF for Kyber.AKE, as well as to encrypt client identity information with a symmetric-key encryption.

If M_A or M_B is not encrypted, there could be malleating attacks, as the underlying ephemeral KE/KT protocol is not secure against adaptive attacks. Specifically, by malleating these components, an MIM adversary can cause two *unmatching* sessions to have related ephemeral KE/KT messages. Then, as the sessions are unmatching, the adversary is allowed to expose any secret states of one session to be against another one.

Suppose a MIM adversary $\mathcal{A}$ learns the secret key SK_B of I_B. After intercepting c_A from I_A, it decrypts c_A with SK_B, and then re-encrypts into c'_A with the public key SK_B. Supposing (c_A, c_B) is not put into the input KDF or explicit mutual authentication (e.g., via $MAC_{K'}$) is not added, this attack causes two sessions to have different transcripts but have the same session-key. To be against this type of attacks, we suggest to put (c_A, c_B) into KDF or explicit authenticate the session transcript via $MAC_{K'}$. If the underlying CCA-secure PKE is implemented with the combination of CCA-secure KEM and a CCA-secure symmetric-key encryption scheme SE, we suggest the following method:

- The random coins for the CCA-secure KEM run by one player, as well as the key materials r_A and r_B, are derived from a random string, the public keys, and the transcript, on which the resultant PKE becomes deterministic. The underlying symmetric encryption is recommended to be authenticated encryption, e.g., AES-GCM. But the random coins used for the ephemeral KE/KT protocol should be derived from another independent random string and the transcript (particularly, the public key of the player itself).[24] Moreover, as we shall see, the random strings r_A and r_B can be implicitly transported without being encrypted with the symmetric-key encryption, which further reduces the bandwidth.

Now, suppose the PKE is built by composing CCA-secure KEM and AEAD. Suppose a MIM adversary $\mathcal{A}$ exposes the underlying key for AEAD from the secret state of one player, which can

[24]In general, the random coins for PKE and KE/KT can be derived from the same random seed. But for CCA-secure KEM via the FO-transformation, the random coins used for KEM and those for ephemeral KE/KT should be independent.

be allowed in the underlying security model, it can use the exposed key to change the AEAD ciphertext part, which again causes two sessions to have different transcripts but have related key materials. To be against this type of attack and to provide robust non-malleability, we let the identity information and ephemeral KE/KT component (which are encrypted with AEAD) are put into the input for deriving the random coins for KEM.[25] Note that we have already make the key materials r_A and r_B getting encrypted with AEAD to be related to the KEM part. This way, the KEM part and the AEAD part are non-malleably combined, which provides robust resistance to MIM attacks and to secrecy exposure.

The basic construction lacks perfect forward security (PFS). To add PFS and add explicit mutual authentication, I_B additionally sends $\tau_B = MAC_{K'}(0)$ in the second round, and I_A sends $\tau_A = MAC_{K'}(1)$ in an extra third round. In this case, we suggest aux_K includes (c_A, c_B).[26] In order to be compatible with TLS1.3, in the actual implementation we prefer to use the Finish mechanism of TLS1.3. Based on [Z16], the analysis of CNKE will be given in a separate work soon.

9.5.4 Instantiation of CNKE from AKCN-MLWE

Let $G : \{0,1\}^* \rightarrow \{0,1\}^\kappa \times \{0,1\}^{p(\kappa)}$, $G_A : \{0,1\}^* \rightarrow \{0,1\}^{p_A(\kappa)}$, $G_B : \{0,1\}^* \rightarrow \{0,1\}^\kappa \times \{0,1\}^{p_B(\kappa)}$ and $H : \{0,1\}^* \rightarrow \{0,1\}^\kappa$ be cryptographic hash functions (or any secure key derivation functions), where $p(\cdot)$, $p_A(\cdot)$ and $p_B(\cdot)$ are positive polynomials. Let $KDF : \{0,1\}^* \rightarrow \mathcal{K}_{\mathsf{ae}}$ be a key derivation function, where $\mathcal{K}_{\mathsf{ae}}$ is the key space of the underlying AEAD scheme. Let $\mathsf{SE} = (\mathsf{K}_{se}, \mathsf{Enc}, \mathsf{Dec})$ be an AEAD scheme, as specified in Section 2.1. The public and secret keys of each player of I_A and I_B are the same as in Algorithm 32 in Section 9.4. Denote by $\mathsf{pk}_A = (\mathsf{seed}, \mathbf{Y}_A)$ and $\mathsf{sk}_A = \{\mathbf{X}_A, \mathsf{z}_A, \mathsf{pk}_A\}$ the public and secret keys of the initiator player I_A, and by $\mathsf{pk}_B = (\mathsf{seed}, \mathbf{Y}_B)$ and $\mathsf{sk}_B = \{\mathbf{X}_B, \mathsf{z}_B, \mathsf{pk}_B\}$ the public and secret key of responder player I_B. For simplicity and symmetry, we assume both players, as well as the underlying ephemeral AKCN/OKCN-MLWE, use the same parameters params and the same matrix $\mathbf{A}$ derived from seed. We also abuse the notation Sample for generating noises of varied length. Also, for simplicity, we assume that for each player the same number of least significant bits are cut off from both the static public key and the ephemeral MLWE-samples: t_A (resp., t_B) for I_A (resp., I_B).

[25] Unlike the FO-transformation proposed in [FO13, D02], it is no need to put all the plaintexts into the input for deriving random coins of KEM. With this approach, we are unaware of meaningful attacks even if $\mathcal{M}_A$ and/or $\mathcal{M}_B$ are sent in plain.

[26] Alternatively, $\tau_B = MAC_{K'}(c_B, c_A, tr_B)$, $\tau_A = MAC_{K'}(tr_B, c_A, c_B)$ or $\tau_A = MAC_{K'}(tr_B, c_A, c_B, \tau_B)$, etc, where tr_B is an empty string or the negotiation transcript prior to session run (e.g., for forwarding I_B to I_A or for negotiating parameters). We remark that, if the CCA-secure PKE is built via the combination of CCA-secure KEM and an AEAD scheme, the explicit MAC mechanism can be waived. In these cases, aux_K can be empty.

Algorithm 35 Round-1 run by I_A

```
1:  $\mathbf{A} := \mathsf{Gen}(\mathsf{seed})$
2:  $\mathsf{s}_A^{kt} \leftarrow \{0,1\}^\kappa$
3:  $\mathsf{r}_A^{kt} = G_A(\mathsf{s}_A^{kt}, I_A)$
4:  $(\mathbf{X}_A^{kt}, \mathbf{E}_A^{kt}) \leftarrow \mathsf{Sample}(1^\kappa; \mathsf{r}_A^{kt})$
5:  $\mathbf{Y}_A^{kt} := \lfloor (\mathbf{A}\mathbf{X}_A^{kt} + \mathbf{E}_A^{kt})/2^{t_A} \rceil$
6:  $\mathsf{s}_A^{kem} \leftarrow \{0,1\}^\kappa$
7:  $(\mathsf{r}_A, \mathsf{r}_A^{kem}) = G(\mathsf{s}_A^{kem}, I_B, \mathbf{Y}_A^{kt})$
8:  $(\mathbf{X}_A^{kem}, \mathbf{E}_A^{kem}, \mathbf{E}_{(\sigma,A)}^{kem}) \leftarrow \mathsf{Sample}(1^\kappa; \mathsf{r}_A^{kem})$
9:  $\mathbf{Y}_A^{kem} := \lfloor (\mathbf{A}^T\mathbf{X}_A^{kem} + \mathbf{E}_A^{kem})/2^{t_A} \rceil$
10: $\mathbf{\Sigma}_A^{kem} := 2^{t_B}\mathbf{Y}_B^T\mathbf{X}_A^{kem} + \mathbf{E}_{(\sigma,A)}^{kem}$
11: $\mathbf{V}_A^{kem} \leftarrow \mathsf{Con}(\mathbf{\Sigma}_A^{kem}, \mathsf{s}_A^{kem}, \mathsf{params})$
12: $\mathbf{K}_A^{ae} = KDF(\mathsf{s}_A^{kem}, \mathsf{c}_A^{kem})$, where $\mathsf{c}_A^{kem} = (\mathbf{Y}_A^{kem}, \mathbf{V}_A^{kem})$
13: $\mathsf{c}_A^{ae} = \mathsf{Enc}_{\mathbf{K}_A^{ae}}(\mathsf{H}_A, \mathsf{r}_A || I_A || \mathbf{Y}_A^{kt})$[a]
14: **Send** $\mathsf{c}_A := (\mathsf{c}_A^{kem}, \mathsf{c}_A^{ae})$ to $I_B$
```

[a]The associated data H_A depends on the application scenarios, which can contain a (possibly empty) subset of $\{\mathsf{c}_A^{kem}, I_B\}$ and the transcript up to now.

