mmCipher: Batching Post-Quantum Public Key Encryption Made Bandwidth-Optimal

Hongxiao Wang
The University of Hong Kong

Ron Steinfeld
Monash University

Markku-Juhani O. Saarinen
Information Security Laboratory, Tampere University

Muhammed F. Esgin
Monash University

Siu-Ming Yiu*
The University of Hong Kong

Abstract

In applications such as secure group communication and broadcasting, it is important to *efficiently* deliver multiple messages to different recipients at once. To this end, multi-message multi-recipient Public Key Encryption (mmPKE) enables the batch encryption of multiple messages for multiple independent recipients in one go, significantly reducing costs–particularly bandwidth–compared to the trivial solution of encrypting each message individually. This capability is especially desirable in the post-quantum setting, where the ciphertext length is typically significantly larger than the corresponding plaintext. However, almost all prior works on mmPKE are limited to quantum-vulnerable traditional assumptions.

In this work, we propose the *first* CPA-secure mmPKE and Multi-Key Encapsulation Mechanism (mmKEM) from the *standard* Module Learning with Errors (MLWE) lattice assumption, named mmCipher-PKE and mmCipher-KEM, respectively. Our design proceeds in two steps: (i) We introduce a novel generic construction of mmPKE by proposing a new PKE variant—*extended reproducible PKE (XR-PKE)*—that enables the reproduction of ciphertexts through additional hints; (ii) We instantiate a lattice-based XR-PKE using a new technique that can precisely estimate the impact of such hints on the ciphertext security while also establishing suitable parameters. We believe both to be of independent interest. As a bonus contribution, we explore generic constructions of *adaptively secure* mmPKE, resisting adaptive corruption and chosen-ciphertext attacks.

We also provide an efficient implementation and thorough evaluation of the practical performance of our mmCipher. The results demonstrate substantial bandwidth and computational savings over the state-of-the-art. For example, for 1024 recipients, our mmCipher-KEM achieves a 23–45× reduction in bandwidth overhead, with ciphertexts only 4–9% larger than the plaintexts (*near optimal bandwidth*), while also offering a 3–5× reduction in computational cost.

*Corresponding author. Email: smyiu@cs.hku.hk

1 Introduction

Public Key Encryption (PKE) and Key Encapsulation Mechanism (KEM) are foundational cryptographic primitives that underpin secure digital communication systems–such as Zoom, Signal, and WhatsApp–serving billions of users. The rapid progress in quantum computing [21] has led to a shift towards post-quantum cryptography. In response, the National Institute of Standards and Technology (NIST) has selected Kyber, a lattice-based KEM/PKE, as a primary candidate for standardization [2]. However, these quantum-resistant constructions generally require significantly more bandwidth resources than their traditional counterparts [8]. Therefore, reducing communication costs for multiple recipients, even for moderately large number of recipients, say $N \geqslant 10$, is already of practical significance.

Multi-message multi-recipient PKE. To address this need, multi-message multi-recipient PKE (mmPKE) was introduced to efficiently batch encryption by Kurosawa [35]. Specifically, given N recipient public keys $(\mathsf{pk}_i)_{i\in[N]}$ and a message vector $(\mathsf{m}_i)_{i\in[N]}$, where each message m_i is intended for recipient i, an mmPKE can output a multi-recipient ciphertext $\mathbf{ct}$ that can be extracted as the individual ciphertext ct_i for each recipient i by any third party (e.g., delivery service server). Roughly speaking, each message m_i should remain private even given the multi-recipient ciphertext $\mathbf{ct}$ and all other recipient decryption keys $(\mathsf{sk}_j)_{j\in[N]\setminus i}$.

Compared to the trivial solutions where each message is encrypted separately, mmPKE allows for significant bandwidth savings, especially valuable in post-quantum settings where ciphertexts are large. We call an mmPKE (asymptotically) *bandwidth-optimal* if the length of its ciphertexts approaches the total length of its *plaintexts* (for a large number of recipients). When each message is an encapsulated key, we obtain the multi-key multi-recipient KEM (mmKEM). A special case of mmPKE and mmKEM is multi-recipient PKE (mPKE) and multi-recipient KEM (mKEM), which only support sending the *same* message or encapsulated key to all recipients. In this case, since each recipient receives the same

message, the security model excludes the *insider adversaries* (recipients).

Applications. In (m)mPKE/KEM schemes, the delivery service is modeled as a public bulletin board, where the sender uploads the multi-recipient ciphertext and each recipient downloads their corresponding individual ciphertext. Thus, a direct application is to replace individual PKE/KEM in multi-recipient scenarios to reduce communication and computation costs at the sender, which are typically much higher than those at each recipient, especially in the post-quantum setting. For example, [32] uses post-quantum mKEM to improve the efficiency of Messaging Layer Security (MLS) protocol, an IETF secure group messaging standard [9], by an order of magnitude. Similarly, [30] employs post-quantum mPKE to double the efficiency of Secure Group Messaging (SGM). In addition, [5] leverages mmPKE to generically build an efficient Continuous Group Key Agreement (CGKA). (m)mPKE is also a promising tool for improving the efficiency of messaging apps over short-range wireless mesh networks such as Bridgefy or BitChat [53] using Bluetooth, where bandwidth-efficient broadcasting is a natural requirement.

Besides secure digital communication, another compelling use case is confidential transactions in account-based blockchains, such as (Anonymous) Zether [16, 22] and PriDe CT [29].[1] Briefly, the spender submits a transaction containing a multi-recipient ciphertext and a well-formedness proof to the blockchain, where each amount is encrypted for its corresponding recipient. Furthermore, receiver anonymity can be achieved, similar to ring signatures [48], where the "real" recipients are hidden among "decoy" recipients, and identical zero-valued messages are encrypted for the latter. Thus, a *full CPA*-secure[2] mmPKE is required to ensure transaction confidentiality, so that no recipient can learn others' amounts, even if some amounts are *identical*. However, since the *only known* post-quantum mmPKE [6] cannot achieve *full CPA* security (as illustrated in Figure 1 and discussed later), it is not applicable in such scenarios. Considering that large transaction sizes (primarily due to ciphertext size) would lead to practically unacceptable transaction (gas) fees and blockchain storage is highly limited, we believe that the absence of *full CPA*-secure mmPKE is the primary bottleneck in shifting such confidential transactions to the post-quantum setting.

Existing works and challenges. Due to their practically appealing and theoretically interesting nature, studies on mmPKE/mmKEM [6, 10, 11, 35, 47] and mPKE/mKEM [7, 18, 30, 32, 39, 50, 55], have attracted significant attention. Among them, the foundational work on mmPKE was proposed by Bellare et al. in [10, 11] that significantly expanded Kurosawa's work [35] by: (1) introducing the *insider adversary* to formalize the *full CPA* security of mmPKE, ensuring that no recipient can obtain another recipient's message; (2) identifying possible attacks (e.g., rogue public key attacks) and introducing the *knowledge-of-secret-key* (KOSK) assumption–that is, each public key is assumed to be well-formed (i.e., the challenger knows the private key of each public key)–for protection; and (3) defining *reproducible PKE* to generically construct mmPKE. Informally, reproducibility requires the existence of an efficient algorithm that can transform a ciphertext into another ciphertext for a different public key and message while using the *same randomness*. They further noticed that only *discrete-log-based* encryption schemes, such as ElGamal [23] and Cramer–Shoup [19], are reproducible and can be extended to mmPKE under the KOSK assumption. Thus, they raised an *open question*—which has stood for over two decades—of whether (full CPA-secure) mmPKE schemes (and its underlying reproducible encryption) under other assumptions exist [10, page 12]. Unfortunately, such property remains *unknown* for post-quantum assumptions, particularly for lattices, since fresh randomness/noise in each ciphertext is inherently required and cannot be fully eliminated.

Currently, the only known post-quantum mmPKE [6] is generically constructed from mKEM, but it *only supports batching consecutive identical messages* in the message vector, as illustrated in Figure 1. Here, we identify two key limitations of this approach: (1) its efficiency is *close to trivial solution* when messages are independent, and (2) it *cannot achieve full CPA* security, as it leaks the structure of the input message vector, i.e., given the multi-recipient ciphertext, others can identify whether any two consecutive messages in the message vector are identical. These significantly limit the application of [6] in many practical scenarios, such as confidential transactions [16, 22, 29], as discussed above.

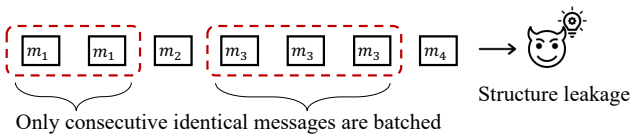
Overall, despite strong practical demand and rapid progress, significant challenges remain in fully realizing the potential of mmPKE in post-quantum settings, especially for generic constructions, leading to our question:

Question: Are there any simple and efficient generic constructions of fully batched mmPKE based on the post-quantum assumptions, while enjoying full CPA-security, regardless of the message vector?

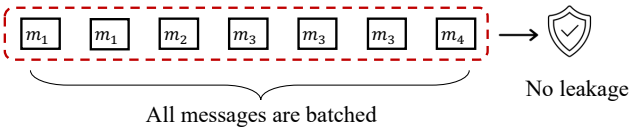
We refer to Table 1 for a summary of the existing post-quantum mmPKE schemes. We note that this comparison excludes [6], as their benchmarks only focus on mKEM, which we consider incomparable to the case of mmKEM/mmPKE. Furthermore, in our setting, since the messages/keys are *independent* of each other, the probability of consecutive identical messages appearing in the message vector is *negligible*. Therefore, as [6] only supports batching consecutive identical messages, its performance under independent messages would be equivalent to the trivial solution with Kyber.

[1] These confidential transactions *implicitly* employ mmPKE, i.e., they directly utilize ElGamal-based mmPKE [10, 35] as a fundamental building block.

[2] What we call "full CPA" security here is the standard CPA security notion. In contrast, some earlier works such as [6] only obtain a *weaker* form of CPA security, which *does not* protect the structure of the message vector.



(a) Existing post-quantum mmPKE [6]



(b) Our scheme

Figure 1: Comparison between the existing post-quantum mmPKE [6] and our scheme. boxes indicates that the enclosed message is encrypted as a ciphertext for an independent recipient, while boxes indicates that each enclosed ciphertexts are batched. Since [6] only supports batching consecutive identical messages, the structure of the message vector can be inferred from its ciphertext (e.g., the first two messages are identical), whereas our scheme supports batching all messages while protecting this structure.

Table 1: Comparison of current lattice-based CPA-secure mmPKE/mmKEM schemes, for $N = 1024$ recipients.

Scheme	PQ-Sec. Level	Enc. Size (KB)	Improve. Factor (×)	Enc. Time (ms)	Full CPA
*Plaintext**	–	32	–	–	×
Baseline:	128	768	–	36	
Kyber [15]	192	1088	–	58	✓
(ML-KEM [43])	256	1568	–	87	
Our work:	128	**33**	**23.1**×	12	
mmCipher-KEM	192	**34**	**31.6**×	16	✓
(Cons. 4.3+A.1)	256	**35**	**44.7**×	17	
Our work:	128	65	11.8×	13	
mmCipher-PKE	192	66	16.4×	16	✓
(Cons. 4.3+5.3)	256	67	23.3×	18	

* Here, Enc. Size is the plaintext size of all encapsulated keys/messages (i.e., *optimal bandwidth*).

Notes: For each scheme, we report the size of the multi-recipient ciphertext (Enc. Size) in kilobytes (KB) as well as the improvement, relative to the trivial solution with CPA-secure Kyber (parameterized by ML-KEM standard [43]), and the encryption/encapsulation time (Enc. Time) in milliseconds (ms), under 128-bit, 192-bit, and 256-bit post-quantum security levels (PQ-Sec. Level), respectively. Each message/key is 256 bits and *independently* chosen across 1024 recipients. *Full CPA* indicates that the scheme protects both semantics and structure of the message vector.

1.1 Contribution

In this work, we answer the above question affirmatively by proposing the *first* full CPA-secure mmPKE and mmKEM, based on the *standard* MLWE assumption, named mmCipher-PKE and mmCipher-KEM, respectively. Specifically, we introduce a new generic construction of mmPKE from a novel variant of PKE, called *extended reproducible PKE (XR-PKE)*. We then present lattice-based instantiations of XR-PKE and provide parameter sets for different security levels. Lastly, we give an efficient implementation and a thorough performance evaluation of our mmCipher. The main contributions of our work are summarized as follows. For detailed technical discussions, see Section 2.

New generic construction of post-quantum mmPKE. Our first contribution is a new generic construction of post-quantum mmPKE from XR-PKE. To accommodate the post-quantum setting—particularly the lattice-based setting—we formally define XR-PKE, which significantly enhances the functionality of the original reproducible PKE [10], in both syntax (by incorporating hints into the reproduction algorithm and providing a hint generation algorithm) and security model (by modeling the semantic security of ciphertexts given the associated hints). We believe such new generic constructions could be of independent interest that may spark other post-quantum instantiations, such as code-based schemes.

mmCipher: the first mmPKE instantiations from lattices. Our second contribution is the construction of lattice-based XR-PKE and XR-KEM schemes, from which we instantiate the *first* lattice-based mmPKE. To achieve extended reproducibility, we leverage the decryption error as a hint to enable ciphertext reproduction. To establish the semantic security of ciphertexts given the associated hints, we rely on the Matrix Hint-MLWE assumption [26], for which a reduction from the standard MLWE assumption exists under suitable parameter choices, arguing that the security impact of the hints is negligible. Along the way, as a bonus technical contribution, we generalize the underlying matrix in Matrix Hint-MLWE to the non-square setting and identify a missing efficient sampleability condition in the parameter instantiation for the reduction of [26]. Both results may be of independent interest for other applications of Hint-MLWE, e.g., [1, 34, 36].

Then, following our generic construction, we instantiate two lattice-based CPA-secure mmPKE under the KOSK assumption: (1) an mmPKE for short messages (mmCipher-PKE) and (2) a hybrid mmKEM-DEM scheme for arbitrary-length messages (mmCipher-KEM). Both achieve *full CPA-security*, protecting both semantics and structure of input message vector, thereby preventing the identification of identical messages. This represents a key improvement over [6] and significantly broadens potential applications.

Furthermore, to fit the real-world applications, we introduce a compiler in Remark 4.5, that removes the KOSK assumption from mmPKE/mmKEM with polynomial-size

number of recipients by leveraging a *multi-proof extractable* Non-Interactive Zero-Knowledge (NIZK) proof system. While [10] observed that the KOSK assumption could be removed using NIZK, no concrete construction or formal security proof was given prior to this work.

Bandwidth-optimal mmPKE implementation and evaluation. We also provide a C implementation[3] of our lattice-based mmPKE schemes (i.e., mmCipher), together with computational performance and bandwidth benchmarks. Compared to the state-of-the-art, the performance of our mmCipher is independent of the message vector structure, i.e., whether the message vector has identical or distinct messages. For $N = 1024$ recipients and different security levels (128-, 192-, 256-bit), our mmCipher-KEM and mmCipher-PKE achieve a 23–45× and 12–23× reduction in bandwidth overhead, respectively, and offer a 3–5× reduction in computational cost, compared to [6] with independent messages and the trivial solution with Kyber. Notably, by using a reconciliation mechanism [45], each *public-key-dependent* ciphertext in our mmCipher-KEM is minimized to the size of the encapsulated key (e.g., 256 bits), thereby making our construction *asymptotically bandwidth-optimal*, with ciphertext size only 4% (resp. 9%) larger than the plaintext size for 128-bit (resp. 256-bit) security levels, when $N = 1024$ recipients.

Generic construction of adaptively secure mmPKE. As a bonus contribution, we propose generic constructions that transform the CPA-secure mmPKE into an adaptively secure mmPKE, achieving security against adaptive corruption and CCA. Specifically, due to the absence of fully batched post-quantum mmPKE constructions, there remains a gap in achieving *adaptive* security in such settings. For example, since the public parameters and randomness are shared among recipients, standard techniques such as the Fujisaki-Okamoto (FO) transform [27, 52], lossy trapdoor functions [46, 51], and the BCHK transform via IBE [12, 17, 40] cannot be applied in the post-quantum mmPKE setting. To this end, we generalize the Naor-Yung paradigm [41, 49] to the mmPKE setting. Furthermore, by leveraging the structure of mmPKE, we can *safely merge* the two ciphertexts into a *single* multi-recipient ciphertext by doubling recipient number from N to $2N$. As a result, only one public-key-independent ciphertext needs to be generated, significantly reducing overhead. The detailed construction is provided in Appendix D.

2 Technical Overview

In this section, we provide a self-contained overview of our techniques for constructing a lattice-based mmPKE. The discussion is given at a high level to provide an intuitive understanding of our approach.

We begin by recalling the syntax of mmPKE [10]. Specifically, the setup, key generation and decryption algorithms of mmPKE are the same as the ones in the standard PKE. For the multi-encryption, i.e., $\mathbf{ct} \leftarrow \mathsf{mmEnc}(\mathsf{pp}, (\mathsf{pk}_i)_{i\in[N]}, (\mathsf{m}_i)_{i\in[N]})$, it takes as input the public parameter pp, a set of public keys $(\mathsf{pk}_i)_{i\in[N]}$ along with a message vector $(\mathsf{m}_i)_{i\in[N]}$ and outputs a multi-recipient ciphertext $\mathbf{ct}$. The multi-recipient ciphertext $\mathbf{ct}$ can later be extracted to the individual ciphertext ct_i for the public key pk_i by some extraction algorithm.

The correctness of mmPKE is that each individual ciphertext ct_i can be successfully decrypted to the message m_i by the corresponding private key sk_i.