Algorithm 36 Round-2 run by I_B upon receiving $\mathsf{c}_A = (\mathsf{c}_A^{kem}, \mathsf{c}_A^{ae})$

```
1: $\mathbf{A} := \mathsf{Gen}(\mathsf{seed})$
2: $\mathsf{s}_B^{kt} \leftarrow \{0,1\}^\kappa$
3: $(\mathsf{k}_B^{kt}, \mathsf{r}_B^{kt}) = G_B(\mathsf{s}_B^{kt}, I_B)$
4: $(\mathbf{X}_B^{kt}, \mathbf{E}_B^{kt}, \mathbf{E}_{(\sigma,B)}^{kt}) \leftarrow \mathsf{Sample}(1^\kappa; \mathsf{r}_B^{kt})$
5: $\mathbf{Y}_B^{kt} := \lfloor (\mathbf{A}^T\mathbf{X}_B^{kt} + \mathbf{E}_B^{kt})/2^{t_B} \rceil$
6: $\mathbf{\Sigma}_B^{kt} := 2^{t_A}(\mathbf{Y}_A^{kt})^T\mathbf{X}_B^{kt} + \mathbf{E}_{(\sigma,B)}^{kt}$
7: $\mathbf{V}_B^{kt} \leftarrow \mathsf{Con}(\mathbf{\Sigma}_B^{kt}, \mathsf{k}_B^{kt}, \mathsf{params})$
8: $\mathbf{\Sigma}_B^{kem} := \mathbf{X}_B^T(2^{t_A}\mathbf{Y}_A^{kem})$
9: $\mathsf{s}_A'^{kem} := \mathsf{Rec}(\mathbf{\Sigma}_B^{kem}, \mathbf{V}_A^{kem}, \mathsf{params})$
10: $\mathbf{K}_A'^{ae} = KDF(\mathsf{s}_A'^{kem}, \mathsf{c}_A^{kem})$
11: $\bar{M}_A = \mathsf{Dec}_{\mathbf{K}_A'^{ae}}(\mathsf{c}_A^{ae})$
12: if $\bar{M}_A \neq \perp$ then
13:     rephrase $\bar{M}_A = (\mathsf{r}'_A, I_A, \mathbf{Y}_A'^{kt})$
14:     $(\mathsf{r}''_A, \mathsf{r}_A'^{kem}) = G(\mathsf{s}_A'^{kem}, I_B, \mathbf{Y}_A'^{kt})$[a]
15: else
16:     $(\mathsf{r}''_A, \mathsf{r}_A'^{kem}) = G(\mathsf{z}_B, \mathsf{c}_A)$[b]
17: end if
18: $(\mathbf{X}_A'^{kem}, \mathbf{E}_A'^{kem}, \mathbf{E}_{(\sigma,A)}'^{kem}) \leftarrow \mathsf{Sample}(1^\kappa; \mathsf{r}_A'^{kem})$
19: $\mathbf{Y}_A'^{kem} := \lfloor (\mathbf{A}^T\mathbf{X}_A'^{kem} + \mathbf{E}_A'^{kem})/2^{t_A} \rceil$
20: $\mathbf{\Sigma}_A'^{kem} := 2^{t_B}\mathbf{Y}_B^T\mathbf{X}_A'^{kem} + \mathbf{E}_{(\sigma,A)}'^{kem}$
21: $\mathbf{V}_A'^{kem} \leftarrow \mathsf{Con}(\mathbf{\Sigma}_A'^{kem}, \mathsf{s}_A'^{kem}, \mathsf{params})$
22: if $(\mathbf{Y}_A'^{kem} \neq \mathbf{Y}_A^{kem} \bigvee \mathbf{V}_A'^{kem} \neq \mathbf{V}_A^{kem} \bigvee \mathsf{r}'_A \neq \mathsf{r}''_A \bigvee \bar{M}_A = \perp)$ then
23:     return $\perp$
24: end if
25: $\mathsf{s}_B^{kem} \leftarrow \{0,1\}^\kappa$
26: $(\mathsf{r}_B, \mathsf{r}_B^{kem}) = G(\mathsf{s}_B^{kem}, I_A, \mathbf{Y}_B^{kt}, \mathbf{V}_B^{kt})$
27: $(\mathbf{X}_B^{kem}, \mathbf{E}_B^{kem}, \mathbf{E}_{(\sigma,B)}^{kem}) \leftarrow \mathsf{Sample}(1^\kappa; \mathsf{r}_B^{kem})$
28: $\mathbf{Y}_B^{kem} := \lfloor (\mathbf{A}^T\mathbf{X}_B^{kem} + \mathbf{E}_B^{kem})/2^{t_B} \rceil$
29: $\mathbf{\Sigma}_B'^{kem} := 2^{t_A}\mathbf{Y}_A^T\mathbf{X}_B^{kem} + \mathbf{E}_{(\sigma,B)}^{kem}$
30: $\mathbf{V}_B^{kem} \leftarrow \mathsf{Con}(\mathbf{\Sigma}_B'^{kem}, \mathsf{s}_B^{kem}, \mathsf{params})$
31: $\mathbf{K}_B^{ae} = KDF(\mathsf{s}_B^{kem}, \mathsf{c}_B^{kem})$, where $\mathsf{c}_B^{kem} = (\mathbf{Y}_B^{kem}, \mathbf{V}_B^{kem})$
32: $\mathsf{c}_B^{ae} = \mathsf{Enc}_{\mathbf{K}_B^{ae}}(\mathsf{H}_B, r_B||\mathbf{Y}_B^{kt}||\mathbf{V}_B^{kt})$
33: $\mathbf{K}_B = KDF(\mathsf{r}'_A, \mathsf{r}_B, \mathsf{k}_B^{kt}, \mathbf{Y}_A'^{kt}, \mathbf{Y}_B^{kt}, \mathbf{V}_B^{kt}, I_A, I_B)$
34: $\mathsf{c}_B^f = \mathsf{Enc}_{\mathbf{K}_B}(\mathsf{H}_B^f, \mathsf{Finish}_B)$, where $\mathsf{Finish}_B = H(\mathsf{c}_A, \mathsf{c}_B, trs)$[c]
35: Send $\{\mathsf{c}_B, \mathsf{c}_B^f\}$ to $I_A$, where $\mathsf{c}_B = (\mathsf{c}_B^{kem}, \mathsf{c}_B^{ae})$
```

[a]In actual implementation, $(\mathsf{s}_A'^{kem}, I_B, \mathbf{Y}_A'^{kt})$ and $(\mathsf{z}_B, \mathsf{c}_A)$ may be padded into the same size. In case $\bar{M}_A = \perp$ or $\mathsf{r}'_A \neq \mathsf{r}''_A$, we can simply abort here. We refrain from doing so to be against potential side-channel attacks.

[b]This step is to hide the performance difference between $\bar{M}_A = \perp$ and $\bar{M}_A \neq \perp$, in order to be against potential side-channel attacks.

[c]Specifically, finish_B is the hash of the transcript up to now, where trs includes transcript determined from the context, e.g., parameter negotiation transcript, players' identity and IP-address information, etc.

Algorithm 37 Round-3 run by I_A upon receiving $(\mathsf{c}_B, \mathsf{c}_B^f)$

```
1:  $\mathbf{\Sigma}_A'^{kem} := \mathbf{X}_A^T(2^{t_B}\mathbf{Y}_B^{kem})$
2:  $\mathsf{s}_B'^{kem} := \mathsf{Rec}(\mathbf{\Sigma}_A'^{kem}, \mathbf{V}_B^{kem}, \mathsf{params})$
3:  $\mathbf{K}_B'^{ae} = KDF(\mathsf{s}_B'^{kem}, \mathsf{c}_B^{kem})$
4:  $\bar{M}_B = \mathsf{Dec}_{\mathbf{K}_B'^{ae}}(\mathsf{c}_B^{ae})$
5:  if $\bar{M}_B \neq \perp$ then
6:      rephrase $\bar{M}_B = (\mathsf{r}_B', \mathbf{Y}_B'^{kt}, \mathbf{V}_B'^{kt})$
7:      $(\mathsf{r}_B'', \mathsf{r}_B'^{kem}) = G(\mathsf{s}_B'^{kem}, I_A, \mathbf{Y}_B'^{kt}, \mathbf{V}_B'^{kt})$
8:  else
9:      $(\mathsf{r}_B'', \mathsf{r}_B'^{kem}) = G(\mathsf{z}_A, \mathsf{c}_B)$
10: end if
11: $(\mathbf{X}_B'^{kem}, \mathbf{E}_B'^{kem}, \mathbf{E}_{(\sigma,B)}'^{kem}) \leftarrow \mathsf{Sample}(1^\kappa; \mathsf{r}_B'^{kem})$
12: $\mathbf{Y}_B'^{kem} := \lfloor(\mathbf{A}^T\mathbf{X}_B'^{kem} + \mathbf{E}_B'^{kem})/2^{t_B}\rceil$
13: $\mathbf{\Sigma}_B'^{kem} := 2^{t_A}\mathbf{Y}_A^T\mathbf{X}_B'^{kem} + \mathbf{E}_{(\sigma,B)}'^{kem}$
14: $\mathbf{V}_B'^{kem} \leftarrow \mathsf{Con}(\mathbf{\Sigma}_B'^{kem}, \mathsf{s}_B'^{kem}, \mathsf{params})$
15: if $(\mathbf{Y}_B'^{kem} \neq \mathbf{Y}_B^{kem} \bigvee \mathbf{V}_B'^{kem} \neq \mathbf{V}_B^{kem} \bigvee \mathsf{r}_B' \neq \mathsf{r}_B'' \bigvee \bar{M}_B = \perp)$ then
16:     return $\perp$
17: end if
18: $\mathbf{\Sigma}_B'^{kt} := (\mathbf{X}_A^{kt})^T(2^{t_B}\mathbf{Y}_B'^{kt})$
19: $\mathsf{k}_B'^{kt} := \mathsf{Rec}(\mathbf{\Sigma}_B'^{kt}, \mathbf{V}_B'^{kt}, \mathsf{params})$
20: $\mathbf{K}_A = KDF(\mathsf{r}_A, \mathsf{r}_B', \mathsf{k}_B'^{kt}, \mathbf{Y}_A^{kt}, \mathbf{Y}_B'^{kt}, \mathbf{V}_B'^{kt}, I_A, I_B)$
21: $\mathsf{Finish}_B' = \mathsf{Dec}_{\mathbf{K}_A}(\mathsf{c}_B^f)$
22: if $\mathsf{Finish}_B'$ is incorrect then
23:     return $\perp$
24: end if
25: $\mathsf{Finish}_A = H(\mathsf{c}_A, \mathsf{c}_B, \mathsf{c}_B^f, trs)$
26: $\mathsf{c}_A^f = \mathsf{Enc}_{\mathbf{K}_A}(\mathsf{H}_A^f, \mathsf{Finish}_A || m_A)$, where $m_A \in \{0,1\}^*$ is the application data
27: Send $\mathsf{c}_A^f$ to $I_B$
```

The values r_A and r_B play multiple roles: (1) serving as the key materials in session-key derivation; (2) non-malleably attaching the KEM ciphertexts to the AEAD ciphertexts; and (3) being resistant to adversary in the quantum random oracle model. In the FO-transformation variants [TU16, HHK17], these values are sent in plain, while get encrypted with AEAD in our construction. Note that the concrete instantiation of CNKE from AKCN-MLWE is actually a secure channel establishment protocol, where the session-key $\mathbf{K}_A = \mathbf{K}_B$ has already been used within the session run. The CNKE protocol is carefully designed to have the following advantages:

- By using ephemeral AKCN-MLWE for transporting k_B^{kt}, it is computationally more efficient, and is more applicable to client/server setting with low-power clients.
- Robust resistance to MIM malleating attacks, to secrecy exposure, and to side-channel attacks.

- Privacy protection. Identity privacy is deemed to be an important privacy issue, and is mandated by some prominent standards like TLS1.3, EMV, etc. Concealing the components of the ephemeral KT protocol not only strengthens security, but is also useful for privacy protection.
- Well compatible with TLS1.3, by explicitly using AEAD (that is mandated by TLS1.3) and using the Finish mechanism of TLS1.3 for mutual authentications.

Acknowledgement. We are grateful to Leixiao Cheng, Yuan Li and Qin Luo for great assistance, and to Sauvik Bhattacharya, Jintai Ding, Vadim Lyubashevsky, Chris Peikert, Peter Schwabe, Moti Yung and Andrew C. Yao for helpful discussions.

References

[AGKS05] M. Abe, R. Gennaro, K. Kurosawa and V. Shoup. Tag-KEM/DEM: A New Framework for Hybrid Encryption and A New Analysis of Kurosawa-Desmedt KEM. *EUROCRYPT 2005:* 128-146.

[A17] M. R. Albrecht. On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL. *EUROCRYPT 2017:* 103-129.

[APS15] M. R. Albrecht, R. Player and S. Scott. On the Concrete Hardness of Learning with Errors. *Journal of Mathematical Cryptology*, Volume 9, Issue 3, pages 169-203, 2015.

[ADPS16] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe. Post-quantum Key Exchange — A New Hope. *25th USENIX Security Symposium (USENIX Security 16)*, pages 327–343.

[ADPS16b] E. Alkim, L. Ducas, T. Pppelmann, and P. Schwabe. NewHope without Reconciliation. *Cryptology ePrint Archive*, Report 2016/1157, 2016.

[AJS16] E. Alkim, P. Jakubeit, and P. Schwabe. A New Hope on ARM Cortex-M. *Cryptology ePrint Archive*, Report 2016/758, 2016.

[ACPS09] B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems. *CRYPTO 2009*: 595-618.

[BLL$^+$15] S. Bai, A. Langlois, T. Lepoint, D. Stehlé, and R. Steinfeld. Improved Security Proofs in Lattice-Based Cryptography: Using the Rényi Divergence rather than the Statistical Distance. *ASIACRYPT 2015*: 3-24.

[BPR12] A. Banerjee and C. Peikert and A. Rosen. Pseudorandom Functions and Lattices. *EUROCRYPT 2012*: 719-737.

[BGM$^+$16] A. Bogdanov, S. Guo, D. Masny, S. Richelson, and A. Rosen. On the Hardness of Learning with Rounding over Small Modulus. *TCC 2016*: 209-224.

[BCD$^+$16] J. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. Stebila. Frodo: Take off the Ring! Practical, Quantum-Secure Key Exchange from LWE. *ACM CCS 2016*: 1006-1018.

[BCNS15] J.W. Bos, C. Costello, M. Naehrig, and D. Stebila. Post-Quantum Key Exchange for the TLS Protocol from the Ring Learning with Errors Problem. *IEEE Symposium on Security and Privacy 2015*, pages 553-570.

[BDK$^+$17] J. W. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, D. Stehlé. CRYSTALS-Kyber: a CCA-Secure Module-lattice-based KEM. *Cryptology ePrint Archive*, Report 2017/634.

[CN11] Y. Chen and P.Q. Nguyen. BKZ 2.0: Better Lattice Security Estimates. *ASIACRYPT 2011*: 1-20.

[CKLS16] J.H. Cheon, D. Kim, J. Lee, and Y. Song. Lizard: Cut Off the Tail! Practical Post-Quantum Public-Key Encryption from LWE and LWR. *Cryptology ePrint Archive*, Report 2016/1126, 2016.

[CW90] D. Coppersmith and S. Winograd. Matrix Multiplication via Arithmetic Progressions. *Journal of Symbolic Computation*, volume 9, issue 3, pages 251-280, 1990.

[CS03] R. Cramer and V. Shoup. Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. *SIAM Journal on Computing*, 33(1): 167226, 2003.

[D02] A. W. Dent. A Designers Guide to KEMs. *Cryptology ePrint Archive*, Report 2002/174, 2002.

[JD12] J. Ding, X. Xie and X. Lin. A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem. *Cryptology ePrint Archive*, Report 2012/688, 2012.

[DORS08] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. *SIAM Journal on Computing*, volume 38, issue 1, pages 97-139, 2008.

[DD12] L. Ducas and A. Durmus. Ring-LWE in Polynomial Rings. *PKC 2012*: 34-51.

[DTV15] A. Duc, F. Tramèr, and S. Vaudenay. Better Algorithms for LWE and LWR. *EUROCRYPT 2015*: 173-202.

[FO99] E. Fujisaki and T. Okamoto. How to Enhance the Security of Public-Key Encryption at Minimum Cost. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* Volume 83, Issue 1, pages 24-32, 1999.

[FO13] E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. *Journal of Cryptology*, Volume 26, Issue 1, pages 80-101, 2013.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for Hard Lattices and New Cryptographic Constructions. *ACM STOC 2008*: 197-206.

[GS16] S. Gueron and F. Schlieker. Speeding Up R-LWE Post-Quantum Key Exchange. *Cryptology ePrint Archive*, Report 2016/467, 2016.

[H14] D. Harvey. Faster Arithmetic for Number-Theoretic Transforms. *Journal of Symbolic Computation*, 60: 113-119, 2014.

[HHK17] D. Hofheinz, K. Hövelmanns, and Eike Kiltz. A Modular Analysis of the Fujisaki-Okamoto Transformation. *Cryptology ePrint Archive*, Report 2017/604.

[KLL15] M. Kaplan, G. Leurent, A. Leverrier and M. Naya-Plasencia. Quantum Differential and Linear Cryptanalysis. *ArXiv Preprint*: 1510.05836, 2015.

[Kra03] H. Krawczyk. *SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols CRYPTO 2003*: 400-425.

[KPW13] H. Krawczyk, K.G. Paterson and H. Wee. On the Security of the TLS Protocol: A Systematic Analysis. *CRYPTO* 2013: 429-448.

[KM10] H. Kuwakado and M. Morii. Quantum Distinguisher between the 3-round Feistel Cipher and the Random Permutation. *IEEE ISIT 2010*: 2682-2685.

[LS15] A. Langlois and D. Stehlé. Worst-case to Average-case Reductions for Module Lattices. Des. Codes Cryptography, 75(3): 565-599, 2015.

[LP11] R. Lindner and C. Peikert. Better Key Sizes (and Attacks) for LWE-Based Encryption. *CT-RSA 2011*: 319-339.

[LMPR08] V. Lyubashevsky, D. Micciancio, C. Peikert, and A. Rosen. SWIFFT: A Mmodest-Proposal for FFT Hashing., *FSE* 2008: 54-72.

[LPR10] V. Lyubashevsky, C. Peikert, and O. Regev. On Ideal Lattices and Learning with Errors over Rings. *EUROCRYPT 2010*: 1-23.

[LPR13b] V. Lyubashevsky, C. Peikert, and O. Regev. A Toolkit for Ring-LWE Cryptography. *EUROCRYPT 2013*: 35-54

[M85] P. Montgomery. Modular Multiplication Without Trial Division. *Mathematics of Computation*, vol. 44, 519521, 1985.

[PRS11] K. G. Paterson, T. Ristenpart, and T. Shrimpton. Tag Size Does Matter: Attacks and Proofs for the TLS Record Protocol. *ASIACRYPT* 2011: 372-389.

[Pei09] C. Peikert. Public-Key Cryptosystems from the Worst-Case Shortest Vector Problem. *STOC 2009*: 333-342.

[Pei14] C. Peikert. Lattice Cryptography for the Internet. *PQCrypto 2014*: 197-219.

[Pei16] C. Peikert. A Decade of Lattice Cryptography. In *Foundations and Trends in Theoretical Computer Science*, Volume 10, Issue 4, pages 283-424, 2016.

[PVW08] C. Peikert, V. Vaikuntanathan, and B. Waters. A Framework for Efficient and Composable Oblivious Transfer. *CRYPTO 2008*: 554-571.

[Pop16] A.V. Poppelen, Cryptographic Decoding of the Leech Lattice. *Cryptology ePrint Archive*, Report 2016/1050, 2016.

[PG13] T. Pöppelmann and T. Güneysu. Towards Practical Lattice-Based Public-Key Encryption on Reconfigurable Hardware. *SAC 2013*: 68-85.

[Reg09] O. Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. *Journal of the ACM (JACM)*, Volume 56, Issue 6, pages 34, 2009.

[Res] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3.

[R02] P. Rogaway. Authenticated-Encryption with Associated-Data. *ACM CCS* 2002: 98-107.

[RS06] P. Rogaway and T. Shrimpton. A Provable-Security Treatment of the Key-Wrap Problem. *EUROCRYPT* 2006: 373-390.

[SE94] C. P. Schnorr and M. Euchner. Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems. *Mathematical Programming*, Volume 66, Issue 2, pages 181-199, Springer, 1994.

[Sim02] M.K. Simon. Probability Distributions Involving Gaussian Random Variables : A Handbook for Engineers and Scientists. Springer, 2012.

[SM16] D. Stebila and M. Mosca. Post-Quantum Key Exchange for the Internet and the Open Quantum Safe Project. *Cryptology ePrint Archive*, Report 2016/1017, 2016.

[Str69] V. Strassen. Gaussian Elimination is not Optimal. *Numerische Mathematik*, Volume 13, Issue 4, pages 354-356, Springer, 1969.

[TU16] E. E. Targhi and D. Unruh. Post-Quantum Security of the Fujisaki-Okamoto and OAEP Transforms. *TCC* 2016-B: 192-216.

[Z16] Y. Zhao. Identity-Concealed Authenticated Encryption and Key Exchange. *ACM-CCS 2016*: 1464-1479.

Algorithm 38 Key consensus scheme in Frodo

```
1: procedure Con($\sigma_1$, params)                         ▷ $\sigma_1 \in [0, q)$
2:     $v = \left\lfloor 2^{-\bar{B}+1}\sigma_1 \right\rceil \bmod 2$
3:     $k_1 = \left\lfloor 2^{-\bar{B}}\sigma_1 \right\rceil \bmod 2^B$
4:     return $(k_1, v)$
5: end procedure
6: procedure Rec($\sigma_2, v$, params)                      ▷ $\sigma_2 \in [0, q)$
7:     find $x \in \mathbb{Z}_q$ closest to $\sigma_2$ s.t. $\left\lfloor 2^{-\bar{B}+1}x \right\rceil \bmod 2 = v$
8:     $k_2 = \left\lfloor 2^{-\bar{B}}x \right\rceil \bmod 2^B$
9:     return $k_2$
10: end procedure
```

A Consensus Mechanism of Frodo

Let the modulo q be power of 2, which can be generalized to arbitrary modulo using the techniques in [Pei14]. Let integer B be a power of 2. $B < (\log q) - 1, \bar{B} = (\log q) - B$ (note that $m = 2^B$ in our notations). The underlying KC mechanism implicitly in Frodo is presented in Figure 38.