The full IND-CPA security model of mmPKE is more complicated than standard PKE, since it considers the *insider attack* where the adversary is allowed to be some recipients, i.e., generate some public keys for the challenger to encrypt the challenge ciphertext. Specifically, the adversary selects ℓ honestly generated (i.e., challenger's) public keys $(\mathsf{pk}_i)_{i\in[\ell]}$ and ℓ message pairs $(\mathsf{m}_i^0, \mathsf{m}_i^1)_{i\in[\ell]}$ It also chooses $N-\ell$ adversarially generated (i.e., adversary's) public keys $(\mathsf{pk}_i)_{i\in[\ell:N]}$ (along with the corresponding private keys $(\mathsf{sk}_i)_{i\in[\ell:N]}$ when under the KOSK assumption) and the associated messages $(\mathsf{m}_i)_{i\in[\ell:N]}$. It should be infeasible for the adversary to distinguish the challenge ciphertext $\mathbf{ct} \leftarrow \mathsf{mmEnc}(\mathsf{pp}, (\mathsf{pk}_i)_{i\in[N]}, (\mathsf{m}_i^b)_{i\in[\ell]}, (\mathsf{m}_i)_{i\in[\ell:N]})$ for a randomly chosen bit $b \in \{0,1\}$.

Recall: traditional mmPKE from reproducible PKE. Before delving into the specifics of our approach, it is useful to recall the traditional constructions of mmPKE from reproducible PKE [10]. The syntax, correctness and security definition of reproducible PKE is the same as standard PKE, except introducing a reproducibility property.

The reproducibility requires that given a ciphertext $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, \mathsf{m}; \mathsf{r})$ which encrypts the message m with the public key pk and some randomness r, there exists an efficient algorithm, called reproduction algorithm, satisfying

$$\mathsf{Enc}(\mathsf{pp}, \mathsf{pk}', \mathsf{m}'; \mathsf{r}) = \mathsf{Rep}(\mathsf{pp}, \mathsf{ct}, \mathsf{m}', \mathsf{sk}', \mathsf{pk}').$$

It means that the Rep algorithm can use the private key sk' to reproduce a ciphertext ct to another ciphertext ct' for the corresponding public key pk' and different message m' but with the *same* randomness r. For example, for ElGamal scheme, given a ciphertext $(g^r, m \cdot (g^x)^r)$ for public key g^x and message m, the other ciphertext for public key $g^{x'}$ and message m' can be reproduced as $(g^r, m' \cdot (g^r)^{x'})$ by the private key x'.

Now, let us discuss how [10] constructs an mmPKE from reproducible PKE. The setup, key generation, and decryption algorithms of mmPKE are the same as the ones in reproducible PKE. In multi-encryption, it uses the same randomness r to encrypt each message m_i for the corresponding public key pk_i to the ciphertext $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{m}_i; \mathsf{r})$ and concatenate the ciphertexts together as multi-recipient ciphertext $\mathbf{ct} := (\mathsf{ct}_1, \ldots, \mathsf{ct}_N)$.

Notably, if all ct_i have a same part due to the randomness reuse, this part *only* needs to be computed and communicated

[3] Provided in our artifact: https://doi.org/10.5281/zenodo.17849532

once in the multi-recipient ciphertext and that is the reason for the bandwidth and computation savings of mmPKE. For example, the part g^r of the ciphertext only needs to be generated once in ElGamal-based mmPKE which can save about half bandwidth and computation compared to the trivial solution.

To reduce the security of mmPKE to that of the underlying reproducible PKE, the reduction, under the KOSK assumption, can obtain the private key of other recipients and generate the multi-recipient ciphertext by reproducing its challenge ciphertext. For details, we refer readers to [10, Theorem 6.2].

Challenge I: generic construction of post-quantum mmPKE from XR-PKE. The major limitation of the above mmPKE [10] is that it does not seem to extend to the post-quantum setting, especially lattice-based setting. The reason is that the randomness of the ciphertext in lattice-based PKE schemes cannot be fully reused as in the discrete-log-based assumptions. In particular, in encryption scheme based on the LWE lattice problem, the ciphertext for message m typically takes the form $(\mathbf{Ar}+\mathbf{e}_u, \langle \mathbf{b}, \mathbf{r}\rangle + y + \lfloor q/2 \rfloor \cdot m)$. For security, the message error term y cannot be reused across multiple messages/public keys. Moreover, there are additional reproducibility security issues caused by such error terms.

To get around this issue, we first consider the (extended) reproducible PKE in a *decomposable* variant. Specifically, a decomposable encryption algorithm Enc takes as input the randomness $\mathsf{r} := (\mathsf{r}_0, \hat{\mathsf{r}})$ and creates a public-key-*independent* ciphertext $\mathsf{ct}_0 \leftarrow \mathsf{Enc}^{\mathsf{i}}(\mathsf{pp}; \mathsf{r}_0)$ and a public-key-*dependent* ciphertext $\widehat{\mathsf{ct}} \leftarrow \mathsf{Enc}^{\mathsf{d}}(\mathsf{pp}, \mathsf{pk}, \mathsf{m}; \mathsf{r}_0, \hat{\mathsf{r}})$. Note that the randomness $\hat{\mathsf{r}}$ in key-dependent ciphertext can be set empty, i.e., $\hat{\mathsf{r}} := \perp$, if it is unnecessary. We view this as a natural formalization of (extended) reproducible PKE as it is satisfied by all the constructions that we are aware of.

Therefore, we intend to reuse only the randomness r_0 in *key-independent* ciphertext instead of the entire randomness $\mathsf{r} = (\mathsf{r}_0, \hat{\mathsf{r}})$, so that we can achieve the same savings in bandwidth and computation as fully reusing the randomness when constructing mmPKE. We formalize this new primitive, called XR-PKE, which significantly improves upon reproducible PKE in both syntax and security model.

From the perspective of syntax, to formalize the property of reproducibility, we introduce an additional input h', called *hint*, into the *reproduction* algorithm. Looking ahead to our lattice-based instantiation, the hint there will be used to provide randomized information on the ciphertext error terms needed to reproduce the ciphertext for new recipient. We require that, given a ciphertext $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, \mathsf{m}; \mathsf{r}_0, \hat{\mathsf{r}})$, the following property always holds

$$\mathsf{Enc}(\mathsf{pp}, \mathsf{pk}', \mathsf{m}'; \mathsf{r}_0, \hat{\mathsf{r}}') = \mathsf{Rep}(\mathsf{pp}, \mathsf{ct}, \mathsf{m}', \mathsf{pk}', \mathsf{sk}', \mathsf{h}').$$

Additionally, we provide an auxiliary algorithm, named *hint generation* algorithm, for generating the hint h'. It takes as input the public parameter pp, the reused randomness r_0, a fresh randomness $\hat{\mathsf{r}}'$, and a public-private key pair $(\mathsf{pk}', \mathsf{sk}')$, i.e.,

$$\mathsf{h}' \leftarrow \mathsf{HintGen}(\mathsf{pp}, \mathsf{r}_0, \hat{\mathsf{r}}', \mathsf{pk}', \mathsf{sk}').$$

Regarding the security model, we require that the adversary's advantage against semantic security remains negligible, even given the hints associated with the challenge ciphertext. More precisely, we introduce a *hint query phase* before the adversary output in the security game. In the hint query phase, after receiving the challenge ciphertext $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{pk}^*, \mathsf{m}_b^*; \mathsf{r}_0, \hat{\mathsf{r}}^*)$, the adversary is allowed to query N hints on the challenge ciphertext by N public-private key pairs $(\mathsf{pk}_i, \mathsf{sk}_i)_{i\in[N]}$. The challenger then computes the hints as $(\mathsf{h}_i)_{i\in[N]} \leftarrow \mathsf{HintGen}(\mathsf{pp}, \mathsf{r}_0, (\hat{\mathsf{r}}_i)_{i\in[N]}, (\mathsf{pk}_i, \mathsf{sk}_i)_{i\in[N]})$ and returns them to the adversary. The formal definitions of XR-PKE are provided in Section 4.

We now describe the generic construction of post-quantum mmPKE from XR-PKE. The setup, key generation, and decryption algorithms are identical to those in XR-PKE, except that the setup algorithm additionally takes the recipient number N as input. In the multi-encryption algorithm mmEnc, the randomness is structured as $\mathsf{r} := (\mathsf{r}_0, \hat{\mathsf{r}}_1, \ldots, \hat{\mathsf{r}}_N)$. The algorithm first generates a key-independent ciphertext $\mathsf{ct}_0 \leftarrow \mathsf{Enc}^{\mathsf{i}}(\mathsf{pp}; \mathsf{r}_0)$, then computes N key-dependent ciphertexts $\widehat{\mathsf{ct}}_i \leftarrow \mathsf{Enc}^{\mathsf{d}}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{m}_i; \mathsf{r}_0, \hat{\mathsf{r}}_i)$, and concatenates them as multi-recipient ciphertext $\mathbf{ct} := (\mathsf{ct}_0, \widehat{\mathsf{ct}}_1, \ldots, \widehat{\mathsf{ct}}_N)$. For each recipient, the individual ciphertext $\mathsf{ct}_i := (\mathsf{ct}_0, \widehat{\mathsf{ct}}_i)$ can be extracted from $\mathbf{ct}$ and decrypted by the private key sk_i.

Finally, we outline the security reduction from our mmPKE to its underlying XR-PKE. The reduction largely follows the structure of the above traditional mmPKE [10], except that, before reproducing the ciphertext, it sends the public-private key pairs $(\mathsf{pk}_i, \mathsf{sk}_i)_{i\in[N]}$ to its challenger during the *hint query phase*, and receive the corresponding hints $(\mathsf{h}_i)_{i\in[N]}$ to complete the reproduction. The detailed proof is given in Theorem 4.4. We emphasize that the *hints* and their associated algorithms are *only used* in the security reduction, *not* in real-world deployment. Both our mmPKE and the traditional mmPKE rely on the KOSK assumption, and we show how to explicitly remove this requirement via NIZK in Remark 4.5.

Challenge II: constructing lattice-based XR-PKE. To the best of our knowledge, no existing lattice-based PKE schemes currently satisfy the extended reproducibility property. The primary reason is that they fail to achieve semantic security of the ciphertext given the associated hints.

To this end, we begin with one of the most efficient lattice-based PKE schemes, Kyber [15], and show a step-by-step transformation to XR-PKE. Our approach may be of independent interest, as it applies to both plain and ring-based lattice settings, such as, Frodo [14] and NewHope [4].

At the beginning, a uniformly random matrix $\mathbf{A} \leftarrow \mathcal{U}(\mathcal{R}_q^{m\times n})$ is sampled as the public parameter where $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^d+1)$ and $\mathcal{R} = \mathbb{Z}[X]/(X^d+1)$. Then, the public key is generated by

$$\mathbf{b} := \mathbf{A}^\top \mathbf{s} + \mathbf{e} \tag{1}$$

where the private key $(\mathbf{s},\mathbf{e}) \leftarrow \mathcal{U}(\mathbb{S}_\nu^m) \times \mathcal{U}(\mathbb{S}_\nu^n)$ has coefficients uniformly randomly sampled from set $[-\nu,...,\nu]$ for $\nu \ll q$. To encrypt a message m, the ciphertext can be *decomposed* into two parts: a key-*independent* ciphertext $\mathbf{c}$, a key-*dependent* ciphertexts u as below,

$$\mathbf{c} := \mathbf{A}\mathbf{r} + \mathbf{e}_u, \qquad u := \langle \mathbf{b}, \mathbf{r} \rangle + y + \lfloor q/2 \rceil \cdot m, \tag{2}$$

where randomness are sampled from some distribution χ over $\mathcal{R}$ as $\mathbf{r} \leftarrow \chi^n$, $\mathbf{e}_u \leftarrow \chi^m$, $y \leftarrow \chi$, and $m \in \{0,1\}^d$ (interpreted as a polynomial in $\mathcal{R}$ with binary coefficients). To decrypt the ciphertext $(\mathbf{c},u)$ to the message m, the recipient uses the private key to compute $u - \langle \mathbf{c},\mathbf{s} \rangle$. Using Equations (1) and (2), we have

$$u - \langle \mathbf{c},\mathbf{s} \rangle = \langle -\mathbf{s} || \mathbf{e}, \mathbf{e}_u || \mathbf{r} \rangle + y + \lfloor q/2 \rceil \cdot m.$$

where $||$ denotes the usual concatenation. If the PKE is correct, i.e., $\| \langle -\mathbf{s} || \mathbf{e}, \mathbf{e}_u || \mathbf{r} \rangle + y \|_\infty \leqslant \lfloor q/4 \rceil$, after rounding the above term as $\lfloor u - \langle \mathbf{c},\mathbf{s} \rangle \rceil_2$, each recipient can obtain the message m and the decryption error $h := \langle -\mathbf{s} || \mathbf{e}, \mathbf{e}_u || \mathbf{r} \rangle + y$.

Here we use the decryption error h as the hint to reproduce the ciphertext. Given a ciphertext $(\mathbf{c},u)$, a new ciphertext $(\mathbf{c},u')$ for another public key $\mathbf{b}' = \mathbf{A}^\top \mathbf{s}' + \mathbf{e}'$ and message m' using randomness $((\mathbf{r},\mathbf{e}_u),y')$ can be reproduced by the corresponding private key $\mathbf{s}'$ and the hint h' as

$$u' := \langle \mathbf{c},\mathbf{s}' \rangle + h' + \lfloor q/2 \rceil \cdot m' = \langle \mathbf{b}', \mathbf{r} \rangle + y' + \lfloor q/2 \rceil \cdot m'. \tag{3}$$

The hint h' is computed via

$$h' = \langle -\mathbf{s}' || \mathbf{e}', \mathbf{e}_u || \mathbf{r} \rangle + y' \tag{4}$$

using the reused independent randomness $\mathsf{r}_0 = (\mathbf{r},\mathbf{e}_u)$, the corresponding private key $(\mathbf{s}',\mathbf{e}')$ and a fresh dependent randomness $\hat{\mathsf{r}}' = y'$. This technique can naturally extend to multiple hints h_i given multiple $(\mathbf{b}_i,\mathbf{s}_i)$ and y_i. As a result, we obtain the *reproduction* algorithm and *hint generation* algorithm.

Since the hints $(h_i)_{i\in[N]}$ reveal partial information about the randomness $(\mathbf{r},\mathbf{e}_u)$, establishing semantic security of the ciphertext is non-trivial. To address this challenge, we rely on the Matrix Hint-MLWE assumption [26] to precisely measure how much information on the randomness (i.e., the MLWE secret) is leaked from the hints and to make that impact on the hardness of MLWE ciphertext negligible under suitable parameter setting. Informally, the Matrix Hint-MLWE assumption states that given a hint vector $\mathbf{h} \in \mathcal{R}^\ell$ where $\mathbf{h} := \mathbf{R}\hat{\mathbf{r}} + \mathbf{y}$, the MLWE instance $[\mathbf{I}|\mathbf{A}]\hat{\mathbf{r}}$ is still indistinguishable from the uniformly random values if $\hat{\mathbf{r}}$ and $\mathbf{y}$ are sampled from appropriate discrete Gaussian distributions. Here, the hint $\mathbf{h}$ in the Matrix Hint-MLWE assumption is composed of the matrix product of an MLWE secret vector $\hat{\mathbf{r}}$ and a bounded *square* matrix $\mathbf{R}$ picked by the adversary, masked by a fresh vector $\mathbf{y}$.

From our intuition in XR-PKE, the hints are in the form of $h_i := \langle \boldsymbol{\gamma}_i, \hat{\mathbf{r}} \rangle + y_i$ for $i \in [N]$. Here, h_i is composed of the inner product of an MLWE secret vector $\hat{\mathbf{r}} := (y || \mathbf{e}_u || \mathbf{r})$ and a vector $\boldsymbol{\gamma}_i := (0 || -\mathbf{s}_i || \mathbf{e}_i)$, which is bounded by $\|\boldsymbol{\gamma}_i\|_\infty \leqslant \nu$ and chosen by the adversary, and masked by a fresh element y_i. Thus, we instantiate Matrix Hint-MLWE for XR-PKE by concatenating the hints $(h_i)_{i\in[N]}$ as a hint vector $\mathbf{h}$ such that $\mathbf{h} := \mathbf{R}\hat{\mathbf{r}} + \mathbf{y}$ where $\mathbf{R} := (\boldsymbol{\gamma}_i^\top)_{i\in[N]}$ and $\mathbf{y} := (y_i)_{i\in[N]}$.

To this end, we generalize the matrix $\mathbf{R}$ to a *non-square* setting, refine the reduction of Matrix Hint-MLWE from standard MLWE, and derive new conditions on the parameters, as presented in Theorem 5.2. To satisfy these conditions, we carefully choose two discrete Gaussian distributions $\mathcal{D}_{\sigma_0}$, $\mathcal{D}_{\sigma_1}$ for the randomness $(\mathbf{r},\mathbf{e}_u)$ and y, respectively, rather than the uniform distribution over intervals used in Kyber. The latter appears to preclude an efficient Matrix Hint-MLWE to standard MLWE security reduction. More details are provided in Section 5.1 and Section 5.3.

Finally, we employ the reconciliation mechanism from [45] and the bit-dropping technique as in Kyber [15] to compress the ciphertext, particularly the key-dependent ciphertexts, as much as possible. These optimizations bring the bandwidth cost of our mmPKE construction close to *optimal*.

3 Preliminaries

In this section, we provide some of the preliminaries needed for our paper. The additional preliminaries are deferred to the full version [54] due to space constraints.

3.1 Notation

Let $\lambda \in \mathbb{N}$ denote the security parameter. For a positive integer n, we denote the set $\{0,\ldots,n-1\}$ by $[n]$ and the set $\{\ell,\ldots,n-1\}$ by $[\ell:n]$. For a positive integer q, we denote $\mathbb{Z}_q$ as the integers modulo q and $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^d+1)$ as the polynomials modulo q and X^d+1. For positive integer ν, we write $\mathbb{S}_\nu$ to denote the set of polynomials in $\mathcal{R}_q$ with infinity norm bounded by ν. The size of the $\mathbb{S}_\nu$ coefficient support is denoted $\bar{\nu} \leqslant 2\nu+1$; for example $\nu = 1, \bar{\nu} = 2$ indicates binary polynomials. We denote assignment as $:=$, e.g., $x := y$ assigns the value of y to x. We denote sampling or output as $\leftarrow$, e.g., $x \leftarrow \mathcal{D}$ indicates that x is sampled from the distribution $\mathcal{D}$, and $x \leftarrow \mathsf{A}(y)$ denotes that x is the output of probabilistic polynomial time (PPT) algorithm A given input y. Particularly, we write $x \leftarrow S$ when $x \in S$ is sampled uniformly randomly from the finite set S. We denote the uniform distribution on a set S as $\mathcal{U}(S)$. We denote $\mathsf{poly}(\lambda)$ as polynomial functions such that $\mathsf{poly}(\lambda) = \bigcup_{c\in\mathbb{N}} O(\lambda^c)$ and $\mathsf{neg}(\lambda)$ as negligible functions such that $\mathsf{neg}(\lambda) = \bigcap_{c\in\mathbb{N}} o(\lambda^{-c})$. We denote rounding operation as $\lfloor\cdot\rceil$, e.g., $\lfloor a \rceil$ rounds the result to the nearest integer of a. For any two subset X, Y of some additive group, we define $-X = \{-x : x \in X\}$ and $X + Y = \{x+y : x \in X, y \in Y\}$.

Vector and matrix. We denote bold lowercase letters as vectors of polynomial elements, e.g., $\mathbf{u} \in \mathcal{R}_q^m$, bold uppercase letters as matrices of polynomial elements, e.g., $\mathbf{U} \in \mathcal{R}_q^{m\times n}$, low-

ercase letters with an arrow as vectors of integers or reals, e.g., $\vec{a} \in \mathbb{Z}_q^m$, and uppercase letters as matrices of integers or reals, e.g., $A \in \mathbb{R}_q^{m \times n}$. For a polynomial element, e.g., $a \in \mathcal{R}_q$, we define its negacyclic matrix as $\bar{A} := \Gamma(a) \in \mathbb{Z}_q^{d \times d}$. Similarly, for a polynomial vector and matrix, e.g., $\mathbf{b} \in \mathcal{R}^m$ and $\mathbf{D} \in \mathcal{R}_q^{m \times n}$, we define their negacyclic matrix as $\bar{B} := \Gamma(\mathbf{b}) \in \mathbb{Z}^{md \times d}$ and $\bar{D} := \Gamma(\mathbf{D}) \in \mathbb{Z}_q^{md \times nd}$, respectively, where each polynomial element in the vector and matrix is replaced by its negacyclic matrix. For the vectors over integers and polynomials, we denote their inner product as $\langle \cdot, \cdot \rangle$, e.g., $\langle \vec{a}, \vec{b} \rangle$ and $\langle \mathbf{a}, \mathbf{b} \rangle$. For a vector $\mathbf{a}$ (or $\vec{a}$), we write $\|\mathbf{a}\|$, $\|\mathbf{a}\|_1$, and $\|\mathbf{a}\|_\infty$ to denote its ℓ_2-norm, ℓ_1-norm and ℓ_∞-norm, respectively. For a matrix $\mathbf{A}$ (or A), we write $\|\mathbf{A}\|$, $\|\mathbf{A}\|_1$ and $\|\mathbf{A}\|_\infty$ to denote its matrix 2-norm (largest singular value), matrix 1-norm (maximum column ℓ_1-norm), and matrix ∞-norm (maximum row ℓ_1-norm), respectively. We write $\sigma_{\min}(\mathbf{A})$ and $\sigma_{\max}(\mathbf{A})$ to denote the smallest and largest singular values of $\mathbf{A}$, respectively.

3.2 Lattice Preliminaries

We show the definition of the standard lattice-based problem. Additional lattice preliminaries are deferred to the full version [54] due to space constraints.

Definition 3.1 (MLWE Problem)**.** Let $m, n > 0$ be positive integers. Let χ be an error distribution over $\mathcal{R}^{m+n}$, $\mathbf{A} \leftarrow \mathcal{U}(\mathcal{R}_q^{m \times n})$. Let $\mathbf{r} \leftarrow \chi$ be a secret vector and $\mathbf{u} \leftarrow \mathcal{U}(\mathcal{R}_q^m)$ be a uniformly random vector. The MLWE problem, denoted by $\mathsf{MLWE}_{\mathcal{R},m,n,q,\chi}$, asks an adversary $\mathcal{A}$ to distinguish between $(\mathbf{A}, [\mathbf{I}_m | \mathbf{A}]\mathbf{r})$ and $(\mathbf{A}, \mathbf{u})$. We say $\mathsf{MLWE}_{\mathcal{R},m,n,q,\chi}$ is hard if for any PPT adversary $\mathcal{A}$, the following advantage of $\mathcal{A}$ is negligible in λ,

$$\mathsf{Adv}^{\mathsf{MLWE}}_{\mathsf{para},\mathcal{A}}(\lambda) := \left| \Pr\left[b = 1 \;\middle|\; \begin{array}{c} \mathbf{A} \leftarrow \mathcal{U}(\mathcal{R}_q^{m \times n}), \mathbf{r} \leftarrow \chi \\ b \leftarrow \mathcal{A}(\mathbf{A}, [\mathbf{I}_m | \mathbf{A}]\mathbf{r}) \end{array} \right] - \Pr\left[b = 1 \;\middle|\; \begin{array}{c} \mathbf{A} \leftarrow \mathcal{U}(\mathcal{R}_q^{m \times n}), \mathbf{u} \leftarrow \mathcal{U}(\mathcal{R}_q^m) \\ b \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{u}) \end{array} \right] \right|$$

where $\mathsf{para} = (\mathcal{R}, m, n, q, \chi)$.

3.3 Multi-Message Multi-Recipient Public Key Encryption

Basically, an mmPKE scheme allows a sender to encrypt a set of messages to a set of public keys. We generalize the syntax of decomposable mPKE in [32] to mmPKE as follows. Like [32], our definition of mmPKE can capture all kinds of mmPKE as well.

Definition 3.2 (Decomposable Multi-Message Multi-Recipient PKE)**.** A decomposable mmPKE scheme with a public-private key pair space $\mathcal{K}$, a message space $\mathcal{M}$, a multi-recipient ciphertext space $\mathcal{C}$, and an individual ciphertext space $\mathcal{C}_\mathsf{s}$ consists of the following algorithms:

- $\mathsf{pp} \leftarrow \mathsf{mmSetup}(1^\lambda, N)$: On input a security parameter 1^λ and a recipient number N, it outputs a public parameter pp.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{mmKGen}(\mathsf{pp})$: On input a public parameter pp, it outputs a public-private key pair $(\mathsf{pk}, \mathsf{sk}) \in \mathcal{K}$.
- $\mathbf{ct} := (\mathsf{ct}_0, (\widehat{\mathsf{ct}}_i)_{i \in [N]}) \leftarrow \mathsf{mmEnc}(\mathsf{pp}, (\mathsf{pk}_i)_{i \in [N]}, (\mathsf{m}_i)_{i \in [N]}; \mathsf{r}_0, (\hat{\mathsf{r}}_i)_{i \in [N]})$: On input a public parameter pp, N public keys $(\mathsf{pk}_i)_{i \in [N]}$, N messages $(\mathsf{m}_i)_{i \in [N]}$, $(N+1)$ randomness r_0, $(\hat{\mathsf{r}}_i)_{i \in [N]}$, it can be split into two algorithms:
  - $\mathsf{ct}_0 \leftarrow \mathsf{mmEnc}^{\mathsf{i}}(\mathsf{pp}; \mathsf{r}_0)$: On input a public parameter pp, and a randomness r_0, it outputs a public-key-*independent* ciphertext ct_0.
  - $\widehat{\mathsf{ct}}_i \leftarrow \mathsf{mmEnc}^{\mathsf{d}}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{m}_i; \mathsf{r}_0, \hat{\mathsf{r}}_i)$: On input a public parameter pp, a public key pk_i, a message $\mathsf{m}_i \in \mathcal{M}$, and randomness r_0, $\hat{\mathsf{r}}_i$, it outputs a public-key-*dependent* ciphertext $\widehat{\mathsf{ct}}_i$.
- $\mathsf{ct}_i := (\mathsf{ct}_0, \widehat{\mathsf{ct}}_i) / \perp \leftarrow \mathsf{mmExt}(\mathsf{pp}, i, \mathbf{ct})$: On input a public parameter pp, a multi-recipient ciphertext $\mathbf{ct} \in \mathcal{C}$, and an index $i \in \mathbb{N}$, it deterministically outputs the individual ciphertext $\mathsf{ct}_i \in \mathcal{C}_\mathsf{s}$ or a symbol $\perp$ to indicate extraction failure.
- $\mathsf{m} / \perp \leftarrow \mathsf{mmDec}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct})$: On input a public parameter pp, a private key sk, and an individual ciphertext $\mathsf{ct} \in \mathcal{C}_\mathsf{s}$, it outputs a message $\mathsf{m} \in \mathcal{M}$ or a symbol $\perp$ to indicate decryption failure.

Correctness. We adopt the correctness definition of mmPKE in [6]. Let $\zeta : \mathbb{N} \to [0, 1]$. We say an mmPKE scheme is ζ-correct, if for all $\lambda, N \in \mathbb{N}$ and $i \in [N]$, message $\mathsf{m}_i \in \mathcal{M}$, the following probability is at most $\zeta(\lambda)$,

$$\Pr\left[\begin{array}{c} \exists i \in [N] : \\ \mathsf{mmDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct}_i) \neq \mathsf{m}_i \end{array} \;\middle|\; \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{mmSetup}(1^\lambda, N); \\ \forall i \in [N] : (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{mmKGen}(\mathsf{pp}); \\ \mathbf{ct} \leftarrow \mathsf{mmEnc}(\mathsf{pp}, (\mathsf{pk}_i)_{i \in [N]}, (\mathsf{m}_i)_{i \in [N]}); \\ \mathsf{ct}_i \leftarrow \mathsf{mmExt}(\mathsf{pp}, i, \mathbf{ct}) \end{array} \right].$$

Security. Following [10], we formalize the security model for mmPKE. In contrast to the model in [6], our definition captures *full CPA (or CCA)* security. Briefly, we do not impose the restriction that the two challenge message vectors must have identical structures.

Let mmPKE be an mmPKE scheme, let N, λ be integers. We define the $\mathsf{mmIND\text{-}CPA}^{\mathsf{KOSK}}$ security game in Figure 2 and defer the remaining security models to the full version [54] due to space constraints, where we also provide a simple extension of our model to the security model in [47].

We say mmPKE is $\mathsf{mmIND\text{-}CPA}^{\mathsf{KOSK}}$ secure if for all PPT adversary $\mathcal{A}$, the following advantage $\mathsf{Adv}^{\mathsf{mmIND\text{-}CPA}^{\mathsf{KOSK}}}_{\mathsf{mmPKE},N,\mathcal{A}}(\lambda)$ is negligible with λ,

$$\left| \Pr[\mathsf{GAME}^{\mathsf{mmIND\text{-}CPA}^{\mathsf{KOSK}}}_{\mathsf{mmPKE},N,\mathcal{A}}(\lambda) = 1] - \frac{1}{2} \right|.$$

We say $\mathcal{A}$ wins if the game outputs 1.

4 Extended Reproducible Public Key Encryption

In this section, we provide the formal definition of XR-PKE, significantly extending on reproducible PKE in [10], and then show how it can be used to build an mmPKE.

Game $\mathsf{GAME}^{\mathsf{mmIND\text{-}CPA}^{\mathsf{KOSK}}}_{\mathsf{mmPKE},N,\mathcal{A}}(\lambda)$

$(\mathcal{A}_0,\mathcal{A}_1,\mathcal{A}_2) \leftarrow \mathcal{A}$
$\mathsf{pp} \leftarrow \mathsf{mmSetup}(1^\lambda, N)$
$(\ell, \mathsf{st}) \leftarrow \mathcal{A}_0(\mathsf{pp})$
$\forall i \in [\ell]$, $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{mmKGen}(\mathsf{pp})$
$((\mathsf{m}_i^0, \mathsf{m}_i^1)_{i\in[\ell]}, (\mathsf{m}_i)_{i\in[\ell:N]}, (\mathsf{pk}_i, \mathsf{sk}_i)_{i\in[\ell:N]}, \mathsf{st}) \leftarrow \mathcal{A}_1((\mathsf{pk}_i)_{i\in[\ell]}, \mathsf{st})$
req: $\forall i \in [\ell]$, $|\mathsf{m}_i^0| = |\mathsf{m}_i^1|$
req: $\forall i \in [\ell : N]$, $(\mathsf{pk}_i, \mathsf{sk}_i) \in \mathcal{K}$
$b \leftarrow \{0,1\}$
$\mathbf{ct} \leftarrow \mathsf{mmEnc}(\mathsf{pp}, (\mathsf{pk}_i)_{i\in[N]}, (\mathsf{m}_i^b)_{i\in[\ell]}, (\mathsf{m}_i)_{i\in[\ell:N]})$
$b' \leftarrow \mathcal{A}_2(\mathbf{ct}, \mathsf{st})$
return $[b = b']$

Figure 2: The $\mathsf{mmIND\text{-}CPA}^{\mathsf{KOSK}}$ security game for mmPKE.

Definition 4.1 (XR-PKE)**.** A (decomposable) XR-PKE with a public-private key space $\mathcal{K}$, a message space $\mathcal{M}$, two randomness distributions $(\mathcal{D}_\mathsf{i}, \mathcal{D}_\mathsf{d})$ for key-independent/key-dependent parts, respectively, and a ciphertext space $\mathcal{C}_\mathsf{s}$ consists of the following algorithms:

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, N)$: On input a security parameter 1^λ and a reproducibility count N, it outputs a public parameter pp.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$: On input a public parameter pp, it outputs a public-private key pair $(\mathsf{pk}, \mathsf{sk}) \in \mathcal{K}$.
- $\mathsf{ct} := (\mathsf{ct}_0, \widehat{\mathsf{ct}}) \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, \mathsf{m}; \mathsf{r}_0, \hat{\mathsf{r}})$: On input a public parameter pp, a public key pk, a messages m, two randomnesses $(\mathsf{r}_0, \hat{\mathsf{r}})$, it can be split into two algorithms:
  - $\mathsf{ct}_0 \leftarrow \mathsf{Enc}^{\mathsf{i}}(\mathsf{pp}; \mathsf{r}_0)$: On input a public parameter pp, and a randomness r_0 sampled from the distribution $\mathsf{r}_0 \leftarrow \mathcal{D}_\mathsf{i}$, it outputs a public-key-*independent* ciphertext ct_0.
  - $\widehat{\mathsf{ct}} \leftarrow \mathsf{Enc}^{\mathsf{d}}(\mathsf{pp}, \mathsf{pk}, \mathsf{m}; \mathsf{r}_0, \hat{\mathsf{r}})$: On input a public parameter pp, a public key pk, a message $\mathsf{m} \in \mathcal{M}$, and randomness r_0, $\hat{\mathsf{r}}$ where the latter is sampled from distribution $\hat{\mathsf{r}} \leftarrow \mathcal{D}_\mathsf{d}$ independently, it outputs a public-key-*dependent* ciphertext $\widehat{\mathsf{ct}}$.
- $\mathsf{m}/\perp \leftarrow \mathsf{Dec}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct})$: On input a public parameter pp, a private key sk, and a ciphertext $\mathsf{ct} \in \mathcal{C}_\mathsf{s}$, it outputs a message $\mathsf{m} \in \mathcal{M}$ or a symbol $\perp$ to indicate decryption failure.
- $(\mathsf{h}_i)_{i\in[N]} \leftarrow \mathsf{HintGen}(\mathsf{r}_0, (\mathsf{pk}_i, \mathsf{sk}_i)_{i\in[N]}, (\hat{\mathsf{r}}_i)_{i\in[N]})$: On input a randomness r_0 sampled from the distribution $\mathsf{r}_0 \leftarrow \mathcal{D}_\mathsf{i}$, N public-private key pairs $(\mathsf{pk}_i, \mathsf{sk}_i)_{i\in[N]} \in \mathcal{K}$, and N randomnesses $(\hat{\mathsf{r}}_i)_{i\in[N]}$ where each of them is sampled from the distribution $\hat{\mathsf{r}}_i \leftarrow \mathcal{D}_\mathsf{d}$ independently, it outputs N hints $(\mathsf{h}_i)_{i\in[N]}$.
- $\mathsf{ct}'/\perp \leftarrow \mathsf{Rep}(\mathsf{ct}, \mathsf{m}', \mathsf{pk}', \mathsf{sk}', \mathsf{h}')$: On input a ciphertext $\mathsf{ct} \in \mathcal{C}_\mathsf{s}$, a message $\mathsf{m}' \in \mathcal{M}$, a public-private key pair $(\mathsf{pk}', \mathsf{sk}') \in \mathcal{K}$, and an associated hint h', it outputs a reproduced ciphertext ct' or a symbol $\perp$ to indicate reproducibility failure.