Claim A.1 ([BCD+16], Claim 3.2). *If $|\sigma_1 - \sigma_2|_q < 2^{\bar{B}-2}$, then $\mathsf{Rec}(\sigma_2, v) = k_1$. i.e. the scheme in Algorithm 38 is correct.*

This claim is equivalence to require $4md < q$.

B Consensus Mechanism of NewHope

Note that, for the consensus mechanism of NewHope, the *rec* procedure is run both in Con and in Rec, and a random bit b is used in Con corresponding to the dbl trick in [Pei14].

C Proof of Corollary 3.2

Proof. For correctness, supposing $|\sigma_1 - \sigma_2|_q \leq d$, by Fact 3.1, there exist $\theta \in \mathbb{Z}$ and $\delta \in [-d, d]$ such that $\sigma_2 = \sigma_1 + \theta q + \delta$. Taking this into line 8 of Algorithm 3, i.e., the formula computing k_2, we have

$$\begin{aligned} k_2 &= \lfloor (\sigma_1 - v + \theta q + \delta)/g \rceil \bmod m \\ &= (k_1 + \theta m + \lfloor \delta/g \rceil) \bmod m. \end{aligned}$$

If $2md < q$, then $|\delta/g| \leq d/g < 1/2$, so that $k_2 = k_1 \bmod m = k_1$.

For security, as a special case of generic scheme described in Algorithm 1, the security of Algorithm 3 follows directly from that of Algorithm 1. □ □

D On KC/AKC vs. Fuzzy Extractor

Our formulations of KC and AKC are abstractions of the core ingredients of previous constructions of KE and PKE from LWE/RLWE. As we shall see in the subsequent sections, the design

Algorithm 39 NewHope Consensus Mechanism

```
procedure $\textsc{Decode}(\mathbf{x} \in \mathbb{R}^4/\mathbb{Z}^4)$        ▷ Return a bit $k$ such that $k\mathbf{g}$ is closest to $\mathbf{x} + \mathbb{Z}^4$
    $\mathbf{v} = \mathbf{x} - \lfloor \mathbf{x} \rceil$
    return $k = 0$ if $\|\mathbf{v}\|_1 \leq 1$, and 1 otherwise
end procedure

$\mathsf{HelpRec}(\mathbf{x}, b) = \mathsf{CVP}_{\tilde{D}_4}\left(\frac{2^r}{q}(\mathbf{x} + b\mathbf{g})\right) \bmod 2^r$        ▷ $b$ corresponds to the dbl trick [Pei14]
$rec\left(\mathbf{x} \in \mathbb{Z}_q^4, \mathbf{v} \in \mathbb{Z}_{2^r}^4\right) = \mathsf{Decode}\left(\frac{1}{q}\mathbf{x} - \frac{1}{2^r}\mathbf{B}\mathbf{v}\right)$

procedure $\textsc{Con}(\boldsymbol{\sigma}_1 \in \mathbb{Z}_q^4, \mathsf{params})$
    $b \leftarrow \{0, 1\}$
    $\mathbf{v} \leftarrow \mathsf{HelpRec}(\boldsymbol{\sigma}_1, b)$
    $k_1 \leftarrow rec(\boldsymbol{\sigma}_1, \mathbf{v})$
    return $(k_1, \mathbf{v})$
end procedure

procedure $\textsc{Rec}(\boldsymbol{\sigma}_2 \in \mathbb{Z}_q^4, \mathbf{v} \in \mathbb{Z}_{2^r}^4, \mathsf{params})$
    $k_2 \leftarrow rec(\boldsymbol{\sigma}_2, \mathbf{v})$
end procedure

```

and analysis of KE and PKE from LWE, LWR and RLWE can be reduced to KC and AKC. We also note that KC and AKC are similar to fuzzy extractor proposed in [DORS08], which extracts shared-keys from biometrics and noisy data. In this section, we make some discussions on the relationship between KC/AKC and fuzzy extractor.

The differences between the definitions of KC/AKC and that of fuzzy extractor lie mainly in the following ways. Firstly, AKC was not considered within the definitional framework of fuzzy extractor. Secondly, the metric $|\cdot|_q$ we use in defining KC and AKC was not considered for fuzzy extractor. Thirdly, in the definitions of KC and AKC, the algorithm Rec (corresponding Rep for fuzzy extractor) is mandated to be *deterministic*, while in the formulation of fuzzy extractor it is probabilistic. Fourthly, in the formulation of fuzzy extractor [DORS08], w, R and P (corresponding σ_1, k and v in KC/AKC) are binary strings; while in the definitions of KC/AKC, the corresponding values $\sigma_1 \in \mathbb{Z}_q$, $k \in \mathbb{Z}_m$ and $v \in \mathbb{Z}_g$ have more structured ranges, which are helpful in deriving the exact upper bound. Finally, for the security of KC and AKC, we require that the signal value v be independent of the shared-key k_1 (that can be subject to arbitrary distribution for AKC); roughly speaking, in the definition of fuzzy extractor [DORS08], it is required that the joint distribution (R, P) be statistically close to (U_l, P) where $R \in \{0,1\}^l$ and U_l is the uniform distribution over $\{0,1\}^l$.

A generic upper bound on the length of key extracted by fuzzy extractor is proposed in [DORS08, Appendix C]. In comparison, the upper bounds for KC and AKC proved in this work are more versatile and precise w.r.t. the metric $|\cdot|_q$. For example, the effect of the length of the signal v, i.e., the bandwidth parameter g, is not considered in the upper bound for fuzzy extractor, but is taken into account in the upper bounds for KC and AKC.

A generic construction of fuzzy extractor from *secure sketch*, together with a generic construction of *secure sketch* for *transitive metric spaces*, is proposed in [DORS08]. We note that

$(\mathbb{Z}_q, |\cdot|_q)$ can be naturally seen as a *transitive matric space*. Compared to the secure sketch based generic constructions of fuzzy extractor, our constructions of KC and AKC are direct and more efficient.

In spite of some similarities between KC/AKC and fuzzy extractors, we remark that before our this work the relation between fuzzy extractor and KE from LWE and its variants is actually opaque. Explicitly identifying and formalizing KC/AKC and reducing lattice-based cryptosystems to KC/AKC in a *black-box* modular way, with inherent bounds on what could or couldn't be done, cut the complexity of future design and analysis of these cryptosystems.

E Overview of the Primal and Dual Attacks

This section is almost verbatim from [ADPS16]. The dual attack tries to distinguish the distribution of LWE samples and the uniform distribution. Suppose $(\mathbf{A}, \mathbf{b} = \mathbf{A}\mathbf{s}+\mathbf{e}) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$ is a LWE sample, where $\mathbf{s}$ and $\mathbf{e}$ are drawn from discrete Gaussian of variance σ_s^2 and σ_e^2 respectively. Then we choose a positive real $c \in \mathbb{R}, 0 < c \leq q$, and construct $L_c(\mathbf{A}) = \{(\mathbf{x}, \mathbf{y}/c) \in \mathbb{Z}^m \times (\mathbb{Z}/c)^n \mid \mathbf{x}^T\mathbf{A} = \mathbf{y}^T \mod q\}$, which is a lattice with dimension $m+n$ and determinant $(q/c)^n$. For a short vector $(\mathbf{x}, \mathbf{y}) \in L_c(\mathbf{A})$ found by the BKZ algorithm, we have $\mathbf{x}^T\mathbf{b} = \mathbf{x}^T(\mathbf{A}\mathbf{s}+\mathbf{e}) = c \cdot \mathbf{y}^T\mathbf{s} + \mathbf{x}^T\mathbf{e} \mod q$. If $(\mathbf{A}, \mathbf{b})$ is an LWE sample, the distribution of the right-hand side will be very close to a Gaussian of standard deviation $\sqrt{c^2\|\mathbf{y}\|^2\sigma_s^2 + \|\mathbf{x}\|^2\sigma_e^2}$, otherwise the distribution will be uniform. $\|(\mathbf{x}, \mathbf{y})\|$ is about $\delta_0^{m+n}(q/c)^{\frac{n}{m+n}}$, where δ_0 is the root Hermite factor. We heuristically assume that $\|\mathbf{x}\| = \sqrt{\frac{m}{m+n}}\,\|(\mathbf{x}, \mathbf{y})\|$, and $\|\mathbf{y}\| = \sqrt{\frac{n}{m+n}}\,\|(\mathbf{x}, \mathbf{y})\|$. Then we can choose $c = \sigma_e/\sigma_s$ that minimizes the standard deviation of $\mathbf{x}^T\mathbf{b}$. The advantage of distinguishing $\mathbf{x}^T\mathbf{b}$ from uniform distribution is $\varepsilon = 4\exp(-2\pi^2\tau^2)$, where $\tau = \sqrt{c^2\|\mathbf{y}\|^2\sigma_s^2 + \|\mathbf{x}\|^2\sigma_e^2}/q$. This attack must be repeated $R = \max\{1, 1/(2^{0.2075b}\varepsilon^2)\}$ times to be successful.

The primal attack reduces the LWE problem to the unique-SVP problem. Let $\Lambda_w(\mathbf{A}) = \{(\mathbf{x}, \mathbf{y}, z) \in \mathbb{Z}^n \times (\mathbb{Z}^m/w) \times \mathbb{Z} \mid \mathbf{A}\mathbf{x} + w\mathbf{y} = z\mathbf{b} \mod q\}$, and a vector $\mathbf{v} = (\mathbf{s}, \mathbf{e}/w, 1) \in \Lambda_w(\mathbf{A})$. $\Lambda_w(\mathbf{A})$ is a lattice of $d = m+n+1$ dimensions, and its determinant is $(q/w)^m$. From geometry series assumption, we can derive $\|\mathbf{b}_i^*\| \approx \delta_0^{d-2i-1}\det(\Lambda_w(\mathbf{A}))^{1/d}$. We heuristically assume that the length of projection of $\mathbf{v}$ onto the vector space spanned by the last b Gram-Schmidt vectors is about $\sqrt{\frac{b}{d}}\,\|(\mathbf{s}, \mathbf{e}/w, 1)\| \approx \sqrt{\frac{b}{d}\left(n\sigma_s^2 + m\sigma_e^2/w^2 + 1\right)}$. If this length is shorter than $\|\mathbf{b}_{d-b}^*\|$, this attack can be successful. Hence, the successful condition is $\sqrt{\frac{b}{d}\left(n\sigma_s^2 + m\sigma_e^2/w^2 + 1\right)} \leq \delta_0^{2b-d-1}\left(\frac{q}{w}\right)^{m/d}$. We know that the optimal w balancing the secret $\mathbf{s}$ and the noise $\mathbf{e}$ is about σ_e/σ_s.