Correctness. Let $\zeta : \mathbb{N} \to [0,1]$. We say a XR-PKE scheme is ζ-correct, if for all $\lambda, N \in \mathbb{N}^+$, the following probability is at most $\zeta(\lambda)$,

$$\Pr\left[\mathsf{Dec}(\mathsf{pp},\mathsf{sk},\mathsf{ct}) \neq \mathsf{m} \;\middle|\; \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, N); \\ (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KGen}(\mathsf{pp}), \mathsf{m} \leftarrow \mathcal{M}; \\ (\mathsf{r}_0, \hat{\mathsf{r}}) \leftarrow \mathcal{D}_\mathsf{i} \times \mathcal{D}_\mathsf{d}; \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp},\mathsf{pk},\mathsf{m};\mathsf{r}_0,\hat{\mathsf{r}}) \end{array}\right].$$

Extended Reproducibility. We first define extended reproducibility game in Figure 3. We say that PKE is *extended reproducible* if for any $\lambda, N \in \mathbb{N}^+$, there exists PPT algorithms HintGen and Rep, called *hint-generation* algorithm and *reproduction* algorithm, respectively, such that $\mathsf{Game}^{\mathsf{ext\text{-}repr}}_{\mathsf{PKE},\mathsf{Rep},N}(\lambda)$ always outputs 1. More precisely, the probability of $\Pr[\mathsf{Game}^{\mathsf{ext\text{-}repr}}_{\mathsf{PKE},\mathsf{Rep},N}(\lambda) = 1] = 1$ holds.

Game $\mathsf{Game}^{\mathsf{ext\text{-}repr}}_{\mathsf{PKE},\mathsf{Rep},N}(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, N)$
$(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{KGen}(\mathsf{pp})$
$\mathsf{m}^* \leftarrow \mathcal{M}$
$(\mathsf{r}_0, \hat{\mathsf{r}}^*) \leftarrow \mathcal{D}_\mathsf{i} \times \mathcal{D}_\mathsf{d}$
$\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{pk}^*, \mathsf{m}^*, \mathsf{r}_0, \hat{\mathsf{r}}^*)$
for all $i \in [N]$
  $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KGen}(\mathsf{pp})$
  $\mathsf{m}_i \leftarrow \mathcal{M}$
  $\hat{\mathsf{r}}_i \leftarrow \mathcal{D}_\mathsf{d}$
end for
$(\mathsf{h}_i)_{i\in[N]} \leftarrow \mathsf{HintGen}(\mathsf{r}_0, (\mathsf{pk}_i, \mathsf{sk}_i)_{i\in[N]}, (\hat{\mathsf{r}}_i)_{i\in[N]})$
if $\forall\ i \in [N]$, $\mathsf{Enc}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{m}_i; \mathsf{r}_0, \hat{\mathsf{r}}_i) = \mathsf{Rep}(\mathsf{ct}^*, \mathsf{m}_i, \mathsf{pk}_i, \mathsf{sk}_i, \mathsf{h}_i)$
then
  return 1
else
  return 0
end if

Figure 3: The extended reproducibility game for XR-PKE.

Security. To fit the property of extended reproducibility, we modify the IND-ATK security of standard PKE to $\mathsf{IND\text{-}ATK}^{\mathsf{XR}}$ for $\mathsf{ATK} = \{\mathsf{CPA}, \mathsf{CCA}\}$. Roughly speaking, we say an XR-PKE is secure if the hints generated by HintGen would not help the adversary to break the security of the challenge ciphertext.

Specifically, let PKE be an XR-PKE and we provide the security game of PKE in Figure 4. With the game $\mathsf{Game}^{\mathsf{IND\text{-}ATK}^{\mathsf{XR}}}_{\mathsf{PKE},N,b,\mathcal{A}}(\lambda)$, we say PKE is $\mathsf{IND\text{-}ATK}^{\mathsf{XR}}$ secure if for all PPT adversary $\mathcal{A}$, the following advantage $\mathsf{Adv}^{\mathsf{IND\text{-}ATK}^{\mathsf{XR}}}_{\mathsf{PKE},N,\mathcal{A}}(\lambda)$ is negligible with λ,

$$\left|\Pr\left[\mathsf{GAME}^{\mathsf{IND\text{-}ATK}^{\mathsf{XR}}}_{\mathsf{PKE},N,0,\mathcal{A}}(\lambda) = 0\right] - \Pr\left[\mathsf{GAME}^{\mathsf{IND\text{-}ATK}^{\mathsf{XR}}}_{\mathsf{PKE},N,1,\mathcal{A}}(\lambda) = 0\right]\right|.$$

Remark 4.2. Our definition of XR-PKE actually captures the case of original reproducible PKE in [10]. When describing the original reproducible PKE, we can make the hint generation algorithm HintGen output nothing, i.e., set each of the output hints h_i as an empty symbol $\perp$.

Game $\mathsf{GAME}^{\mathsf{IND\text{-}ATK}^{\mathsf{XR}}}_{\mathsf{PKE},N,b,\mathcal{A}}(\lambda)$ for $\mathsf{ATK}=\{\mathsf{CPA},\mathsf{CCA}\}$

$(\mathcal{A}_0,\mathcal{A}_1,\mathcal{A}_2)\leftarrow\mathcal{A}$
$\mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda,N)$
$(\mathsf{pk}^*,\mathsf{sk}^*)\leftarrow\mathsf{KGen}(\mathsf{pp})$
if $\mathsf{ATK}=\mathsf{CPA}$ **do** $(\mathsf{m}_0^*,\mathsf{m}_1^*,\mathsf{st})\leftarrow\mathcal{A}_0(\mathsf{pp},\mathsf{pk}^*)$
if $\mathsf{ATK}=\mathsf{CCA}$ **do** $(\mathsf{m}_0^*,\mathsf{m}_1^*,\mathsf{st})\leftarrow\mathcal{A}_0^{\mathsf{Dec}_0}(\mathsf{pp},\mathsf{pk}^*)$
req: $|\mathsf{m}_0^*|=|\mathsf{m}_1^*|$
$(\mathsf{r}_0,\hat{\mathsf{r}}^*)\leftarrow\mathcal{D}_{\mathsf{t}}\times\mathcal{D}_{\mathsf{d}}$
$\mathsf{ct}^*\leftarrow\mathsf{Enc}(\mathsf{pp},\mathsf{pk}^*,\mathsf{m}_b^*;\mathsf{r}_0,\hat{\mathsf{r}}^*)$
if $\mathsf{ATK}=\mathsf{CPA}$ **do** $((\mathsf{pk}_i,\mathsf{sk}_i)_{i\in[N]},\mathsf{st})\leftarrow\mathcal{A}_1(\mathsf{ct}^*,\mathsf{st})$
if $\mathsf{ATK}=\mathsf{CCA}$ **do** $((\mathsf{pk}_i,\mathsf{sk}_i)_{i\in[N]},\mathsf{st})\leftarrow\mathcal{A}_1^{\mathsf{Dec}_1}(\mathsf{ct}^*,\mathsf{st})$
req: $\forall i\in[N],(\mathsf{pk}_i,\mathsf{sk}_i)\in\mathcal{K}$
$\forall i\in[N]:\hat{\mathsf{r}}_i\leftarrow\mathcal{D}_{\mathsf{d}}$
$(\mathsf{h}_i)_{i\in[N]}\leftarrow\mathsf{HintGen}(\mathsf{r}_0,(\mathsf{pk}_i,\mathsf{sk}_i)_{i\in[N]},(\hat{\mathsf{r}}_i)_{i\in[N]})$
if $\mathsf{ATK}=\mathsf{CPA}$ **do** $b'\leftarrow\mathcal{A}_2((\mathsf{h}_i)_{i\in[N]},\mathsf{st})$
if $\mathsf{ATK}=\mathsf{CCA}$ **do** $b'\leftarrow\mathcal{A}_2^{\mathsf{Dec}_1}((\mathsf{h}_i)_{i\in[N]},\mathsf{st})$
return b'

Oracle $\mathsf{Dec}_0(\mathsf{ct})$
return $\mathsf{m}\leftarrow\mathsf{Dec}(\mathsf{pp},\mathsf{sk}^*,\mathsf{ct})$

Oracle $\mathsf{Dec}_1(\mathsf{ct})$
req: $\mathsf{ct}\neq\mathsf{ct}^*$
return $\mathsf{m}\leftarrow\mathsf{Dec}(\mathsf{pp},\mathsf{sk}^*,\mathsf{ct})$

Figure 4: The $\mathsf{IND\text{-}ATK}^{\mathsf{XR}}$ security game for XR-PKE with $\mathsf{ATK}=\{\mathsf{CPA},\mathsf{CCA}\}$.

4.1 Generic Construction of mmPKE from XR-PKE

In this subsection, we show the generic construction of mmPKE from XR-PKE.

Construction 4.3 (XR-PKE→mmPKE Compiler)**.** For $\mathsf{ATK}\in\{\mathsf{CPA},\mathsf{CCA}\}$, let $\mathsf{PKE}=(\mathsf{Setup},\mathsf{KGen},\mathsf{Enc}=(\mathsf{Enc}^{\mathsf{i}},\mathsf{Enc}^{\mathsf{d}}),\mathsf{Dec})$ be a (decomposable) $\mathsf{IND\text{-}ATK}^{\mathsf{XR}}$ secure XR-PKE with public-private key space $\mathcal{K}$ and two randomness distributions $(\mathcal{D}_{\mathsf{t}},\mathcal{D}_{\mathsf{d}})$ for key-independent/key-dependent parts, respectively. Let Compress, Decompress be the compression and decompression algorithms which can be ignored if there does not exist suitable algorithms. Our compiler $\mathsf{Comp}^{\mathsf{mmPKE}}[\mathsf{PKE}]$ is defined in Figure 5, which outputs an $\mathsf{mmIND\text{-}ATK}^{\mathsf{KOSK}}$ secure mmPKE.

Correctness. It is not difficult to see that correctness of our Construction 4.3 follows if the input PKE is correct and the output by decompression algorithm Decompress can still be successfully decrypted with overwhelming probability.

Security. Some intuitive discussion on the security reduction was provided in Technical Overview (Section 2). At a high level, since the provided hints do not help the adversary (or reduction) to break the security of the underlying XR-PKE, we can establish the security of its corresponding mmPKE. Formally, we have the following theorem, the proofs are deferred to the full version [54] due to space constraints.

Theorem 4.4 (Security)**.** *For* $\mathsf{ATK}\in\{\mathsf{CPA},\mathsf{CCA}\}$*, if* PKE *is* $\mathsf{IND\text{-}ATK}^{\mathsf{XR}}$ *secure and satisfies extended reproducibility, our* $\mathsf{mmPKE}\leftarrow\mathsf{Comp}^{\mathsf{mmPKE}}[\mathsf{PKE}]$ *output by Construction 4.3 is* $\mathsf{mmIND\text{-}ATK}^{\mathsf{KOSK}}$ *secure.*

Remark 4.5 (Removing the KOSK Assumption). We also present a generic KOSK compiler that can eliminate the reliance on the KOSK assumption for both our post-quantum and traditional mmPKE schemes, e.g., [10, 11, 35]. Detailed construction with its formal proof, and an instantiation for our lattice-based mmPKE are given in Appendix B.

5 Lattice-Based XR-PKE

In this section, we construct lattice-based XR-PKE which can be used to build efficient mmPKE/KEM via the compiler introduced in the last section.

Our constructions are based on the Matrix Hint-MLWE assumption [26], a variant of the MLWE assumption generalized from the Hint-MLWE assumption [34] and can be reduced from the standard MLWE via appropriate parameters. Specifically, we first present a more general Matrix Hint-MLWE along with our refined reduction, followed by an instantiation for our XR-PKE. We then detail the constructions. Finally, we specify the parameter choices and present a theoretical analysis of our mmPKE, comparing it with the trivial solution with Kyber.

5.1 Refined Matrix Hint-MLWE Assumption

In this subsection, we generalize Matrix Hint-MLWE to a *non-square* version, refine its reduction from standard MLWE by introducing a *sampleability condition* missing in prior works, and then derive a new parameter setting. Next, we provide an instantiation of Matrix Hint-MLWE to establish the CPA security of our XR-PKE introduced in the following subsection. We start by generalizing the definition of Matrix Hint-MLWE in [26].

Definition 5.1 (Matrix Hint-MLWE, generalized [26])**.** Let m, n, ℓ be positive integers. Let $\mathcal{S},\chi_0,\chi_1$ be distributions over $\mathcal{R}^{\ell\times(m+n)},\mathcal{R}^{m+n},\mathcal{R}^{\ell}$, respectively. The Matrix Hint-MLWE, denoted by $\mathsf{MatrixHint\text{-}MLWE}^{\ell,\chi_1,\mathcal{S}}_{\mathcal{R},m,n,q,\chi_0}$, asks a PPT adversary $\mathcal{A}$ to distinguish the following two cases:

1. $(\mathbf{A},[\mathbf{I}_m|\mathbf{A}]\mathbf{r},\mathbf{R},\mathbf{h})$ for $\mathbf{A}\leftarrow\mathcal{U}(\mathcal{R}_q^{m\times n})$, $\mathbf{r}\leftarrow\chi_0$, $\mathbf{y}\leftarrow\chi_1$, $\mathbf{R}\leftarrow\mathcal{S}$, and $\mathbf{h}:=\mathbf{R}\mathbf{r}+\mathbf{y}$.
2. $(\mathbf{A},\mathbf{u},\mathbf{R},\mathbf{h})$ for $\mathbf{A}\leftarrow\mathcal{U}(\mathcal{R}_q^{m\times n})$, $\mathbf{u}\leftarrow\mathcal{U}(\mathcal{R}_q^{m})$, $\mathbf{r}\leftarrow\chi_0$, $\mathbf{y}\leftarrow\chi_1$, $\mathbf{R}\leftarrow\mathcal{S}$, and $\mathbf{h}:=\mathbf{R}\mathbf{r}+\mathbf{y}$.

We say $\mathsf{MatrixHint\text{-}MLWE}^{\ell,\chi_1,\mathcal{S}}_{\mathcal{R},m,n,q,\chi_0}$ is hard if for any PPT adversary $\mathcal{A}$, the following advantage of $\mathcal{A}$ is negligible in λ,

$$\mathsf{Adv}^{\mathsf{MatrixHint\text{-}MLWE}}_{\mathsf{para},\mathcal{A}}(\lambda):=\left|\Pr\left[b=1\;\middle|\;\begin{array}{c}\mathbf{A}\leftarrow\mathcal{U}(\mathcal{R}_q^{m\times n}),\mathbf{r}\leftarrow\chi_0,\mathbf{y}\leftarrow\chi_1,\\ \mathbf{R}\leftarrow\mathcal{S},\mathbf{h}:=\mathbf{R}\mathbf{r}+\mathbf{y},\\ b\leftarrow\mathcal{A}(\mathbf{A},[\mathbf{I}_m|\mathbf{A}]\mathbf{r},\mathbf{R},\mathbf{h})\end{array}\right]\right.$$
$$\left.-\Pr\left[b=1\;\middle|\;\begin{array}{c}\mathbf{A}\leftarrow\mathcal{U}(\mathcal{R}_q^{m\times n}),\\ \mathbf{r}\leftarrow\chi_0,\mathbf{y}\leftarrow\chi_1,\mathbf{R}\leftarrow\mathcal{S},\\ \mathbf{h}:=\mathbf{R}\mathbf{r}+\mathbf{y},\mathbf{u}\leftarrow\mathcal{U}(\mathcal{R}_q^{m}),\\ b\leftarrow\mathcal{A}(\mathbf{A},\mathbf{u},\mathbf{R},\mathbf{h})\end{array}\right]\right|$$

$\underline{\mathsf{mmSetup}(1^\lambda, N)}$
Input:
- security parameter 1^λ
- recipient number N

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, N)$
return pp

$\underline{\mathsf{mmKGen}(\mathsf{pp})}$
Input: public parameter pp
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$
return $(\mathsf{pk}, \mathsf{sk})$

$\underline{\mathsf{mmEnc}(\mathsf{pp}, (\mathsf{pk}_i)_{i\in[N]}, (\mathsf{m}_i)_{i\in[N]})}$
Input:
- public parameter pp
- a set of public keys $(\mathsf{pk}_i)_{i\in[N]}$
- a set of messages $(\mathsf{m}_i)_{i\in[N]}$

$\mathsf{r}_0 \leftarrow \mathcal{D}_{\mathsf{r}}$
$\mathsf{ct}_0 \leftarrow \mathsf{Enc}^{\mathsf{i}}(\mathsf{pp}; \mathsf{r}_0)$
$\bar{\mathsf{ct}}_0 \leftarrow \mathsf{Compress}(\mathsf{ct}_0)$
for $i \in [N]$
  $\hat{\mathsf{r}}_i \leftarrow \mathcal{D}_{\mathsf{d}}$
  $\hat{\mathsf{ct}}_i \leftarrow \mathsf{Enc}^{\mathsf{d}}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{m}_i; \mathsf{r}_0, \hat{\mathsf{r}}_i)$
end for
$\mathbf{ct} := (\bar{\mathsf{ct}}_0, (\hat{\mathsf{ct}}_i)_{i\in[N]})$
return ct

$\underline{\mathsf{mmExt}(\mathbf{ct}, k)}$
Input: multi-recipient ciphertext ct, index k
req: $k \in [N]$
$(\bar{\mathsf{ct}}_0, (\hat{\mathsf{ct}}_i)_{i\in[N]}) \leftarrow \mathbf{ct}$
return $\mathsf{ct}_k := (\bar{\mathsf{ct}}_0, \hat{\mathsf{ct}}_k)$

$\underline{\mathsf{mmDec}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct})}$
Input:
- public parameter pp
- private key sk
- individual ciphertext ct

$(\bar{\mathsf{ct}}_0, \hat{\mathsf{ct}}) \leftarrow \mathsf{ct}$
$\mathsf{ct}'_0 \leftarrow \mathsf{Decompress}(\bar{\mathsf{ct}}_0)$
$\mathsf{m} \leftarrow \mathsf{Dec}(\mathsf{pp}, \mathsf{sk}, (\mathsf{ct}'_0, \hat{\mathsf{ct}}))$
return m

Figure 5: Generic constructions of $\mathsf{mmIND\text{-}ATK}^{\mathsf{KOSK}}$ mmPKE output by the compiler $\mathsf{Comp}^{\mathsf{mmPKE}}[\mathsf{PKE}]$ for $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CCA}\}$.

where $\mathsf{para} = ((\mathcal{R}, m, n, q, \chi_0), (\ell, \chi_1, \mathcal{S}))$.

We slightly adapt the notation towards our needs. In [26], the public matrix $\mathbf{R}$ is defined only for the square case (i.e., $\ell = m + n$). Here, we relax this requirement and generalize $\mathbf{R}$ to a rectangular form with ℓ not necessarily equal to $m + n$.

Theorem 5.2 (Hardness of Matrix Hint-MLWE)**.** *Let m, n, q, ℓ be positive integers. Let $\mathcal{S}$ be a distribution over $\mathcal{R}^{\ell\times(m+n)}$. Let $B > 0$ be a real number such that $\|\bar{R}\|^2 \leqslant B$ where $\bar{R} := \Gamma(\mathbf{R})$ for all possible $\mathbf{R} \leftarrow \mathcal{S}$. Let $\sigma_0, \sigma_1, \sigma, \delta > 0$ be real numbers. Let Σ_1, $\Sigma_{\mathbf{y}}$ be a positive definite symmetric matrices over $\mathbb{R}^{(m+n)d\times(m+n)d}$ and $\mathbb{R}^{\ell d\times\ell d}$, respectively, such that $\|\Sigma_1^{-1}\| \leqslant \frac{1}{\sigma_0^2}$ and $\|\Sigma_{\mathbf{y}}^{-1}\| \leqslant \frac{1}{\sigma_1^2}$. Let $\chi_0 := \mathcal{D}_{\mathbb{Z}^{(m+n)d}, \sqrt{\Sigma_1}}$, $\chi_1 := \mathcal{D}_{\mathbb{Z}^{\ell d}, \sqrt{\Sigma_{\mathbf{y}}}}$, $\chi := \mathcal{D}_{\mathbb{Z}^{(m+n)d}, \sigma}$ be distributions over $\mathcal{R}^{m+n}, \mathcal{R}^{\ell}, \mathcal{R}^{m+n}$, respectively. There exists an efficient reduction from $\mathsf{MLWE}_{\mathcal{R},m,n,q,\chi}$ to $\mathsf{MatrixHint\text{-}MLWE}^{\ell,\chi_1,\mathcal{S}}_{\mathcal{R},m,n,q,\chi_0}$ that reduces the advantage by at most 2ε, if the sampleability condition*

$$\frac{1}{(1+\delta)\sigma^2 + \delta_0} \geqslant \frac{1}{\sigma_0^2} + \frac{B}{\sigma_1^2} \tag{5}$$

where $\delta_0 := \sqrt{\frac{\ln(2(m+n)d)+4}{\pi}}$, and the convolution condition

$$\sigma \geqslant \sqrt{1 + 1/\delta} \cdot \eta_\varepsilon(\mathbb{Z}^{(m+n)d}) \tag{6}$$

are satisfied.

Specifically, for any PPT adversary $\mathcal{A}$ against the $\mathsf{MatrixHint\text{-}MLWE}^{\ell,\chi_1,\mathcal{S}}_{\mathcal{R},m,n,q,\chi_0}$ assumption, there exists a PPT adversary $\mathcal{B}$ against the $\mathsf{MLWE}_{\mathcal{R},m,n,q,\chi}$ assumption, such that

$$\mathsf{Adv}^{\mathsf{MatrixHint\text{-}MLWE}}_{\mathsf{para}_0,\mathcal{A}}(\lambda) \leqslant \mathsf{Adv}^{\mathsf{MLWE}}_{\mathsf{para}_1,\mathcal{B}}(\lambda) + 2\varepsilon$$

where $\mathsf{para}_0 = ((\mathcal{R}, m, n, q, \chi_0), (\ell, \chi_1, \mathcal{S}))$ and $\mathsf{para}_1 = (\mathcal{R}, m, n, q, \chi)$.

The proof is deferred to the full version [54] due to space constraints, which presents a refined version of [26].

Matrix Hint-MLWE Instantiation for XR-PKE. We first define the distribution $\mathcal{S}$ such that matrix $\mathbf{R}$ can be sampled as follows,

$$\mathbf{R} := \begin{pmatrix} 0 & -\mathbf{s}_0^\top & \mathbf{e}_0^\top \\ \vdots & \vdots & \vdots \\ 0 & -\mathbf{s}_{\ell-1}^\top & \mathbf{e}_{\ell-1}^\top \end{pmatrix} \in \mathcal{R}^{\ell\times(1+m+n)} \tag{7}$$

where $\mathbf{s}_i \leftarrow \mathcal{U}(\mathbb{S}_{\mathsf{v}}^n)$, $\mathbf{e}_i \leftarrow \mathcal{U}(\mathbb{S}_{\mathsf{v}}^m)$ for each $i \in [\ell]$.

Then, we transfer the polynomial matrix $\mathbf{R}$ to its integer matrix $\bar{R} := \Gamma(\mathbf{R}) \in \mathbb{Z}^{\ell d\times(1+m+n)d}$ by substitute the polynomial elements in each vector $\mathbf{s}_i$, $\mathbf{e}_i$ by its negacyclic matrix $\Gamma(\cdot)$ as follows,

$$\bar{R} := \begin{pmatrix} 0 & \Gamma(-\mathbf{s}_0) & \Gamma(\mathbf{e}_0) \\ \vdots & \vdots & \vdots \\ 0 & \Gamma(-\mathbf{s}_{\ell-1}) & \Gamma(\mathbf{e}_{\ell-1}) \end{pmatrix}.$$

To bound the norm of the matrix $\bar{R}$, we use the inequality $\|\bar{R}\| \leqslant \sqrt{\|\bar{R}\|_1 \cdot \|\bar{R}\|_\infty}$, where $\|\bar{R}\|_1 \leqslant \mathsf{v}\ell d$ and $\|\bar{R}\|_\infty \leqslant \mathsf{v}(m+n)d$. Thus, $\|\bar{R}\|^2 \leqslant B$, where

$$B := \ell(m+n)(d\mathsf{v})^2 \tag{8}$$

Last, we define the matrix $\Sigma_1 \in \mathbb{R}^{(1+m+n)d\times(1+m+n)d}$ and $\Sigma_{\mathbf{y}} \in \mathbb{R}^{\ell d\times\ell d}$ below,

$$\Sigma_1 := \begin{pmatrix} \sigma_1 I_d & 0 \\ 0 & \sigma_0 I_{(m+n)d} \end{pmatrix}, \quad \Sigma_{\mathbf{y}} := \sigma_1 I_{\ell d}. \tag{9}$$

We set $\sigma_1 \geqslant \sigma_0$ so that we have $\|\Sigma_1^{-1}\| = \max(\frac{1}{\sigma_0^2}, \frac{1}{\sigma_1^2}) \leqslant \frac{1}{\sigma_0^2}$ and $\|\Sigma_{\mathbf{y}}^{-1}\| \leqslant 1/\sigma_1^2$.

5.2 Construction of XR-PKE

In this subsection, we present the lattice-based construction of XR-PKE. At a high level, we leverage the decryption error as a hint to enable ciphertext reproducibility. To this end, we sample the ciphertext randomness from carefully chosen Gaussian distributions, allowing us to reduce the security of our XR-PKE scheme to the hardness of the Matrix Hint-MLWE problem.

Construction 5.3 (XR-PKE from Lattices)**.** Let λ be a security parameter, $m = m(\lambda)$, $n = n(\lambda)$, $d = d(\lambda)$, $q = q(\lambda)$, $N = N(\lambda)$, $\nu = \nu(\lambda)$ be positive integers. Let $\sigma_0 = \sigma_0(\lambda)$, $\sigma_1 = \sigma_1(\lambda)$ be Gaussian width parameters. For the message space $\mathcal{M} = \{0,1\}^d$, the detailed construction is shown in Figure 6. We summarize the notations in Table 2.

Table 2: Summary of main notations used in our lattice-based XR-PKE/KEM.

Notation	Description
λ	security parameter
ζ	correctness parameter
N	# of recipients
m,n	# of rows of $\mathbf{A}$, # of columns of $\mathbf{A}$
q	system modulus
d	ring dimension of $\mathcal{R} = \mathbb{Z}[X]/(X^d+1)$
ℓ	dimension of hint vector $\mathbf{h}$ in Matrix Hint-MLWE
ν	ℓ_∞-norm bound on private key $(\mathbf{s}_i, \mathbf{e}_i)$
$\bar{\nu}$	support size $\bar{\nu} \leqslant 2\nu+1$ of private key $(\mathbf{s}_i, \mathbf{e}_i)$
$\bar{\chi}$	private key distribution
σ_0	Gaussian width of $(\mathbf{r}, \mathbf{e}_u)$ in the ciphertext
χ_1, σ_1	distribution and Gaussian width of y in the ciphertext
χ, σ	distribution and Gaussian width of secret in MLWE (hardness equal to Matrix Hint-MLWE)
χ_0, Σ_1	distribution and covariance matrix of secret in Matrix Hint-MLWE
B	square of matrix 2-norm bound on $\bar{R} := \Gamma(\mathbf{R})$
$\mathcal{S}$	distribution of $\mathbf{R}$
d_u	# of bits of each coefficient in key-independent ciphertext
d_v	# of bits of each coefficient in key-dependent ciphertext

Extended Reproducibility. We show the extended reproducibility of our construction as follows. The proof is deferred to the full version [54] due to space constraints.

Theorem 5.4 (Extended Reproducibility)**.** *For any positive integer N, our* PKE *in Construction 5.3 is extended reproducible. More precisely, for the extended reproducible game in Figure 3, the probability of* $\Pr[\mathsf{Game}^{\mathsf{ext\text{-}repr}}_{\mathsf{PKE},\mathsf{Rep},N}(\lambda) = 1] = 1$ *holds.*

Correctness. We set $\mathsf{Compress}(x) = \lfloor x \mod q \rceil_{2^{d_u}}$ and $\mathsf{Decompress}(x) = \lfloor x \mod 2^{d_u} \rceil_q$. Here, we mainly consider the case that the (key-independent) ciphertext is compressed and then decompressed before the decryption, as done in mmPKE compiler of Construction 4.3.

We show the correctness of our construction as follows. We will select parameters in Section 5.3 to make our construction ζ-correct with $\zeta \leqslant 2^{-128}$. The proof is deferred to the full version [54] due to space constraints.

Theorem 5.5 (Correctness)**.** *Let $\mathbf{e}, \mathbf{s}, \mathbf{r}, \mathbf{e}_u, y$ be random variables that have the corresponding distribution as in Construction 5.3. Denote ζ as*

$$\Pr[\ \|\langle \mathbf{e}, \mathbf{r}\rangle + y - \langle \mathbf{s}, \mathbf{e}_u\rangle - c_v + \langle \mathbf{s}, \mathbf{c}_u\rangle\|_\infty \geqslant \lfloor q/4 \rceil\ ]$$

where $\mathbf{c}_u := \mathbf{c} - \lfloor \lfloor \mathbf{c} \bmod q \rceil_{2^{d_u}} \rceil_q \in \mathcal{R}^m$, and $c_v := c - \lfloor \lfloor c \bmod q \rceil_{2^{d_v}} \rceil_q \in \mathcal{R}$. We say our Construction 5.3 is ζ-correct.

Security. We show that our Construction 5.3 is $\mathsf{IND\text{-}CPA}^{\mathsf{XR}}$ secure if the MLWE assumption and the Matrix Hint-MLWE assumption are hard. The proof is deferred to the full version [54] due to space constraints.

Theorem 5.6 (Security)**.** *Let m,n,d,q,N,ν be positive integers parameters. Let $\sigma, \sigma_0, \sigma_1$ be Gaussian width parameters. Let the positive real matrices Σ_1 and $\Sigma_{\mathbf{y}}$ be as Equation (9). Let the distribution $\mathcal{S}$ and the bound B be as Equation (7) and (8) respectively. Let the distribution $\chi_0 := \mathcal{D}_{\mathbb{Z}^{(m+n+1)d}, \sqrt{\Sigma_1}}$, $\chi_1 := \mathcal{D}_{\mathbb{Z}^{Nd}, \sqrt{\Sigma_{\mathbf{y}}}}$, $\bar{\chi} := \mathcal{U}(\mathbb{S}_\nu)$. Suppose Equation (5) and (6) hold.*

Our PKE *in Construction 5.3 is* $\mathsf{IND\text{-}CPA}^{\mathsf{XR}}$ *secure under the* $\mathsf{MLWE}_{\mathcal{R},n,m,q,\bar{\chi}}$ *and* $\mathsf{MatrixHint\text{-}MLWE}^{N,\chi_1,\mathcal{S}}_{\mathcal{R},m+1,n,q,\chi_0}$ *assumptions. More precisely, for any PPT adversary $\mathcal{A}$, there exist PPT adversaries $\mathcal{B}_0$, $\mathcal{B}_1$ against MLWE assumption and Matrix Hint-MLWE assumption, such that*

$$\mathsf{Adv}^{\mathsf{IND\text{-}CPA}^{\mathsf{XR}}}_{\mathsf{PKE},N,\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathsf{MLWE}}_{\mathsf{para}_0,\mathcal{B}_0}(\lambda) + \mathsf{Adv}^{\mathsf{MatrixHint\text{-}MLWE}}_{\mathsf{para}_1,\mathcal{B}_1}(\lambda)$$

where $\mathsf{para}_0 := (\mathcal{R}, n, m, q, \bar{\chi})$ *and* $\mathsf{para}_1 := ((\mathcal{R}, m+1, n, q, \chi_0), (N, \chi_1, \mathcal{S}))$.

Remark 5.7 (Lattice-based XR-KEM (mmKEM)). By applying the reconciliation mechanism [45] (see the full version [54]) to our XR-PKE (mmPKE), we can minimize the key-dependent ciphertext to the length of the encapsulated key (e.g., 256 bits), thus achieving an asymptotically *bandwidth-optimal* mmKEM that can be extended to an mmPKE for arbitrary-length message via a Data Encapsulation Mechanism (DEM), as in [47]. Detailed constructions are provided in Appendix A.

5.3 Parameter Setting

In this subsection, we discuss parameter selection for the above constructions. Then, we theoretically demonstrate the performance of the mmPKE/mmKEM built from our constructions, compared to the trivial solution with Kyber.

As discussed before, we need to guarantee that our lattice-based constructions of XR-PKE/KEM satisfy the follow properties:

- $\mathsf{MLWE}_{\mathcal{R},n,m,q,\bar{\chi}}$ problem is hard (at 128-bit, 192-bit, and 256-bit security).