F Security Estimation of the Parameters of Frodo

Scheme	Attack	Rounded Gaussian					Post-reduction		
		m'	b	C	Q	P	C	Q	P
Classical	Primal	549	442	138	126	100	**132**	120	95
	Dual	544	438	136	124	99	**130**	119	94
Recommended	Primal	716	489	151	138	110	145	**132**	104
	Dual	737	485	150	137	109	144	**130**	103
Paranoid	Primal	793	581	179	163	129	178	162	**129**
	Dual	833	576	177	161	128	177	161	**128**

Table 21: Security estimation of the parameters proposed for Frodo in [BCD+16], as specified in Table 11.

G Security Analysis of LWE-Based Key Exchange

Definition G.1. *A KC or AKC based key exchange protocol from LWE is* secure*, if for any sufficiently large security parameter λ and any PT adversary $\mathcal{A}$, $\left|\Pr[b' = b] - \frac{1}{2}\right|$ is negligible, as defined w.r.t. game G_0 specified in Algorithm 40.*

Algorithm 40 Game G_0

```
1: $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times n}$
2: $\mathbf{X}_1, \mathbf{E}_1 \leftarrow \chi^{n\times l_A}$
3: $\mathbf{Y}_1 = \mathbf{A}\mathbf{X}_1 + \mathbf{E}_1$
4: $\mathbf{X}_2, \mathbf{E}_2 \leftarrow \chi^{n\times l_B}$
5: $\mathbf{Y}_2 = \mathbf{A}^T\mathbf{X}_2 + \mathbf{E}_2$
6: $\mathbf{E}_\sigma \leftarrow \chi^{l_A\times l_B}$
7: $\mathbf{\Sigma}_2 = \mathbf{Y}_1^T\mathbf{X}_2 + \mathbf{E}_\sigma$
8: $\left(\mathbf{K}_2^0, \mathbf{V}\right) \leftarrow \mathsf{Con}(\mathbf{\Sigma}_2, \mathsf{params})$
9: $\mathbf{K}_2^1 \leftarrow \mathbb{Z}_m^{l_A\times l_B}$
10: $b \leftarrow \{0,1\}$
11: $b' \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{Y}_1, \lfloor \mathbf{Y}_2/2^t \rfloor, \mathbf{K}_2^b, \mathbf{V})$
```

Before starting to prove the security, we first recall some basic properties of the LWE assumption. The following lemma is derived by a direct hybrid argument [PVW08, BCD+16].

Lemma G.1 (LWE in the matrix form)**.** *For positive integer parameters $(\lambda, n, q \geq 2, l, t)$, where n, q, l, t all are polynomial in λ, and a distribution χ over $\mathbb{Z}_q$, denote by $L_\chi^{(l,t)}$ the distribution over $\mathbb{Z}_q^{t\times n} \times \mathbb{Z}_q^{t\times l}$ generated by taking $\mathbf{A} \leftarrow \mathbb{Z}_q^{t\times n}, \mathbf{S} \leftarrow \chi^{n\times l}, \mathbf{E} \leftarrow \chi^{t\times l}$ and outputting $(\mathbf{A}, \mathbf{AS} + \mathbf{E})$. Then, under the standard LWE assumption on indistinguishability between $A_{q,\mathbf{s},\chi}$ (with $\mathbf{s} \leftarrow \chi^n$) and $\mathcal{U}(\mathbb{Z}_q^n \times \mathbb{Z}_q)$, no PT distinguisher $\mathcal{D}$ can distinguish, with non-negligible probability, between the distribution $L_\chi^{(l,t)}$ and $\mathcal{U}(\mathbb{Z}_q^{t\times n} \times \mathbb{Z}_q^{t\times l})$ for sufficiently large λ.*

Theorem G.1. *If $(\mathsf{params}, \mathsf{Con}, \mathsf{Rec})$ is a* correct *and* secure *KC or AKC scheme, the key exchange protocol described in Figure 7 is* secure *under the (matrix form of) LWE assumption.*

Proof. The proof is similar to, but actually simpler than, that in [Pei14, BCD+16]. The general idea is that we construct a sequence of games: G_0, G_1 and G_2, where G_0 is the original game for defining security. In every move from game G_i to G_{i+1}, $0 \leq i \leq 1$, we change a little. All games G_i's share the same PT adversary $\mathcal{A}$, whose goal is to distinguish between the matrices chosen uniformly at random and the matrices generated in the actual key exchange protocol. Denote by T_i, $0 \leq i \leq 2$, the event that $b = b'$ in Game G_i. Our goal is to prove that $\Pr[T_0] < 1/2 + negl$, where $negl$ is a negligible function in λ. For ease of readability, we re-produce game G_0 below. For presentation simplicity, in the subsequent analysis, we always assume the underlying KC or AKC is correct. The proof can be trivially extended to the case that correctness holds with overwhelming probability (i.e., failure occurs with negligible probability).

Algorithm 41 Game G_0

```
1: $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times n}$
2: $\mathbf{X}_1, \mathbf{E}_1 \leftarrow \chi^{n\times l_A}$
3: $\mathbf{Y}_1 = \mathbf{A}\mathbf{X}_1 + \mathbf{E}_1$
4: $\mathbf{X}_2, \mathbf{E}_2 \leftarrow \chi^{n\times l_B}$
5: $\mathbf{Y}_2 = \mathbf{A}^T\mathbf{X}_2 + \mathbf{E}_2$
6: $\mathbf{E}_\sigma \leftarrow \chi^{l_A\times l_B}$
7: $\mathbf{\Sigma}_2 = \mathbf{Y}_1^T\mathbf{X}_2 + \mathbf{E}_\sigma$
8: $(\mathbf{K}_2^0, \mathbf{V}) \leftarrow \mathsf{Con}(\mathbf{\Sigma}_2, \mathsf{params})$
9: $\mathbf{K}_2^1 \leftarrow \mathbb{Z}_m^{l_A\times l_B}$
10: $b \leftarrow \{0,1\}$
11: $b' \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{Y}_1, \lfloor \mathbf{Y}_2/2^t \rfloor, \mathbf{K}_2^b, \mathbf{V})$
```

Algorithm 42 Game G_1

```
1: $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times n}$
2: $\mathbf{X}_1, \mathbf{E}_1 \leftarrow \chi^{n\times l_A}$
3: $\mathbf{Y}_1 \leftarrow \mathbb{Z}_q^{n\times l_A}$
4: $\mathbf{X}_2, \mathbf{E}_2 \leftarrow \chi^{n\times l_B}$
5: $\mathbf{Y}_2 = \mathbf{A}^T\mathbf{X}_2 + \mathbf{E}_2$
6: $\mathbf{E}_\sigma \leftarrow \chi^{l_A\times l_B}$
7: $\mathbf{\Sigma}_2 = \mathbf{Y}_1^T\mathbf{X}_2 + \mathbf{E}_\sigma$
8: $(\mathbf{K}_2^0, \mathbf{V}) \leftarrow \mathsf{Con}(\mathbf{\Sigma}_2, \mathsf{params})$
9: $\mathbf{K}_2^1 \leftarrow \mathbb{Z}_m^{l_A\times l_B}$
10: $b \leftarrow \{0,1\}$
11: $b' \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{Y}_1, \lfloor \mathbf{Y}_2/2^t \rfloor, \mathbf{K}_2^b, \mathbf{V})$
```

Lemma G.2. *$|\Pr[T_0] - \Pr[T_1]| < negl$, under the indistinguishability between $L_\chi^{(l_A,n)}$ and $\mathcal{U}(\mathbb{Z}_q^{n\times n} \times \mathbb{Z}_q^{n\times l_A})$.*

Proof. Construct a distinguisher $\mathcal{D}$, in Algorithm 43, who tries to distinguish $L_\chi^{(l_A,n)}$ from $\mathcal{U}(\mathbb{Z}_q^{n\times n} \times \mathbb{Z}_q^{n\times l_A})$.

Algorithm 43 Distinguisher $\mathcal{D}$

```
1: procedure $\mathcal{D}(\mathbf{A}, \mathbf{B})$          ▷ $\mathbf{A} \in \mathbb{Z}_q^{n\times n}, \mathbf{B} \in \mathbb{Z}_q^{n\times l_A}$
2:     $\mathbf{Y}_1 = \mathbf{B}$
3:     $\mathbf{X}_2, \mathbf{E}_2 \leftarrow \chi^{n\times l_B}$
4:     $\mathbf{Y}_2 = \mathbf{A}^T\mathbf{X}_2 + \mathbf{E}_2$
5:     $\mathbf{E}_\sigma \leftarrow \chi^{l_A\times l_B}$
6:     $\mathbf{\Sigma}_2 = \mathbf{Y}_1^T\mathbf{X}_2 + \mathbf{E}_\sigma$
7:     $(\mathbf{K}_2^0, \mathbf{V}) \leftarrow \mathsf{Con}(\mathbf{\Sigma}_2, \mathsf{params})$
8:     $\mathbf{K}_2^1 \leftarrow \mathbb{Z}_m^{l_A\times l_B}$
9:     $b \leftarrow \{0,1\}$
10:    $b' \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{Y}_1, \lfloor \mathbf{Y}_2/2^t \rfloor, \mathbf{K}_2^b, \mathbf{V})$
11:    if $b' = b$ then
12:        return 1
13:    else
14:        return 0
15:    end if
16: end procedure
```

If $(\mathbf{A}, \mathbf{B})$ is subject to $L_\chi^{(l_A,n)}$, then $\mathcal{D}$ perfectly simulates G_0. Hence, $\Pr\left[\mathcal{D}\left(L_\chi^{(l_A,n)}\right) = 1\right] = \Pr[T_0]$. On the other hand, if $(\mathbf{A}, \mathbf{B})$ is chosen uniformly at random from $\mathbb{Z}_q^{n\times n} \times \mathbb{Z}_q^{n\times l_A}$, which are denoted as $(\mathbf{A}^{\mathcal{U}}, \mathbf{B}^{\mathcal{U}})$, then $\mathcal{D}$ perfectly simulates G_1. So, $\Pr[\mathcal{D}(\mathbf{A}^{\mathcal{U}}, \mathbf{B}^{\mathcal{U}}) = 1] = \Pr[T_1]$. Hence, $|\Pr[T_0] - \Pr[T_1]| = \left|\Pr[\mathcal{D}(L_\chi^{(l_A,n)}) = 1] - \Pr[\mathcal{D}(\mathbf{A}^{\mathcal{U}}, \mathbf{B}^{\mathcal{U}}) = 1]\right| < negl$. □ □