$\underline{\mathsf{Setup}(1^\lambda, N)}$
Input:
- security parameter 1^λ
- recipient number N

$\mathbf{A} \leftarrow \mathcal{U}(\mathcal{R}_q^{m\times n})$
return $\mathsf{pp} := \mathbf{A}$

$\underline{\mathsf{KGen}(\mathsf{pp})}$
Input: $\mathsf{pp} = \mathbf{A}$
$(\mathbf{s}, \mathbf{e}) \leftarrow \mathcal{U}(\mathbb{S}_\nu^m) \times \mathcal{U}(\mathbb{S}_\nu^n)$
$\mathbf{b} := \mathbf{A}^\top \mathbf{s} + \mathbf{e}$
return $(\mathsf{pk} := \mathbf{b}, \mathsf{sk} := \mathbf{s})$

$\underline{\mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, \mathsf{m})}$
Input:
- public parameter $\mathsf{pp} = \mathbf{A}$
- public key $\mathsf{pk} = \mathbf{b}$
- message m

$\mathsf{r}_0 := (\mathbf{r}, \mathbf{e}_u) \leftarrow \mathcal{D}_{\sigma_0}^n \times \mathcal{D}_{\sigma_0}^m$
$\mathsf{ct}_0 \leftarrow \mathsf{Enc}^{\mathsf{i}}(\mathsf{pp}; \mathsf{r}_0)$
$\hat{\mathsf{r}} := y \leftarrow \mathcal{D}_{\sigma_1}$
$\widehat{\mathsf{ct}} \leftarrow \mathsf{Enc}^{\mathsf{d}}(\mathsf{pp}, \mathsf{pk}, \mathsf{m}; \mathsf{r}_0, \hat{\mathsf{r}})$
return $\mathsf{ct} := (\mathsf{ct}_0, \widehat{\mathsf{ct}})$

$\underline{\mathsf{Enc}^{\mathsf{d}}(\mathsf{pp}, \mathsf{pk}, \mathsf{m}; \mathsf{r}_0, \hat{\mathsf{r}})}$
Input:
- public parameter $\mathsf{pp} = \mathbf{A}$
- public key $\mathsf{pk} = \mathbf{b}$
- message $\mathsf{m} = m \in \{0,1\}^d$
- randomness $\mathsf{r}_0 = (\mathbf{r}, \mathbf{e}_u)$
- randomness $\hat{\mathsf{r}} = y$

$c := \langle \mathbf{b}, \mathbf{r} \rangle + y + \lfloor \frac{q}{2} \rceil \cdot m$
$u := \lfloor c \mod q \rceil_{2^{d_v}}$
return $\widehat{\mathsf{ct}} := u$

$\underline{\mathsf{Enc}^{\mathsf{i}}(\mathsf{pp}; \mathsf{r}_0)}$
Input:
- public parameter $\mathsf{pp} = \mathbf{A}$
- randomness $\mathsf{r}_0 = (\mathbf{r}, \mathbf{e}_u)$

$\mathbf{c} := \mathbf{A}\mathbf{r} + \mathbf{e}_u$
return $\mathsf{ct}_0 := \mathbf{c}$

$\underline{\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct})}$
Input:
- public parameter $\mathsf{pp} = \mathbf{A}$
- private key $\mathsf{sk} = \mathbf{s}$
- ciphertext $\mathsf{ct} = (\mathbf{c}, u)$

$u' := \lfloor u \mod 2^{d_v} \rceil_q$
$m := \lfloor u' - \langle \mathbf{c}, \mathbf{s} \rangle \mod 2^{d_u} \rceil_2$
return $\mathsf{m} := m$

$\underline{\mathsf{HintGen}(\mathsf{pp}, \mathsf{r}_0, (\mathsf{pk}_i, \mathsf{sk}_i)_{i\in[N]})}$
Input:
- public parameter $\mathsf{pp} = \mathbf{A}$
- randomness $\mathsf{r}_0 = (\mathbf{r}, \mathbf{e}_u)$
- a set of public-private key pairs $(\mathsf{pk}_i, \mathsf{sk}_i)_{i\in[N]} = (\mathbf{b}_i, \mathbf{s}_i)_{i\in[N]}$

for all $i \in [N]$
$y_i \leftarrow \mathcal{D}_{\sigma_1}$
$\mathbf{e}_i := \mathbf{b}_i - \mathbf{A}^\top \mathbf{s}_i$
$h_i := \langle \mathbf{r}, \mathbf{e}_i \rangle - \langle \mathbf{e}_u, \mathbf{s}_i \rangle + y_i$
end for
return $(h_i)_{i\in[N]}$

$\underline{\mathsf{Rep}(\mathsf{ct}, \mathsf{m}', \mathsf{pk}', \mathsf{sk}', \mathsf{h}')}$
Input:
- ciphertext $\mathsf{ct} = (\mathbf{c}, u)$
- message $\mathsf{m}' = m' \in \{0,1\}^d$
- public-private key $(\mathsf{pk}', \mathsf{sk}') = (\mathbf{b}', \mathbf{s}')$
- hint $\mathsf{h}' = h'$

$c' := \langle \mathbf{c}, \mathbf{s}' \rangle + h' + \lfloor \frac{q}{2} \rceil \cdot m'$
$u' := \lfloor c' \mod q \rceil_{2^{d_v}}$
return $\mathsf{ct}' := (\mathbf{c}, u')$

Figure 6: An IND-CPA$^{\mathsf{XR}}$ secure lattice-based XR-PKE.

- MatrixHint-MLWE$^{N,\chi_1,\mathcal{S}}_{\mathcal{R},m+1,n,q,\chi_0}$ problem is hard (at 128-bit, 192-bit, and 256-bit security).
- ζ-correctness holds with $\zeta \leqslant 2^{-128}$.

To estimate the practical hardness of MLWE problem against known attacks, we follow a strategy similar to Kyber [15] and use the Lattice Estimator (a.k.a. LWE Estimator [3]). For MatrixHint-MLWE, we follow a strategy as in the original Hint MLWE paper [26, 34] and estimate the practical hardness of the related MLWE problem. The parameters of our constructions are summarized in Table 3.

Table 3: Parameter set for our lattice-based constructions of XR-PKE and -KEM, aiming at ζ-correctness with $\zeta \leqslant 2^{-128}$.

N	$\lceil \log q \rceil$	d	m	n	$(\nu, \bar{\nu})$	(d_u, d_v)	(σ_0, σ_1)	pq-sec
2^{10}	25	256	4	4	(1,3)	(10,2)	(15.9,368459)	128
2^{10}	25	256	7	7	(1,2)	(11,2)	(15.9,488797)	192
2^{10}	25	256	9	9	(1,2)	(11,2)	(15.9,554941)	256

We now present a step-by-step procedure for selecting the parameters. First, we choose $\nu = 1$, fixing the ℓ_∞-norm of $\mathcal{S}$ and the private key. We choose ternary ($\bar{\nu} = 3$) support $\{0, \pm 1\}$ for $\mathcal{S}$ in the 128-bit parameter set, and binary ($\bar{\nu} = 2$) support $\{0,1\}$ for 192- and 256-bit parameter sets.

Second, we fix $\delta = 1$ in Theorem 5.2. Then, we need to guarantee that $2\varepsilon \leqslant 2^{-128}$ and the requirements in Equation (5) and (6) hold. By [54, Lemma A.4], we set $\sigma := \sqrt{2} \cdot \sqrt{\ln(2d(m+n)(1+1/\varepsilon))/\pi}$ so that $\sigma \geqslant \sqrt{2} \cdot \eta_\varepsilon(\mathbb{Z}^{(m+n)d})$ holds. Then, we set $\sigma_0 := 2\sqrt{\sigma^2 + \delta_0/2}$, and $\sigma_1 := 2\sqrt{B}\sqrt{\sigma^2 + \delta_0/2}$ where $\delta_0 := \sqrt{(\ln(2(m+n)d) + 4)/\pi}$ so that $\frac{1}{2\sigma^2 + \delta_0} \geqslant \frac{1}{\sigma_0^2} + \frac{B}{\sigma_1^2}$ holds. Here, we set the bound B as in Equation (8), i.e., $B := N(m+n)(d\nu)^2$.

Third, we set $n = m$ and $d = 256$. Thus, the encapsulated key space and the short message space $\mathcal{M} = \{0,1\}^{256}$ is the same as the one in Kyber.

Fourth, we pick the recipient numbers N (e.g., $N = 1024$) for usability. By [54, Lemma A.3], we can derive the tail bound of the Gaussian distribution to guarantee that the ℓ_∞-norm bound β_{PKE} of the following term in Theorem 5.5 for XR-PKE holds except with negligible probability, i.e., 2^{-128},

$$\beta_{\mathsf{PKE}} := \|\langle \mathbf{e}, \mathbf{r} \rangle + y - \langle \mathbf{s}, \mathbf{e}_u \rangle + \langle \mathbf{s}, \mathbf{c}_u \rangle - c_v\|_\infty < \frac{q}{4}$$

where $(\mathbf{s}, \mathbf{e}) \leftarrow \mathcal{U}(\mathbb{S}_\nu^n) \times \mathcal{U}(\mathbb{S}_\nu^m)$, $(\mathbf{r}, \mathbf{e}_u) \leftarrow \mathcal{D}_{\sigma_0}^n \times \mathcal{D}_{\sigma_0}^m$, $y \leftarrow \mathcal{D}_{\sigma_1}$, $\mathbf{c}_u := \mathbf{c} - \lfloor \lfloor \mathbf{c} \rceil_{2^{d_u}} \rceil_q$, and $c_v := c - \lfloor \lfloor c \rceil_{2^{d_v}} \rceil_q$. Thus, we can bound the ℓ_∞-norms by $\|\mathbf{c}_u\|_\infty \leqslant q/2^{d_u+1}$, and $\|c_v\|_\infty \leqslant q/2^{d_v+1}$, respectively. Similarly, we can derive the tail bound of the ℓ_∞-norm β_{KEM} in Theorem A.2 as well.

Fifth, towards XR-PKE, we fix $d_v = 2$ in advance to compress the size of key-*dependent* ciphertext as much as possible. Note that the sizes of key-*dependent* ciphertext in the constructions of XR-KEM and XR-PKE are both independent with the value of reproducibility count N, i.e., $|\widehat{\mathsf{ct}}| = d/8 = 32$ Bytes and $|\widehat{\mathsf{ct}}| = d \cdot d_v/8 = 64$ Bytes, respectively.

Sixth, we begin by setting the modulus $q \approx 2^{12}$ and

$d_u := \lfloor \log q \rceil$. We compute n, m with $\bar{\chi} := \mathcal{U}(\mathbb{S}_v)$ and $\chi := \mathcal{D}_\sigma$ by the LWE estimator [3] to guarantee practical hardness of $\mathsf{MLWE}_{\mathcal{R},n,m,q,\bar{\chi}}$ and $\mathsf{MLWE}_{\mathcal{R},m+N,n,q,\chi}$ at 128-bit, 192-bit, and 256-bit security levels. The latter MLWE assumption stems from $\mathsf{MatrixHint\text{-}MLWE}^{N,\chi_1,S}_{\mathcal{R},m+N,n,q,\chi_0}$ problem via the reduction in Theorem 5.2. As earlier works [15, 24, 25, 37], we use root Hermite factor (RHF) around 1.0045, 1.0029, 1.0023 to measure the practical hardness of MLWE at 128-bit, 192-bit, and 256-bit secure level, respectively. With the specific n, m, N, q, we compute the ℓ_∞-norm bound β and compare β with $\lceil q/4 \rceil$. We increase the modulus q by factor 2 and repeat computing the parameters until $\beta < \lceil \frac{q}{4} \rceil$.

In the end, after finding the smallest modulus q, we show how to find the smallest d_u in the compression function of mmPKE constructions which can compress the key-independent ciphertext as much as possible. We first change $d_u = 1$ and increase d_u until $\beta < \lceil q/4 \rceil$ holds with overwhelming probability. We provide a script to compute a tight upper bound on ζ as part of our implementation code.

Following the metric in [32], for $\mathsf{CON} \in \{\mathsf{KEM}, \mathsf{PKE}\}$, we define

$$k_{\mathsf{com}}^{\mathsf{CON}} := \frac{N \cdot \left|\mathsf{ct}^{\mathsf{Kyber}}\right|}{\left|\mathsf{ct}_0^{\mathsf{CON}}\right| + N \cdot \left|\widehat{\mathsf{ct}}^{\mathsf{CON}}\right|} \xrightarrow{N \to \infty} \frac{\left|\mathsf{ct}^{\mathsf{Kyber}}\right|}{\left|\widehat{\mathsf{ct}}^{\mathsf{CON}}\right|},$$

which measures the *compactness* of our mmPKE/mmKEM compared to the trivial solution via Kyber in the asymptotic regime. Notably, we achieve significant improvements, with $k_{\mathsf{com}}^{\mathsf{KEM}} = 24, 34, 49$ and $k_{\mathsf{com}}^{\mathsf{PKE}} = 12, 17, 24.5$ when compared to Kyber512, Kyber768, and Kyber1024 [15], respectively.

6 Implementations and Benchmarks

To evaluate the performance of our constructions, we have implemented the lattice-based mmPKE and mmKEM built from our XR-PKE and XR-KEM, named mmCipher-PKE and mmCipher-KEM, respectively, in portable C [4]. Further details of our implementations and benchmarks are shown in Appendix C.

As a baseline comparison, we compare our plain C implementations to the official C reference implementation of (CPA-secure) Kyber[5] with the standard parameter settings of ML-KEM-512, ML-KEM-768, ML-KEM-1024 to achieve 128-bit, 192-bit, 256-bit security, respectively [43, Table 2]. We compare this baseline to our C implementation using the same compiler and target system, an AMD Ryzen 7 4850U Linux laptop running at 3.3 GHz (with overclocking disabled) for 1000 repetitions. Average timing is reported.

In mmPKE/mmKEM, encryption/encapsulation is the most costly operation as its cost increases with the number of recipients N. Our main contribution is to significantly reduce this cost. We summarize the results on the encryption/encapsulation operation comparing with CPA-secure Kyber (ML-KEM) in Figure 7 and Figure 8, while deferring the results for other operations to Appendix C.

As predicted by the theoretical analysis in Section 5.3, for $N = 1024$ recipients, among different security levels, mmCipher-KEM and mmCipher-PKE achieve a 23–45× and 12–23× reduction in bandwidth, respectively. In particular, for $N \geqslant 16$ recipients, our constructions already demonstrate a significant improvement (by a factor of over 5). Furthermore, for $N = 1024$ recipients, the bandwidth of our mmCipher-KEM is only 4–9% larger than the plaintext size (*near optimal bandwidth*).

Regarding the computational cost of encapsulation/encryption, for $N = 1024$ recipients, among different security levels, our mmCipher- KEM and mmCipher-PKE offer 3–5× reduction. For $N \geqslant 4$, our constructions are already faster than the baseline. This is because the most expensive operation, i.e., generating the key-independent ciphertext, is amortized across recipients.

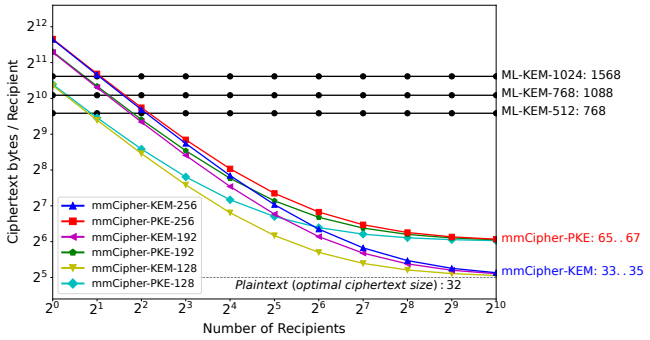


Figure 7: mmCipher and ML-KEM total ciphertext output in bytes when sending N 256-bit messages (or keys) to N recipients, divided by the number of recipients.

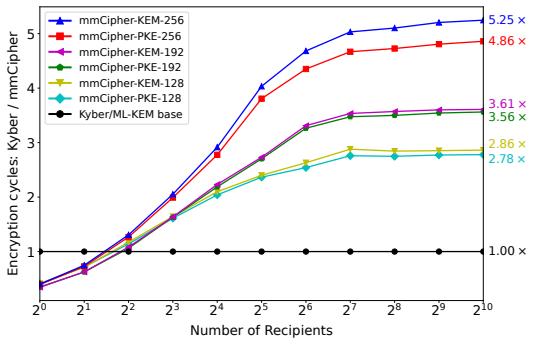


Figure 8: mmCipher encryption/encapsulation speed when sending N 256-bit messages (or keys) to N recipients, relative to ML-KEM at the same security level.

[4] Provided in our artifact: https://doi.org/10.5281/zenodo.17849532

[5] Kyber C reference code (`ref`): https://github.com/pq-crystals/kyber

Ethical Considerations

Our work proposes more efficient post-quantum multi-recipient encryption techniques, which may impact several stakeholders, including end-users of secure communication systems, developers deploying PQC, and broader society. While stronger and more scalable encryption improves privacy and security, it also carries inherent dual-use concerns because the same capabilities may be misused to conceal harmful activity. Recognizing these implications, we assess our contributions primarily as enhancing the practical adoption and performance of post-quantum protection in large-scale systems, while acknowledging the possibility of misuse that exists for all cryptographic primitives.

All experiments were conducted using synthetic data in isolated environments, without interacting with real users, external systems, or production networks. No artifacts that could meaningfully aid malicious behavior are released. We believe publishing this work is ethically justified because it promotes transparency in cryptographic design, supports secure system development, and informs the community about both the capabilities and limitations of emerging post-quantum mechanisms. Some of this ethical reflection is necessarily post hoc, and we hope it will help guide future research on the broader societal impact of foundational cryptographic tools.

Open Science

Artifact URL: https://doi.org/10.5281/zenodo.17849532.

We provide self-contained Python and portable C implementations of the scheme (with some aspects that may need further optimization for production-level deployment). The artifact also contains the code used for generating the comparative benchmarks reported in this work, source code for ZK proof experiments using the LaZer Library, and scripts and tools used for security parameter selection (computation of lattice parameter sets and decryption failure probabilities).

Acknowledgments

R. Steinfeld, M.F. Esgin was supported by Australian Research Council Discovery Grant DP250100229. R. Steinfeld was also supported by Australian Research Council Discovery Grant DP220101234. Siu-Ming Yiu was supported by HKU-SCF FinTech Academy, Shenzhen-Hong Kong-Macao Science and Technology Plan Project (Category C Project: SGDX20210823103537030), Theme-based Research Scheme of RGC, Hong Kong (T35-710/20-R). We thank the anonymous reviewers for their helpful comments.

References

[1] C. Abou Haidar, A. Passelègue, and D. Stehlé. Efficient updatable public-key encryption from lattices. In *Advances in Cryptology – ASIACRYPT 2023*, pages 342–373, 2023.

[2] G. Alagic, G. Alagic, D. Apon, D. Cooper, Q. Dang, T. Dang, J. Kelsey, J. Lichtinger, Y.-K. Liu, C. Miller, et al. Status report on the third round of the nist post-quantum cryptography standardization process. Technical report, National Institute of Standards and Technology, 2022.

[3] M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

[4] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe. Post-quantum key exchange — a new hope. In *25th USENIX security symposium (USENIX Security 16)*, pages 327–343, 2016.

[5] J. Alwen, D. Hartmann, E. Kiltz, and M. Mularczyk. Server-aided continuous group key agreement. In *Proc. CCS 2022*, page 69–82, 2022.

[6] J. Alwen, D. Hartmann, E. Kiltz, M. Mularczyk, and P. Schwabe. Post-quantum multi-recipient public key encryption. In *Proc. CCS 2023*, page 1108–1122, 2023.