Algorithm 44 Game G_1

```
1: $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times n}$
2: $\mathbf{X}_1, \mathbf{E}_1 \leftarrow \chi^{n\times l_A}$
3: $\mathbf{Y}_1 \leftarrow \mathbb{Z}_q^{n\times l_A}$
4: $\mathbf{X}_2, \mathbf{E}_2 \leftarrow \chi^{n\times l_B}$
5: $\mathbf{Y}_2 = \mathbf{A}^T\mathbf{X}_2 + \mathbf{E}_2$
6: $\mathbf{E}_\sigma \leftarrow \chi^{l_A\times l_B}$
7: $\mathbf{\Sigma}_2 = \mathbf{Y}_1^T\mathbf{X}_2 + \mathbf{E}_\sigma$
8: $(\mathbf{K}_2^0, \mathbf{V}) \leftarrow \mathsf{Con}(\mathbf{\Sigma}_2, \mathsf{params})$
9: $\mathbf{K}_2^1 \leftarrow \mathbb{Z}_m^{l_A\times l_B}$
10: $b \leftarrow \{0,1\}$
11: $b' \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{Y}_1, \lfloor \mathbf{Y}_2/2^t \rfloor, \mathbf{K}_2^b, \mathbf{V})$
```

Algorithm 45 Game G_2

```
1: $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times n}$
2: $\mathbf{X}_1, \mathbf{E}_1 \leftarrow \chi^{n\times l_A}$
3: $\mathbf{Y}_1 \leftarrow \mathbb{Z}_q^{n\times l_A}$
4: $\mathbf{X}_2, \mathbf{E}_2 \leftarrow \chi^{n\times l_B}$
5: $\mathbf{Y}_2 \leftarrow \mathbb{Z}_q^{n\times l_B}$
6: $\mathbf{E}_\sigma \leftarrow \chi^{l_A\times l_B}$
7: $\mathbf{\Sigma}_2 \leftarrow \mathbb{Z}_q^{l_A\times l_B}$
8: $(\mathbf{K}_2^0, \mathbf{V}) \leftarrow \mathsf{Con}(\mathbf{\Sigma}_2, \mathsf{params})$
9: $\mathbf{K}_2^1 \leftarrow \mathbb{Z}_m^{l_A\times l_B}$
10: $b \leftarrow \{0,1\}$
11: $b' \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{Y}_1, \lfloor \mathbf{Y}_2/2^t \rfloor, \mathbf{K}_2^b, \mathbf{V})$
```

Lemma G.3. $|\Pr[T_1] - \Pr[T_2]| < negl$, *under the indistinguishability between* $L_\chi^{(l_B,n+l_A)}$ *and* $\mathcal{U}(\mathbb{Z}_q^{(n+l_A)\times n} \times \mathbb{Z}_q^{(n+l_A)\times l_B})$.

Proof. As $\mathbf{Y}_1$ is subject to uniform distribution in G_1, $(\mathbf{Y}_1^T, \mathbf{\Sigma}_2)$ can be regarded as an $L_\chi^{(l_B,l_A)}$ sample of secret $\mathbf{X}_2$ and noise $\mathbf{E}_\sigma$. Based on this observation, we construct the following distinguisher $\mathcal{D}'$.

Algorithm 46 Distinguisher $\mathcal{D}'$

```
1: procedure $\mathcal{D}'(\mathbf{A}', \mathbf{B})$ where $\mathbf{A}' \in \mathbb{Z}_q^{(n+l_A)\times n}, \mathbf{B} \in \mathbb{Z}_q^{(n+l_A)\times l_B}$
2:     Denote $\mathbf{A}' = \begin{pmatrix} \mathbf{A}^T \\ \mathbf{Y}_1^T \end{pmatrix}$        ▷ $\mathbf{A} \in \mathbb{Z}_q^{n\times n}, \mathbf{Y}_1^T \in \mathbb{Z}_q^{l_A\times n}$
3:     Denote $\mathbf{B} = \begin{pmatrix} \mathbf{Y}_2 \\ \mathbf{\Sigma}_2 \end{pmatrix}$        ▷ $\mathbf{Y}_2 \in \mathbb{Z}_q^{n\times l_B}, \mathbf{\Sigma}_2 \in \mathbb{Z}_q^{l_A\times l_B}$
4:     $(\mathbf{K}_2^0, \mathbf{V}) \leftarrow \mathsf{Con}(\Sigma_2, \mathsf{params})$
5:     $\mathbf{K}_2^1 \leftarrow \mathbb{Z}_m^{l_A\times l_B}$
6:     $b \leftarrow \{0,1\}$
7:     $b' \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{Y}_1, \lfloor \mathbf{Y}_2/2^t \rfloor, \mathbf{K}_2^b, \mathbf{V})$
8:     if $b' = b$ then
9:         return 1
10:    else
11:        return 0
12:    end if
13: end procedure
```

If $(\mathbf{A}', \mathbf{B})$ is subject to $L_\chi^{(l_B,n+l_A)}$, $\mathbf{A}' \leftarrow \mathbb{Z}_q^{(n+l_A)\times n}$ corresponds to $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times n}$ and $\mathbf{Y}_1 \leftarrow \mathbb{Z}_q^{n\times l_A}$ in G_1; and $\mathbf{S} \leftarrow \chi^{n\times l_B}$ (resp., $\mathbf{E} \leftarrow \chi^{(n+l_A)\times l_B}$) in generating $(\mathbf{A}', \mathbf{B})$ corresponds to

$\mathbf{X}_2 \leftarrow \chi^{n\times l_B}$ (resp., $\mathbf{E}_2 \leftarrow \chi^{n\times l_B}$ and $\mathbf{E}_\sigma \leftarrow \chi^{l_A\times l_B}$) in G_1. In this case, we have

$$\begin{aligned}\mathbf{B} = \mathbf{A}'\mathbf{S} + \mathbf{E} &= \begin{pmatrix} \mathbf{A}^T \\ \mathbf{Y}_1^T \end{pmatrix} \mathbf{X}_2 + \begin{pmatrix} \mathbf{E}_2 \\ \mathbf{E}_\sigma \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{A}^T\mathbf{X}_2 + \mathbf{E}_2 \\ \mathbf{Y}_1^T\mathbf{X}_2 + \mathbf{E}_\sigma \end{pmatrix} = \begin{pmatrix} \mathbf{Y}_2 \\ \mathbf{\Sigma}_2 \end{pmatrix}\end{aligned}$$

Hence $\Pr\left[\mathcal{D}'\left(L_\chi^{(l_B,n+l_A)}\right) = 1\right] = \Pr[T_1]$.

On the other hand, if $(\mathbf{A}', \mathbf{B})$ is subject to uniform distribution $\mathcal{U}(\mathbb{Z}_q^{(n+l_A)\times n} \times \mathbb{Z}_q^{(n+l_A)\times l_B})$, then $\mathbf{A}, \mathbf{Y}_1, \mathbf{Y}_2, \mathbf{\Sigma}_2$ all are also uniformly random; So, the view of $\mathcal{D}'$ in this case is the same as that in game G_2. Hence, $\Pr[\mathcal{D}'(\mathbf{A}', \mathbf{B}) = 1] = \Pr[T_2]$ in this case. Then $|\Pr[T_1] - \Pr[T_2]| = |\Pr[\mathcal{D}'(L_\chi^{(l_B,n+l_A)}) = 1] - \Pr[\mathcal{D}'(\mathcal{U}(\mathbb{Z}_q^{(n+l_A)\times n} \times \mathbb{Z}_q^{(n+l_A)\times l_B})) = 1]| < negl.$ □ □

Lemma G.4. *If the underlying KC or AKC is* secure, $\Pr[T2] = \frac{1}{2}$.

Proof. Note that, in Game G_2, for any $1 \le i \le l_A$ and $1 \le j \le l_B$, $\left(\mathbf{K}_2^0[i,j], \mathbf{V}[i,j]\right)$ only depends on $\mathbf{\Sigma}_2[i,j]$, and $\mathbf{\Sigma}_2$ is subject to uniform distribution. By the *security* of KC, we have that, for each pair (i,j), $\mathbf{K}_2^0[i,j]$ and $\mathbf{V}[i,j]$ are independent, and $\mathbf{K}_2^0[i,j]$ is uniform distributed. Hence, $\mathbf{K}_2^0$ and $\mathbf{V}$ are independent, and $\mathbf{K}_2^0$ is uniformly distributed, which implies that $\Pr[T_2] = 1/2$.
□ □

This finishes the proof of Theorem G.1. □ □

H Construction and Analysis of AKCN-4:1

H.1 Overview of NewHope

By extending the technique of [PG13], in NewHope the coefficients of $\boldsymbol{\sigma}_1$ (i.e., the polynomial of degree n) are divided into $n/4$ groups, where each group contains four coordinates. On the input of four coordinates, only one bit (rather than four bits) consensus is reached, which reduces the error rate to about 2^{-61} which is viewed to be negligible in practice.

Specifically, suppose Alice and Bob have $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ in $\mathbb{Z}_q^4$ respectively, and they are close to each other. One can regard the two vectors as elements in $\mathbb{R}^4/\mathbb{Z}^4$, by treating them as $\frac{1}{q}\boldsymbol{\sigma}_1$ and $\frac{1}{q}\boldsymbol{\sigma}_2$. Consider the matrix $\mathbf{B} = (\mathbf{u}_0, \mathbf{u}_1, \mathbf{u}_2, \mathbf{g}) \in \mathbb{R}^{4\times 4}$, where $\mathbf{u}_i$, $0 \le i \le 2$, is the canonical unit vector whose i-th coordinate is 1, and $\mathbf{g} = (1/2, 1/2, 1/2, 1/2)^T$. Denote by $\tilde{D}_4$ the lattice generated by $\mathbf{B}$. Note that $\mathbb{Z}^4 \subset \tilde{D}_4 \subset \mathbb{R}^4$. Denote by $\mathcal{V}$ the close Voronoi cell of the origin in $\tilde{D}_4$. In fact, $\mathcal{V}$ is the intersection of the unit ball in norm 1 and the unit ball in infinity norm (the reader is referred to NewHope [ADPS16, Appendix C] for details). The following procedure $\mathsf{CVP}_{\tilde{D}_4}(\mathbf{x})$ returns the vector $\mathbf{v}$ such that $\mathbf{Bv}$ is closest to $\mathbf{x}$, i.e., $\mathbf{x} \in \mathbf{Bv} + \mathcal{V}$, where the distance is measured in the Euclidean norm.

Algorithm 47 CVP$_{\tilde{D}_4}$ in NewHope [ADPS16]

```
1: procedure CVP$_{\tilde{D}_4}(\mathbf{x} \in \mathbb{R}^4)$
2:     $\mathbf{v}_0 = \lfloor \mathbf{x} \rceil$
3:     $\mathbf{v}_1 = \lfloor \mathbf{x} - \mathbf{g} \rceil$
4:     $k = 0$ if $\|\mathbf{x} - \mathbf{v}_0\|_1 < 1$ and 1 otherwise
5:     $(v_0, v_1, v_2, v_3)^T = \mathbf{v}_k$
6:     return $\mathbf{v} = (v_0, v_1, v_2, k)^T + v_3 \cdot (-1, -1, -1, 2)^T$
7: end procedure
```

If $\boldsymbol{\sigma}_1$ is in the Voronoi cell of $\mathbf{g}$, then the consensus bit is set to be 1, and 0 otherwise. Hence, Alice finds the closest lattice vector of $\boldsymbol{\sigma}_1$ by running the $\mathsf{CVP}_{\tilde{D}_4}$ procedure described in Algorithm 47, and calculates their difference which is set to be the hint signal $\mathbf{v}$. Upon receiving $\mathbf{v}$, Bob subtracts the difference from $\boldsymbol{\sigma}_2$. Since $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ are very close, the subtraction moves $\frac{1}{q}\boldsymbol{\sigma}_2$ towards a lattice point in $\tilde{D}_4$. Then Bob checks whether or not the point after the move is in the Voronoi cell of $\boldsymbol{g}$, and so the consensus is reached. Furthermore, to save bandwidth, NewHope chooses an integer r, and discretizes the Voronoi cell of $\boldsymbol{g}$ to 2^{4r} blocks, so that only $4r$ bits are needed to transfer the hint information. To make the distribution of consensus bit uniform, NewHope adds a small noise to $\boldsymbol{\sigma}_1$, similar to the dbl trick used in [Pei14]. The Con and Rec procedures, distilled from NewHope, are presented in Algorithm 39 in Appendix B.

H.2 Construction and Analysis of AKCN-4:1

For any integer q and vector $\mathbf{x} = (x_0, x_1, x_2, x_3)^T \in \mathbb{Z}_q^4$, denote by $\|\mathbf{x}\|_{q,1}$ the sum $|x_0|_q + |x_1|_q + |x_2|_q + |x_3|_q$. For two vectors $\mathbf{a} = (a_0, a_1, a_2, a_3)^T, \mathbf{b} = (b_0, b_1, b_2, b_3)^T \in \mathbb{Z}^4$, let $\mathbf{a} \bmod \mathbf{b}$ denote the vector $(a_0 \bmod b_0, a_1 \bmod b_1, a_2 \bmod b_2, a_3 \bmod b_3)^T \in \mathbb{Z}^4$. The scheme of AKCN-4:1 is presented in Algorithm 48.

Compared with the consensus mechanism of NewHope presented in Appendix B, AKCN-4:1 can be simpler and computationally more efficient. In specific, the uniformly random bit b used in NewHope (corresponding the dbl trick in [Pei14]) is eliminated with AKCN-4:1, which saves 256 (resp., 1024) random bits in total when reaching 256 (resp., 1024) consensus bits. In addition, as k_1, as well as $k_1(q+1)\mathbf{g}$, can be opaline computed and used (e.g., for encryption, in parallel with the protocol run), AKCN-4:1 enjoys online/offline speeding-up and parallel computing.

Theorem H.1. *If $\|\boldsymbol{\sigma}_1 - \boldsymbol{\sigma}_2\|_{q,1} < q\left(1 - \frac{1}{g}\right) - 2$, then the AKCN-4:1 scheme depicted in Algorithm 48 is* correct.

Proof. Suppose $\mathbf{v}' = \mathsf{CVP}_{\tilde{D}_4}(g(\boldsymbol{\sigma}_1 + k_1(q+1)\mathbf{g})/q)$. Then, $\mathbf{v} = \mathbf{v}' \bmod (g, g, g, 2g)$, and so there exits $\boldsymbol{\theta} = (\theta_0, \theta_1, \theta_2, \theta_3) \in \mathbb{Z}^4$ such that $\mathbf{v} = \mathbf{v}' + g(\theta_0, \theta_1, \theta_2, 2\theta_3)^T$. From the formula calculating $\mathbf{v}'$, we know there exits $\boldsymbol{\varepsilon} \in \mathcal{V}$, such that $g(\boldsymbol{\sigma}_1 + k_1(q+1)\mathbf{g})/q = \boldsymbol{\varepsilon} + \mathbf{B}\mathbf{v}'$. Hence, $\mathbf{B}\mathbf{v}' = g(\boldsymbol{\sigma}_1 + k_1(q+1)\mathbf{g})/q - \boldsymbol{\varepsilon}$.

From the formula computing $\mathbf{x}$ in Rec, we have $\mathbf{x} = \mathbf{B}\mathbf{v}/g - \boldsymbol{\sigma}_2/q = \mathbf{B}\mathbf{v}'/g - \boldsymbol{\sigma}_2/q + \mathbf{B}(\theta_0, \theta_1, \theta_2, 2\theta_3)^T = k_1\mathbf{g} + k_1\mathbf{g}/q - \boldsymbol{\varepsilon}/g + (\boldsymbol{\sigma}_1 - \boldsymbol{\sigma}_2)/q + \mathbf{B}(\theta_0, \theta_1, \theta_2, 2\theta_3)^T$. Note that the last term $\mathbf{B}(\theta_0, \theta_1, \theta_2, 2\theta_3)^T \in \mathbb{Z}^4$, and in line 7 of Algorithm 48 we subtract $\lfloor \mathbf{x} \rceil \in \mathbb{Z}^4$ from $\mathbf{x}$, so the difference between $\mathbf{x} - \lfloor \mathbf{x} \rceil$ and $k_1\mathbf{g}$ in norm 1 is no more than $2/q + 1/g + \|\boldsymbol{\sigma}_1 - \boldsymbol{\sigma}_2\|_{q,1}/q < 1$. Hence, $k_2 = k_1$. □ □

Algorithm 48 AKCN-4:1

```
1: procedure Con($\boldsymbol{\sigma}_1 \in \mathbb{Z}_q^4, k_1 \in \{0,1\}$, params)
2:     $\mathbf{v} = \mathsf{CVP}_{\tilde{D}_4}(g(\boldsymbol{\sigma}_1 + k_1(q+1)\mathbf{g})/q) \bmod (g,g,g,2g)^T$
3:     return v
4: end procedure
5: procedure Rec($\boldsymbol{\sigma}_2 \in \mathbb{Z}_q^4, \boldsymbol{v} \in \mathbb{Z}_g^3 \times \mathbb{Z}_{2g}$, params)
6:     $\mathbf{x} = \mathbf{B}\mathbf{v}/g - \boldsymbol{\sigma}_2/q$
7:     return $k_2 = 0$ if $\|\mathbf{x} - \lfloor \mathbf{x} \rceil\|_1 < 1$, 1 otherwise.
8: end procedure
```

Theorem H.2. *AKCN-4:1 depicted in Algorithm 48 is* secure. *Specifically, if $\boldsymbol{\sigma}_1$ is subject to uniform distribution over $\mathbb{Z}_q^4$, then $\mathbf{v}$ and k_1 are independent.*

Proof. Let $\mathbf{y} = (\boldsymbol{\sigma}_1 + k_1(q+1)\mathbf{g}) \bmod q \in \mathbb{Z}_q^4$. First we prove that $\mathbf{y}$ is independent of k_1, when $\boldsymbol{\sigma}_1 \leftarrow \mathbb{Z}_q^4$. Specifically, for arbitrary $\tilde{\mathbf{y}} \in \mathbb{Z}_q^4$ and arbitrary $\tilde{k}_1 \in \{0,1\}$, we want to prove that $\Pr[\mathbf{y} = \tilde{\mathbf{y}} \mid k_1 = \tilde{k}_1] = \Pr[\boldsymbol{\sigma}_1 = (\tilde{\mathbf{y}} - k_1(q+1)\mathbf{g}) \bmod q \mid k_1 = \tilde{k}_1] = 1/q^4$. Hence, $\mathbf{y}$ and k_1 are independent.

For simplicity, denote by $\mathbf{G}$ the vector $(g,g,g,2g)$. Map $\phi : \mathbb{Z}^4 \to \mathbb{Z}_g^3 \times \mathbb{Z}_{2g}$ is defined by $\phi(\mathbf{w}) = \mathsf{CVP}_{\tilde{D}_4}(g\mathbf{w}/q) \bmod \mathbf{G}$. We shall prove that, for any $\boldsymbol{\theta} \in \mathbb{Z}^4$, $\phi(\mathbf{w} + q\boldsymbol{\theta}) = \phi(\mathbf{w})$. By definition of ϕ, $\phi(\mathbf{w} + q\boldsymbol{\theta}) = \mathsf{CVP}_{\tilde{D}_4}(g\mathbf{w}/q + g\boldsymbol{\theta}) \bmod \mathbf{G}$. Taking $\mathbf{x} = g\mathbf{w}/q + g\boldsymbol{\theta}$ into Algorithm 47, we have $\mathsf{CVP}_{\tilde{D}_4}(g\mathbf{w}/q + g\boldsymbol{\theta}) = \mathsf{CVP}_{\tilde{D}_4}(g\mathbf{w}/q) + \mathbf{B}^{-1}(g\boldsymbol{\theta})$. It is easy to check that the last term $\mathbf{B}^{-1}(g\boldsymbol{\theta})$ always satisfies $\mathbf{B}^{-1}(g\boldsymbol{\theta}) \bmod \mathbf{G} = 0$.