[7] M. Barbosa and P. Farshim. Randomness reuse: Extensions and improvements. In *Cryptography and Coding: 11th IMA International Conference*, pages 257–276, 2007.

[8] E. Barker, L. Chen, S. Keller, A. Roginsky, A. Vassilev, and R. Davis. Recommendation for pair-wise key-establishment schemes using discrete logarithm cryptography. Technical report, National Institute of Standards and Technology, 2017.

[9] R. Barnes, B. Beurdouche, R. Robert, J. Millican, E. Omara, and K. Cohn-Gordon. The Messaging Layer Security (MLS) Protocol. RFC 9420, July 2023.

[10] M. Bellare, A. Boldyreva, and J. Staddon. Multi-recipient encryption schemes: Security notions and randomness re-use. In *Public Key Cryptography — PKC 2003*, pages 85–99, 2002. URL https://cseweb.ucsd.edu/~mihir/papers/bbs.pdf.

[11] M. Bellare, A. Boldyreva, K. Kurosawa, and J. Staddon. Multirecipient encryption schemes: How to save on bandwidth and computation without sacrificing security. *IEEE Transactions on Information Theory*, 53(11):3927–3943, 2007.

[12] D. Boneh, R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. *SIAM Journal on Computing*, 36(5):1301–1328, 2007.

[13] J. Bootle, V. Lyubashevsky, N. K. Nguyen, and A. Sorniotti. A framework for practical anonymous credentials from lattices. In *Advances in Cryptology – CRYPTO 2023*, pages 384–417, 2023.

[14] 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. In *Proc. CCS 2016*, pages 1006–1018, 2016.

[15] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehle. Crystals - kyber: A cca-secure module-lattice-based kem. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 353–367, 2018.

[16] B. Bünz, S. Agrawal, M. Zamani, and D. Boneh. Zether: Towards privacy in a smart contract world. In *Financial Cryptography and Data Security*, pages 423–443, 2020.

[17] R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In *Advances in Cryptology-EUROCRYPT 2004*, pages 207–222. Springer, 2004.

[18] H. Cheng, X. Li, H. Qian, and D. Yan. Cca secure multi-recipient kem from lpn. In *Information and Communications Security*, pages 513–529, 2018.

[19] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology — CRYPTO '98*, pages 13–25, 1998.

[20] R. del Pino and S. Katsumata. A new framework for more efficient round-optimal lattice-based (partially) blind signature via trapdoor sampling. In *Advances in Cryptology – CRYPTO 2022*, pages 306–336, 2022.

[21] M. H. Devoret and R. J. Schoelkopf. Superconducting circuits for quantum information: an outlook. *Science*, 339(6124):1169–1174, 2013.

[22] B. E. Diamond. Many-out-of-many proofs and applications to anonymous zether. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1800–1817, 2021.

[23] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

[24] M. F. Esgin, R. Steinfeld, and R. K. Zhao. MatRiCT+: More efficient post-quantum private blockchain payments. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1281–1298, 2022.

[25] M. F. Esgin, R. Steinfeld, D. Liu, and S. Ruj. Efficient hybrid exact/relaxed lattice proofs and applications to rounding and vrfs. In *Advances in Cryptology – CRYPTO 2023*, pages 484–517, 2023.

[26] T. Espitau, G. Niot, and T. Prest. Flood and submerse: Distributed key generation and robust threshold signature from lattices. In *Advances in Cryptology – CRYPTO 2024*, pages 425–458, 2024.

[27] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Advances in Cryptology — CRYPTO' 99*, pages 537–554, 1999.

[28] C. Gentry and B. Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In *Advances in Cryptology - EUROCRYPT 2009*, pages 171–188, 2009.

[29] Y. Guo, H. Karthikeyan, A. Polychroniadou, and C. Huussin. Pride ct: Towards public consensus, private transactions, and forward secrecy in decentralized payments. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 3904–3922. IEEE, 2024.

[30] K. Hashimoto, S. Katsumata, E. Postlethwaite, T. Prest, and B. Westerbaan. A concrete treatment of efficient continuous group key agreement via multi-recipient pkes. In *Proc. CCS 2021*, page 1441–1462, 2021.

[31] S. Katsumata. A new simple technique to bootstrap various lattice zero-knowledge proofs to qrom secure nizks. In *Advances in Cryptology – CRYPTO 2021*, pages 580–610, 2021.

[32] S. Katsumata, K. Kwiatkowski, F. Pintore, and T. Prest. Scalable ciphertext compression techniques for post-quantum kems and their applications. In *Advances in Cryptology – ASIACRYPT 2020*, pages 289–320, 2020.

[33] J. Katz and N. Wang. Efficiency improvements for signature schemes with tight security reductions. In *Proc. CCS 2003*, page 155–164, 2003.

[34] D. Kim, D. Lee, J. Seo, and Y. Song. Toward practical lattice-based proof of knowledge from hint-mlwe. In *Advances in Cryptology – CRYPTO 2023*, pages 549–580, 2023.

[35] K. Kurosawa. Multi-recipient public-key encryption with shortened ciphertext. In *Public Key Cryptography*, pages 48–63, 2002.

[36] Z. Liu, K. Sotiraki, E. Tromer, and Y. Wang. Snake-eye resistant pke from lwe for oblivious message retrieval and robust encryption. In *Advances in Cryptology – EUROCRYPT 2025*, pages 126–156. Springer, 2025.

[37] V. Lyubashevsky, N. K. Nguyen, and M. Plançon. Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In *Advances in Cryptology – CRYPTO 2022*, pages 71–101, 2022.

[38] V. Lyubashevsky, G. Seiler, and P. Steuer. The lazer library: Lattice-based zero knowledge and succinct proofs for quantum-safe privacy. In *Proc. CCS 2024*, page 3125–3137, 2024.

[39] T. Matsuda and G. Hanaoka. Key encapsulation mechanisms from extractable hash proof systems, revisited. In *Public-Key Cryptography – PKC 2013*, pages 332–351, 2013.

[40] D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 700–718. Springer, 2012.

[41] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 427–437, 1990.

[42] NIST. SHA-3 standard: Permutation-based hash and extendable-output functions. Federal Information Processing Standards Publication FIPS 202, August 2015.

[43] NIST. Module-Lattice-based Key-Encapsulation Mechanism Standard. Federal Information Processing Standards Publication FIPS 203, August 2024.

[44] NIST. Module-Lattice-Based Digital Signature Standard. Federal Information Processing Standards Publication FIPS 204, August 2024.

[45] C. Peikert. Lattice cryptography for the internet. In *Post-Quantum Cryptography*, pages 197–219, 2014.

[46] C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 187–196, 2008.

[47] A. Pinto, B. Poettering, and J. C. Schuldt. Multi-recipient encryption, revisited. In *Proc. AsiaCCS 2014*, page 229–238, 2014.

[48] R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *International conference on the theory and application of cryptology and information security*, pages 552–565. Springer, 2001.

[49] A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th annual symposium on foundations of computer science (Cat. No. 99CB37039)*, pages 543–553, 1999.

[50] N. P. Smart. Efficient key encapsulation to multiple parties. In *Security in Communication Networks*, pages 208–219, 2005.

[51] R. Steinfeld, S. Ling, J. Pieprzyk, C. Tartary, and H. Wang. Ntrucca: How to strengthen ntruencrypt to chosen-ciphertext security in the standard model. In *Public Key Cryptography–PKC 2012*, pages 353–371. Springer, 2012.

[52] E. E. Targhi and D. Unruh. Post-quantum security of the fujisaki-okamoto and oaep transforms. In *Theory of Cryptography*, pages 192–216, 2016.

[53] P. Technology. BitChat Protocol Whitepaper. https://github.com/permissionlesstech/bitchat/blob/main/WHITEPAPER.md, 2025. Accessed: 2025-08-25.

[54] H. Wang, R. Steinfeld, M.-J. O. Saarinen, M. F. Esgin, and S.-M. Yiu. mmCipher: Batching post-quantum public key encryption made bandwidth-optimal. Cryptology ePrint Archive, Paper 2025/1000, 2025. URL https://eprint.iacr.org/2025/1000.

[55] Z. Yang. On constructing practical multi-recipient key-encapsulation with short ciphertext and public key. *Security and Communication Networks*, 8(18):4191–4202, 2015.

A Construction of XR-KEM

Employing the reconciliation mechanism (as introduced in the full version [54]), we can further compress the ciphertext size, especially for the key-*dependent* ciphertext, and then obtain a lattice-based XR-KEM, which can be used to build an mmKEM. Following [47], the mmKEM can be extended to an mmPKE for arbitrary-length message via a DEM.

Construction A.1 (XR-KEM from Lattices)**.** Let λ be a security parameter, $m = m(\lambda)$, $n = n(\lambda)$, $d = d(\lambda)$, $q = q(\lambda)$, $N = N(\lambda)$, $\nu = \nu(\lambda)$ be positive integers. Let $\sigma_0 = \sigma_0(\lambda)$, $\sigma_1 = \sigma_1(\lambda)$ be Gaussian width parameters. Let $\mathsf{dbl}(\cdot)$, $\mathsf{rec}(\cdot,\cdot)$, $\lfloor\cdot\rceil_2$, and $\langle\cdot\rangle_2$ be the functions as define in [54, Lemma A.6] and [54, Lemma A.7] which are extended to $\mathcal{R}_q$ per [54, Remark A.8]. For the encapsulated key space $\mathcal{M} = \{0,1\}^d$, the detailed construction is shown in Figure 9. We summarize the notations in Table 2.

The extended reproducibility of our XR-KEM is analogous to that of our XR-PKE in Construction 5.3. The security proof of our XR-KEM closely follows that of our XR-PKE, except that, due to the reconciliation mechanism, the encapsulated key is statistically indistinguishable from random under the (Matrix Hint-)MLWE assumption.

We focus on its correctness, as follows.

Correctness. We set $\mathsf{Compress}(x) = \lfloor x \mod q \rceil_{2^{d_u}}$ and $\mathsf{Decompress}(x) = \lfloor x \mod 2^{d_u} \rceil_q$. Like our XR-PKE, here, we mainly consider the case that the key-independent ciphertext is compressed and then decompressed before the decryption, as done in mmPKE compiler of Construction 4.3.

Theorem A.2 (Correctness). *Let $\mathbf{e}, \mathbf{s}, \mathbf{r}, \mathbf{e}_u, y$ be random variables that have the corresponding distribution as in Construction A.1. Denote ζ as*

$$\Pr\left[\left\| 2\Big(\langle \mathbf{e}, \mathbf{r}\rangle + y - \langle \mathbf{s}, \mathbf{e}_u\rangle + \langle \mathbf{s}, \mathbf{c}_u\rangle\Big) - \bar{e}\right\|_\infty \geqslant \frac{q}{4}\right]$$

where $\mathbf{c}_u := \mathbf{c} - \lfloor \lfloor \mathbf{c} \mod q \rceil_{2^{d_u}} \rceil_q \in \mathcal{R}^m$, and $\bar{e}$ denotes the error in $\mathsf{dbl}(c)$ function. We say our Construction A.1 is ζ-correct.

Proof. Considering the compression and decompression of *independent* ciphertext $\mathbf{c}$, the value $\mathbf{c}$ (renamed as $\mathbf{c}'$) in Decap algorithm is

$$\mathbf{c}' := \lfloor \lfloor \mathbf{c} \mod q \rceil_{2^{d_u}} \rceil_q .$$

One can observe that the decapsulation is made via reconciliation mechanism. It means that the decapsulation succeeds if and only if the following equation holds,

$$\lfloor \bar{c} \rceil_2 = \mathsf{rec}\big(2 \cdot \langle \mathbf{c}', \mathbf{s}\rangle, \langle \bar{c} \rangle_2\big).$$

By [54, Lemma A.7], $\mathsf{rec}(\cdot,\cdot)$ works if the following holds,

$$\| \bar{c} - 2 \cdot \langle \mathbf{c}', \mathbf{s}\rangle \mod 2q \|_\infty < \frac{2q}{8} = \frac{q}{4}.$$

Plugging $\bar{c} = \mathsf{dbl}(c) = 2c - \bar{e}$ and $\mathbf{c}' = \mathbf{c} - \mathbf{c}_u$, the above inequality is equivalent to

$$\| 2c - \bar{e} - 2 \cdot \langle \mathbf{c} - \mathbf{c}_u, \mathbf{s}\rangle \|_\infty < \frac{q}{4}.$$

Since the value of $c := \langle \mathbf{b}, \mathbf{r}\rangle + y$ in $\mathsf{Encap}^{\mathsf{d}}$ algorithm where the value of $\mathbf{b} := \mathbf{A}^\top \mathbf{s} + \mathbf{e}$, we can obtain

$$2c - \bar{e} - 2 \cdot \langle \mathbf{c} - \mathbf{c}_u, \mathbf{s}\rangle = 2\Big(\langle \mathbf{e}, \mathbf{r}\rangle + y - \langle \mathbf{s}, \mathbf{e}_u\rangle + \langle \mathbf{s}, \mathbf{c}_u\rangle\Big) - \bar{e}.$$

It means that when ℓ_∞-norm of the decapsulation error is no less than $q/4$, i.e., $\|2(\langle \mathbf{e}, \mathbf{r}\rangle + y - \langle \mathbf{s}, \mathbf{e}_u\rangle + \langle \mathbf{c}_u, \mathbf{s}\rangle) - \bar{e}\|_\infty \geqslant q/4$, the decapsulation will fail. Thus, the value ζ is no more than the probability of decapsulation failure. □

B Removing the KOSK Assumption

In this section, using a multi-proof extractable NIZK argument system, we present a compiler that can remove the KOSK assumption of the mmPKE with the polynomial-sized number of recipients and provide a detailed analysis of its security. Last, we provide an instantiation for our mmPKE.

Construction B.1 (KOSK Compiler). For $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CCA}\}$, let mmPKE' be an $\mathsf{mmIND\text{-}ATK}^{\mathsf{KOSK}}$ secure mmPKE with public-private key space $\mathcal{K}$ and randomness distributions $\mathcal{D}_{\mathsf{i}}$, $\mathcal{D}_{\mathsf{d}}$. Let Π be a NIZK argument system. Denote the relation R_Π in Π as

$$R_\Pi := \{(\mathsf{pk}; \mathsf{sk}) \mid (\mathsf{pk}, \mathsf{sk}) \in \mathcal{K}\}$$

We assume the hash value $\mathsf{H}(0) = \mathsf{crs}_\Pi$. The construction of compiler $\mathsf{Comp}^{\mathsf{KOSK}}[\mathsf{mmPKE}', \Pi]$ is defined in Figure 10 which outputs an mmIND-ATK secure mmPKE.

The correctness is easy to see. We show how to reduce the security of mmPKE output by $\mathsf{Comp}^{\mathsf{KOSK}}[\mathsf{mmPKE}', \Pi]$ to the security of input mmPKE' and Π. The proof is deferred to the full version [54] due to space constraints.

Theorem B.2 (Security). *For $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CCA}\}$, if mmPKE' is $\mathsf{mmIND\text{-}ATK}^{\mathsf{KOSK}}$ secure and Π is a NIZK argument system satisfies correctness, multi-proof extractability and zero knowledge, our $\mathsf{mmPKE} \leftarrow \mathsf{Comp}^{\mathsf{KOSK}}[\mathsf{mmPKE}', \Pi]$ output by Construction B.1 is mmIND-ATK secure.*

Remark B.3 (Recipient Registration and Delegate Verification). In practice, each recipient can be required to "register" to some semi-honest third party, e.g., server in advance. Both proving and the verification for each public key are *one-time* and the latter can be delegated to the server as well. Thus, in this setting, both bandwidth and computation for the encryption do not increase.

B.1 NIZK Instantiations in KOSK Compiler

In this subsection, we discuss the post-quantum instantiations of NIZK in the KOSK complier. and present *proof-of-concept* implementations of the NIZK instantiations, which help estimate their practical cost.

Specifically, we recommend Schnorr-like lattice-based protocols that satisfy knowledge soundness and can efficiently prove the well-formedness of ciphertexts and keys. To achieve the multi-proof extractability, we can apply Katsumata Transform [31] as demonstrated in [13, 20], which leverages an *extractable linear homomorphic commitment* (LHC) that can be seen as a linear homomorphic encryption scheme with pseudo-random public keys.

Among them, LNP22 [37] is one of the most efficient lattice-based NIZKs and has recently been implemented in the LaZer library [38]. Recent work [13] extends LNP22 to achieve multi-proof extractability, but they do not provide the implementation of this variant. Therefore, we report on the results of the regular LNP22 implementation from the LaZer library as a *proof-of-concept*.