From the above property of ϕ, we have $\phi(\mathbf{y}) = \phi((\boldsymbol{\sigma}_1 + k_1(q+1)\mathbf{g}) \bmod q) = \phi(\boldsymbol{\sigma}_1 + k_1(q+1)\mathbf{g}) = \mathbf{v}$. As k_1 is independent of $\mathbf{y}$, and $\mathbf{v}$ only depends on $\mathbf{y}$, k_1 and $\mathbf{v}$ are independent. □ □

I Implementing $\mathbf{Hx}^T$ in SEC with Simple Bit Operations

```c
uint16_t getCode(uint16_t x)
{
    uint16_t c, p;
    c = (x >> 4) ^ x;
    c = (c >> 2) ^ c;
    p = ((c >> 1) ^ c) & 1;
    x = (x >> 8) ^ x;
    c = (x >> 2) ^ x;
    p = (((c >> 1) ^ c) & 1) | (p << 1);
    x = (x >> 4) ^ x;
    p = (((x >> 1) ^ x) & 1) | (p << 1);
    x = (x >> 2) ^ x;
    p = (x & 1) | (p << 1);

    return p;
}
```

Listing 1: An implementation of $\mathbf{Hx}^T$ with C language

J CCA-Secure KEM from OKCN-MLWE

The transformation from AKCN-MLWE to CCA-secure KEM is specified in detail in [BDK+17]. Here, we present the CCA-secure KEM from OKCN-MLWE, which is instantiated from [HHK17].

For schemes based on MLWE, LWE and LWR, the security parameter κ is set to be 256.[27] Let $G : \{0,1\}^* \to \{0,1\}^\kappa \times \{0,1\}^\kappa \times \{0,1\}^{p_1(\kappa)} \times \{0,1\}^{p_2(\kappa)}$, where p_1 and p_2 are positive polynomials, and $H : \{0,1\}^* \to \{0,1\}^\kappa$ be two cryptographic hash functions (or any secure key derivation function). We write $(\mathbf{X}_2, \mathbf{E}_2, \mathbf{E}_\sigma) \leftarrow \mathsf{Sample}(1^\kappa; \mathsf{r}_1)$ to denote the process of sampling the noises: $\mathbf{X}_2, \mathbf{E}_2 \leftarrow S_\eta^{l\times 1}$ and $\mathbf{E}_\sigma \leftarrow S_\eta$, using randomness $\mathsf{r}_1 \in \{0,1\}^{p_1(\kappa)}$. Denote by $\mathsf{Con}(\mathbf{\Sigma}_2, \mathsf{params}; \mathsf{r}_2)$ the process of running $\mathsf{Con}(\mathbf{\Sigma}_2, \mathsf{params})$ with randomness $\mathsf{r}_2 \in \{0,1\}^{p_2(\kappa)}$.

Algorithm 49 $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$

```
1: $\mathsf{z} \leftarrow \{0,1\}^\kappa$
2: $\mathsf{seed} \leftarrow \{0,1\}^\kappa$
3: $\mathbf{A} := \mathsf{Gen}(\mathsf{seed})$
4: $\mathbf{X}_1, \mathbf{E}_1 \leftarrow S_\eta^{l\times 1}$
5: $\mathbf{Y}_1 := \lfloor (\mathbf{A}\mathbf{X}_1 + \mathbf{E}_1)/2^{t_1} \rceil$
6: return $(\mathsf{pk} := (\mathsf{seed}, \mathbf{Y}_1), \mathsf{sk} := \{\mathbf{X}_1, \mathsf{z}, \mathsf{pk}\})$
```

Algorithm 50 $(\mathsf{ct}, \mathsf{key}) \leftarrow \mathsf{Encaps}(\mathsf{pk})$

```
1: $\mathsf{S} \leftarrow \{0,1\}^\kappa$
2: $(\mathsf{k}, \varpi, \mathsf{r}_1, \mathsf{r}_2) = G(\mathsf{pk}, \mathsf{S})$
3: $(\mathbf{X}_2, \mathbf{E}_2, \mathbf{E}_\sigma) \leftarrow \mathsf{Sample}(1^\kappa; \mathsf{r}_1)$
4: $\mathbf{A} := \mathsf{Gen}(\mathsf{seed})$
5: $\mathbf{Y}_2 := \lfloor (\mathbf{A}^T\mathbf{X}_2 + \mathbf{E}_2)/2^{t_2} \rceil$
6: $\mathbf{\Sigma}_2 := 2^{t_1}\mathbf{Y}_1^T\mathbf{X}_2 + \mathbf{E}_\sigma$
7: $(\mathbf{K}_2, \mathbf{V}) \leftarrow \mathsf{Con}(\mathbf{\Sigma}_2, \mathsf{params}; \mathsf{r}_2)$
8: $\overline{\mathbf{K}} = H(\mathbf{K}_2) \oplus \mathsf{S}$[a]
9: return $(\mathsf{ct} := (\mathbf{Y}_2, \mathbf{V}, \overline{\mathbf{K}}, \varpi), \mathsf{key} := H(\mathsf{k}, \mathsf{ct}))$[b]
```

[a]A variant is to set $\overline{\mathbf{K}} = H(\mathbf{K}_2, \mathsf{ct}) \oplus \mathsf{S}$.

[b]In practice, we may suggest to encrypt ϖ with the symmetric-key encryption scheme to be composed with KEM.

[27]For schemes based on RLWE, we may suggest $\kappa = 512$.

Algorithm 51 $\mathsf{key}' \leftarrow \mathsf{Decaps}(\mathsf{sk}, \mathsf{ct} = (\mathbf{Y}_2, \mathbf{V}, \overline{\mathbf{K}}, \varpi))$

```
1: $\mathbf{\Sigma}_1 := \mathbf{X}_1^T(2^{t_2}\mathbf{Y}_2)$
2: $\mathbf{K}_1 := \mathsf{Rec}(\mathbf{\Sigma}_1, \mathbf{V}, \mathsf{params})$
3: $\mathsf{S}' = H(\mathbf{K}_1) \oplus \overline{\mathbf{K}}$
4: $(\mathsf{k}', \varpi', \mathsf{r}_1', \mathsf{r}_2') = G(\mathsf{pk}, \mathsf{S}')$
5: $(\mathbf{X}_2', \mathbf{E}_2', \mathbf{E}_\sigma') \leftarrow \mathsf{Sample}(1^\kappa; \mathsf{r}_1')$
6: $\mathbf{A} := \mathsf{Gen}(\mathsf{seed})$ [a]
7: $\mathbf{Y}_2' := \lfloor(\mathbf{A}^T\mathbf{X}_2' + \mathbf{E}_2')/2^{t_2}\rceil$
8: $\mathbf{\Sigma}_2' := 2^{t_1}\mathbf{Y}_1^T\mathbf{X}_2' + \mathbf{E}_\sigma'$
9: $(\mathbf{K}_2', \mathbf{V}') \leftarrow \mathsf{Con}(\mathbf{\Sigma}_2', \mathsf{params}; \mathsf{r}_2')$
10: if $(\mathbf{Y}_2 = \mathbf{Y}_2' \bigwedge \mathbf{V} = \mathbf{V}' \bigwedge \mathbf{K}_1 = \mathbf{K}_2' \bigwedge \varpi = \varpi')$ then [b]
11:     $\mathsf{key}' = H(\mathsf{k}', \mathsf{ct})$
12: else
13:     $\mathsf{key}' = H(\mathsf{z}, \mathsf{ct})$
14: end if
15: return $\mathsf{key}'$
```

[a] $\mathbf{A}$ can be directly specified as part of pk and sk in place of seed.

[b] The condition whether $\varpi = \varpi'$ can be checked just after Step 4. We refrain from doing so to be against potential side-channel attacks.

K More Variants of CNKE

We can have more variants of CNKE:[28]

- Let $G : \{0,1\}^* \to \{0,1\}^\kappa \times \{0,1\}^\kappa \times \{0,1\}^{p(\kappa)}$, and set $(\mathsf{r}_A, \mathsf{d}_A, \mathsf{r}_A^{kem}) = G(\mathsf{s}_A^{kem}, I_B, \mathbf{Y}_A^{kt})$ and $(\mathsf{r}_B, \mathsf{d}_B, \mathsf{r}_B^{kem}) = G(\mathsf{s}_B^{kem}, I_A, \mathbf{Y}_B^{kt}, \mathbf{V}_B^{kt})$. The value d_A (resp., d_B) is sent by I_A (resp., I_B) in the second (resp., third) round, and is treated as a part of c_A^{kem}) (resp., c_B^{kem}). In this case, r_A (resp., r_B) does not need to be sent explicitly. Specifically, in this case, we set $\mathsf{c}_A^{ae} = \mathsf{Enc}_{\mathbf{K}_A^{ae}}(\mathsf{H}_A, I_A||\mathbf{Y}_A^{kt}||m_0)$ and $\mathsf{c}_B^{ae} = \mathsf{Enc}_{\mathbf{K}_B^{ae}}(\mathsf{H}_B, \mathbf{Y}_B^{kt}||\mathbf{V}_B^{kt})$.

- $(\mathsf{r}_A, \mathbf{K}_A^{ae}) = KDF'(\mathsf{s}_A^{kem}, \mathsf{c}_A^{kem})$, and $\mathbf{K}_B^{ae} = KDF'(\mathsf{s}_B^{kem}, \mathsf{c}_B^{kem})$. For instance, let $G : \{0,1\}^* \to \{0,1\}^\kappa \times \{0,1\}^{p(\kappa)}$, and set $(\mathsf{d}_A, \mathsf{r}_A^{kem}) = G(\mathsf{s}_A^{kem}, I_B, \mathbf{Y}_A^{kt})$ and $(\mathsf{d}_B, \mathsf{r}_B^{kem}) = G(\mathsf{s}_B^{kem}, I_A, \mathbf{Y}_B^{kt}, \mathbf{V}_B^{kt})$. The value d_A (resp., d_B) is sent by I_A (resp., I_B) in the second (resp., third) round, and is treated as a part of c_A^{kem}) (resp., c_B^{kem}). Let $\mathsf{c}_A^{ae} = \mathsf{Enc}_{\mathbf{K}_A^{ae}}(\mathsf{H}_A, I_A||\mathbf{Y}_A^{kt}||m_0)$ and $\mathsf{c}_B^{ae} = \mathsf{Enc}_{\mathbf{K}_B^{ae}}(\mathsf{H}_B, \mathbf{Y}_B^{kt}||\mathbf{V}_B^{kt})$. In this case, r_A (resp., r_B) is sent implicitly.

- A more aggressive variant is that: the values of r_A and r_B are removed from the AEAD ciphertext parts c_A^{ae} and c_B^{ae}, but still kept in the input of KDF. To our knowledge, as long as the session run is complete, it does not cause any meaningful vulnerability.

[28] All these variants are also applicable when using the CCA-secure KEMs proposed in [BDK+17, HHK17].