Specifically, we need to generate the "exact" range proof for the private key $(\mathbf{s}_i, \mathbf{e}_i)$, i.e., $||(\mathbf{s}_i, \mathbf{e}_i)||_\infty \leqslant 1$, along with a linear relation $\mathbf{A}^\top \mathbf{s}_i + \mathbf{e}_i = \mathbf{b}_i$. For $\bar{\nu} = 2$ (i.e., each coefficient

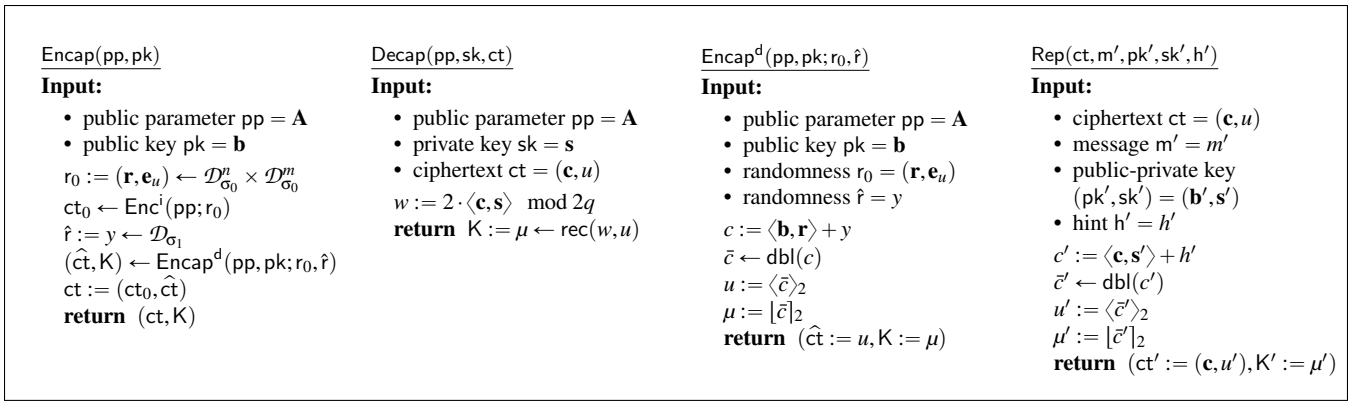


Figure 9: An IND-CPA$^{\mathsf{XR}}$ secure lattice-based XR-KEM where Setup, KGen, Enc$^{\mathsf{i}}$, and HintGen are the same as the ones in Construction 5.3.

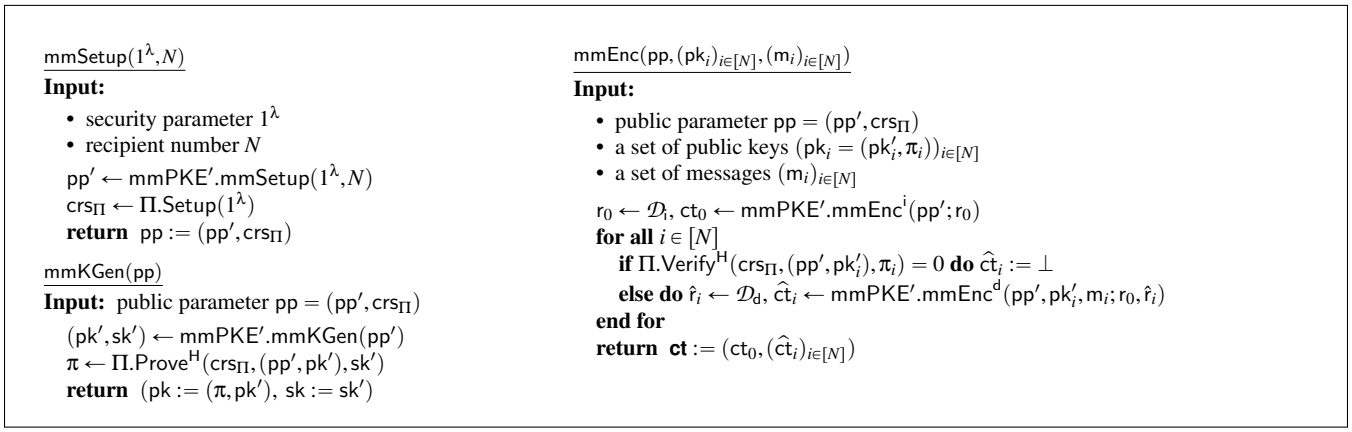


Figure 10: An mmIND-ATK secure mmPKE output by the KOSK compiler Comp$^{\mathsf{KOSK}}[\mathsf{mmPKE}', \Pi]$ for ATK $\in \{\mathsf{CPA}, \mathsf{CCA}\}$. mmExt and mmDec are the same as the ones in mmPKE′.

is in $\{0,1\}$), we simply use the concatenation $(\ \mathbf{s}^\top \ ||\ \mathbf{e}^\top\ )$ as a binary witness, proving that $\left(\ \mathbf{A}^\top \ ||\ \mathbf{I}\ \right)(\ \mathbf{s}^\top \ ||\ \mathbf{e}^\top\ )^\top - \mathbf{b} = 0$. For ternary secrets ($\bar{\nu} = 3$), the secret key is split into binary components representing positive and negative coefficients and the proof is of the form

$$\left(\ \mathbf{A}^\top \ ||\ \mathbf{I}\ ||\ -\mathbf{A}^\top \ ||\ -\mathbf{I}\ \right)(\ \mathbf{s}_+^\top \ ||\ \mathbf{e}_+^\top \ ||\ \mathbf{s}_-^\top \ ||\ \mathbf{e}_-^\top\ )^\top - \mathbf{b} = 0.$$

During the proof, we need to first prove the witness with binary coefficients and then prove the linear relation. Here although the modulus q may be smaller than the modulus in the proof system, LNP22 and its implementation in LaZer can still prove such relations efficiently. More details can be referred their papers [37, 38].

Table 4 offers representative numbers (timings on an AMD Ryzen 7 7840U laptop, 3.3 GHz). Note that the proofs have not been optimized for size or tuned for the target security level. We observe that these NIZK proofs, which need to be verified *only once* after generation, are less than 30 KB in size. Furthermore, proof generation and verification are very efficient. In practice, this process can be delegated to a semi-honest third party, e.g., a server, and completed in "registration" phase. Hence, this NIZK has minimal impact on the performance of both encapsulation and decapsulation.

Table 4: mmCipher public-key NIZK proof sizes, proof generation and verification timings using LNP22.

Scheme	Dist.	Proof Size	Proof time	Verify time
mmCipher-128	$\bar{\nu} = 3$	26,473 B	0.078 s	0.040 s
mmCipher-192	$\bar{\nu} = 2$	25,261 B	0.075 s	0.042 s
mmCipher-256	$\bar{\nu} = 2$	28,022 B	0.105 s	0.056 s

C Implementation and Benchmarking Details

In this section, we detail the implementation aspects of our CPA-secure primitives mmCipher-KEM (Cons. 4.3+A.1) and mmCipher-PKE (Cons. 4.3+5.3) and provide further benchmarks.

We list some key technical similarities and differences between Kyber (ML-KEM) and mmCipher that impact performance characteristics. In the following, references are made to Kyber components as described in the ML-KEM standard FIPS 203 [43] rather than in the original Kyber paper [15]. Our comparison uses the reference code and parameter sets of the final standard.

- The programming interfaces of mmCipher are designed for batch encryption of unique messages/shared secrets to a large number of recipients. This is the main use case and optimization target of the implementation.

- The parameter selection of the implementation supports 2^{10} recipients, which requires a larger modulus $q = 2^{25} - 2^{12} + 1$. Hence, the Number Theoretic Transforms (NTTs) operate on 32-bit elements rather than 16-bit elements, as with Kyber's $q = 3329$. Our 25-bit ring is quite similar to the 23-bit, degree-256 ring of Dilithium (ML-DSA [44]). The binary and ternary secret distributions of mmCipher also allow efficient non-NTT ring multiplication operations based on conditional additions, although these may be difficult to implement in constant time in software.

- We use the SHAKE eXtendable-Output Function (XOF) [42] for all random sampling, as is done in ML-KEM. SHAKE-128 is used for all operations at the 128-bit security level and for $\mathbf{A}$ matrix expansion at all security levels (as in ML-KEM). SHAKE-256 is used for other samplers and hashes at levels 192 and 256.

- Secret keys are sampled from a narrow uniform distribution $\mathcal{U}(\mathbb{S}_\nu)$ instead of a Centered Binomial Distribution (CBD) as in ML-KEM. The ternary ($\bar{\nu} = 3$) sampler uses rejection sampling of bytes against $3^5 = 243$; only $1 - 243/256 \approx 5\%$ of bytes are rejected, while accepted bytes yield 5 ternary digits $\{-1, 0, +1\}^5$. The "base-243" system also allows a convenient and compact storage format for ternary secret keys. Binary ($\bar{\nu} = 2$) secret key sampling and storage is trivial and optimally efficient.

- We sample ephemeral randomness from discrete Gaussian distributions $\mathcal{D}_{\sigma_0}$ and $\mathcal{D}_{\sigma_1}$ rather than from CBD. [6] Note that Gaussian widths $\sigma := \sqrt{2\pi} \cdot \mathfrak{s}$ are related to standard deviation $\mathfrak{s}$. More precisely, following Section 5.3, we fix the Gaussian width $\sigma_0 = 15.90$ and $\sigma_1 = 368459.34, 488797.36, 554941.07$ to support up to 2^{10} recipients at 128-bit, 192-bit, and 256-bit security, respectively.

- The encryption/decryption mechanism of mmCipher-PKE is similar to Kyber, but mmCipher-KEM uses a reconciliation mechanism over $\mathcal{R}_q$, requiring the cross-rounding function $\langle\cdot\rangle_2$ and the reconciliation function $\mathsf{rec}(\cdot,\cdot)$. The Python implementation also has the randomized doubling function $\mathsf{dbl}(\cdot)$ available, but since entropy leakage ("bias") fixed by $\mathsf{dbl}(\cdot)$ can be shown to be practically negligible with our q value, the C code does not implement randomization here. These implementations are interoperable (and produce fully matching ciphertext with high probability.)

- Since SHAKE/SHA3 [42] computation is typically the biggest individual ML-KEM performance bottleneck (on some platforms consuming more than half of total key establishment cycles), for a fair comparison, the underlying KECCAK-$p[1600, 24]$ permutation implementation in mmCipher is the same plain C code as in the Kyber reference implementation.

Note that these algorithms would greatly benefit from handcrafted SIMD and vectorization optimizations (e.g., AVX-512 or ARM SVE2). However, we currently only have a portable C implementation for mmCipher, so we are comparing such implementations of both schemes.

We also list the computational costs of other operations in Table 5. The results show that the key generation and decryption/decapsulation operations are equally fast or faster than those equivalent Kyber operations at the same security level. Note that the input seed (e.g., 32 bytes) for $\mathsf{mmSetup}()$ is a public "system parameter" shared by all users, and the operation needs to be re-run only when it changes.

Regarding the bandwidth of key generation, for $N = 1024$ recipients, the public key sizes are 2.4, 4.3, 5.5 KB larger than the one in the baseline, among 128-, 192-, 256-bit security. However, these additional costs are *one-time* and can be amortized over multiple uses, minimizing their impact on the overall efficiency.

Towards the bandwidth of decryption/decapsulation, for $N = 1024$ recipients, the individual ciphertext in our mmCipher is only 0.5, 1.4, 1.6 KB larger than the one in the baseline, among 128-, 192-, 256-bit security. These additional costs will likely not affect the usability of the scheme in the use cases for which it is best suited.

In the end, more comprehensive benchmark results together are deferred to the full version [54] due to space constraints.

[6] Artifact code uses rounded Gaussians for $\mathcal{D}_{\sigma_0}$ and $\mathcal{D}_{\sigma_1}$, using the polar Marsaglia method implemented with 64-bit IEEE 754 arithmetic to sample from a rounding-compensated $s' = \sqrt{\sigma^2/2\pi - 1/12}$ continuous Gaussian distribution. This sampler is approximate and not constant-time; it is a placeholder implementation. A more appropriate Discrete Gaussian sampler is required for production-level implementation.

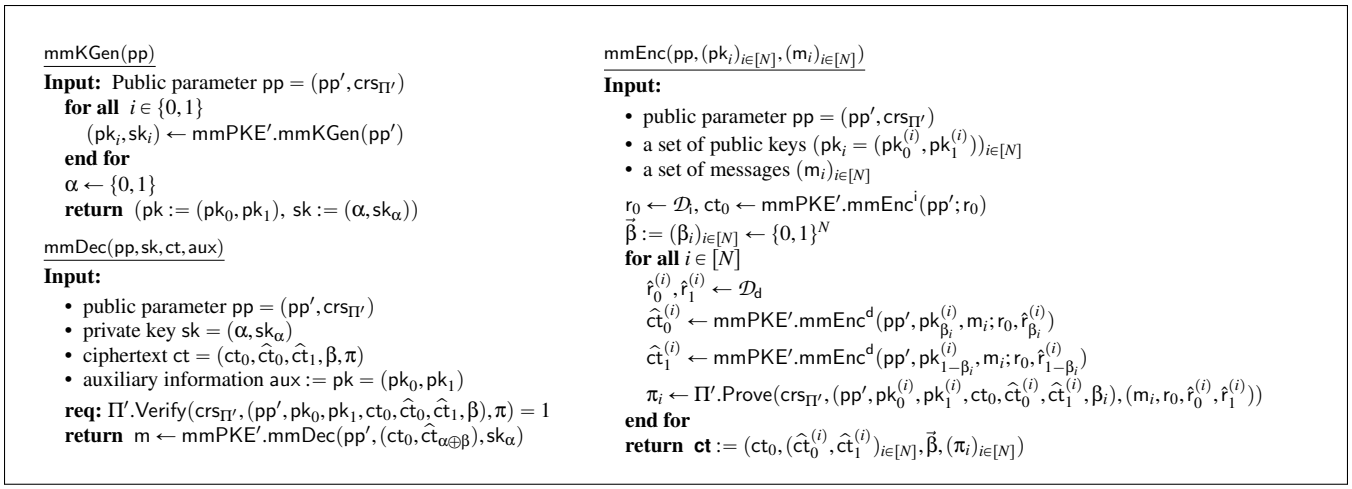


Figure 11: An adaptively secure mmPKE output by the compiler $\mathsf{Comp}^{\mathsf{CCA}}[\mathsf{mmPKE}', \Pi']$. mmExt with input index i is defined by picking the relevant components $(\mathsf{ct}_0, \hat{\mathsf{ct}}_0^{(i)}, \hat{\mathsf{ct}}_1^{(i)}, \beta_i, \pi_i)$ from **ct**. mmSetup is the same as the one in Construction B.1 except for replacing Π by Π'.

Table 5: Cycle counts of other operations in mmCipher and ML-KEM (Kyber). Note that K-PKE is an internal CPA subcomponent of ML-KEM.

Operation	*PQ Security*		
	128-bit	**192-bit**	**256-bit**
mmSetup()	188,755	543,640	916,016
mmKGen()	58,815	78,383	106,504
mmDec()	43,511	68,072	85,872
mmDecap()	43,246	67,705	85,323
ML-KEM.KeyGen()	99,145	170,323	262,044
ML-KEM.Decaps()	168,358	259,511	372,644
K-PKE.Decrypt()	40,987	54,547	68,070

D Our Adaptively Secure mmPKE

In this section, we propose a generic construction that transforms a CPA-secure mmPKE into an adaptively secure mmPKE. Our approach generalizes the Naor-Yung paradigm [41, 49] to mmPKE, introducing an optimization: we merge the double encryption into a single multi-recipient ciphertext, only need to generate a single *independent* ciphertext. This optimization significantly reduces the size of both multi-recipient and individual ciphertexts. With our construction, not only can our lattice-based mmPKEs be transformed to achieve adaptive security, but also can the traditional mmPKEs proposed in [10, 11, 35, 47].

Compared to other adaptively secure (m)PKE constructions [6, 30, 32], our approach requires only the addition of NIZK proofs. These proofs can be aggregated, making the size constant or polylogarithmic in the number of recipients, and verification can be delegated to a server, making our construction remain both flexible and efficient, especially for large numbers of recipients.

In addition, our constructions also imply an adaptive corruption compiler which enables both CPA- and CCA-secure mmPKEs, such as the ones in [10, 11, 35, 47], to resist adaptive corruption, with some requiring KOSK assumption removal through our KOSK compiler in advance.

Construction D.1 (Adaptive Security Compiler)**.** Let mmPKE' be an mmIND-CPA secure mmPKE with the randomness distributions $\mathcal{D}_1$, $\mathcal{D}_d$. Let Π' be a NIZK argument system. Denote the relation $R_{\Pi'}$ in Π' as

$$\left\{ \begin{array}{l} ((\mathsf{pp}', \mathsf{pk}_0, \mathsf{pk}_1, \\ \mathsf{ct}_0, \hat{\mathsf{ct}}_0, \hat{\mathsf{ct}}_1, \beta); \\ (\mathsf{m}, \mathsf{r}_0, \hat{\mathsf{r}}_0, \hat{\mathsf{r}}_1)) \end{array} \middle| \begin{array}{c} \mathsf{ct}_0 = \mathsf{mmPKE}'.\mathsf{mmEnc}^{\mathsf{i}}(\mathsf{pp}'; \mathsf{r}_0) \wedge \\ \hat{\mathsf{ct}}_0 = \mathsf{mmPKE}'.\mathsf{mmEnc}^{\mathsf{d}}(\mathsf{pp}', \mathsf{pk}_\beta, \mathsf{m}; \mathsf{r}_0, \hat{\mathsf{r}}_\beta) \wedge \\ \hat{\mathsf{ct}}_1 = \mathsf{mmPKE}'.\mathsf{mmEnc}^{\mathsf{d}}(\mathsf{pp}', \mathsf{pk}_{1-\beta}, \mathsf{m}; \mathsf{r}_0, \hat{\mathsf{r}}_{1-\beta}) \end{array} \right\}.$$

The construction of compiler $\mathsf{Comp}^{\mathsf{CCA}}[\mathsf{mmPKE}', \Pi']$ is defined in Figure 11 which outputs an $\mathsf{mmIND\text{-}CCA}^{\mathsf{Cor}}$ secure mmPKE.

The analysis of correctness and security are deferred to the full version [54] due to space constraints.

Remark D.2 (Adaptive Corruption Compiler). By removing the NIZK component from our CCA compiler, we obtain an adaptive corruption compiler that generalizes the double encryption technique [28, 33] to the mmPKE setting.