Full-Authority Data Sharing Systems: Ciphertext-Dependent Proxy Re-Encryption with Dynamic Key Generation

Haotian Yin, Jie Zhang, Wanxin Li, Yuji Dong, Eng Gee Lim, Dominik Wojtczak

***Abstract*—Proxy re-encryption (PRE) is a powerful primitive for secure cloud storage sharing. Suppose Alice stores encrypted datasets (ciphertexts) in a cloud server (proxy). If Bob requests data sharing, Alice shares the ciphertexts by computing and sending a re-encryption key to the proxy, which will perform the re-encryption operation that generates the ciphertexts that are decryptable to Bob. Still, the proxy cannot access the plaintexts/datasets. Traditionally, the re-encryption key can convert all of Alice's ciphertexts, and the proxy should operate the re-encryption on the ciphertexts selected by the users (Alice/Bob). There is a trust issue: Alice must grant full decryption rights (losing control) to rely on proxy-enforced access policies (vulnerable to collusion). Existing PRE schemes fail to reconcile fine-grained control with collusion resistance. If Alice uses different keys to encrypt each dataset, the re-encryption complexity is linear to the number of requested datasets. We propose full-authority data sharing, a novel paradigm combining ciphertext-dependent PRE (cdPRE) and dynamic key generation (dKG). Unlike traditional PRE, cdPRE binds re-encryption keys to specific ciphertexts, ensuring collusion resistance (i.e., proxy + Bob cannot access unauthorised data). dKG dynamically connects keys via key derivation functions; for example, the chain system reduces per-dataset delegation cost to $O(1)$ for sequential release in publication/subscription systems (vs. $O(k)$ in trivial solutions, where k is the number of datasets). We instantiate this paradigm with Kyber (NIST-PQC standardised) and AES, prove its security, and experimentally verify the high efficiency of the scheme.**

***Index Terms*—Proxy Re-Encryption, Cloud Data Sharing, Lattice-Based Cryptography**

I. Introduction

PUBLIC key encryption provides a convenient way for encryption, enabling any sender to access a *public* key for encryption and send the ciphertext to an intended receiver *non-interactively*. Blaze, Bleumer, and Strauss [1] extended the public key encryption with a *re-encryption* property, where a delegator can generate a re-encryption key that transforms ciphertexts to be decryptable by other receivers without revealing the plaintext to the proxy.

This work was funded by XJTLU Research Development Fund under Grant No. RDF-21-02-014 and XJTLU Teaching Development Fund under Grant No. TDF2223-R25-207. The author Haotian Yin thanks Di Liu at XJTLU for the initial discussion and support of the work, and Yiming Lin, Yuxin Xia at XJTLU for discussing the key generation of the KDF tree. *(Corresponding author: Jie Zhang, Wanxin Li)*

Haotian Yin, Jie Zhang, Wanxin Li, and Eng Gee Lim are with School of Advanced Technology, Xi'an Jiaotong-Liverpool University, Suzhou 215000, China (email: haotian.yin23@student.xjtlu.edu.cn, {jie.zhang01, wanxin.li, enggee.lim}@xjtlu.edu.cn)

Yuji Dong is with School of Internet of Things, Xi'an Jiaotong-Liverpool University, Suzhou 215000, China (email: yuji.dong02@xjtlu.edu.cn)

Dominik Wojtczak is with Department of Computer Science, University of Liverpool, Liverpool L69 7ZX, United Kingdom (email: d.wojtczak@liverpool.ac.uk)

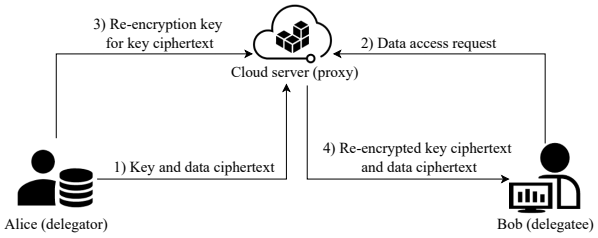

Fig. 1. Key and data ciphertext paradigm for the data sharing system

Today, proxy re-encryption (PRE) has proven to be a handy tool in data sharing scenarios. Besides the general cloud storage sharing, PRE also plays a role in publish/subscribe (pub/sub) systems [2], [3] and distributed file systems [4]. This paper is interested in the *unidirectional* and *non-interactive* PREs. If the delegator Alice authorises Bob to decrypt her ciphertext, in an unidirectional PRE, the proxy can only convert Alice's ciphertexts into Bob's, but not the reverse (convert Bob's ciphertexts into Alice's). Non-interactive means that Alice can calculate the re-encryption key from Bob's public key without establishing a secret channel with Bob. We consider the *hybrid encryption* paradigm [5], where the public key is used for encrypting the *symmetric key*, and the symmetric key is the key used for encrypting the actual datasets. This mode combines the high efficiency of symmetric encryption and the convenient key distribution of public key encryption; this is also aligned with the practical application. In most cases, encrypting data directly with a public key is prohibitively expensive.

See Fig. 1 for a general workflow of the data sharing system. Key ciphertext refers to the ciphertext generated by encrypting the symmetric key with Alice's public key; data ciphertext refers to the ciphertext generated by encrypting the dataset with the symmetric key; re-encrypted key ciphertext is decryptable to Bob, i.e., encrypted by Bob's public key.

A. Research question

Cohen [6] pointed out that Bob in PRE of [2] will reveal Alice's secret key, which is undesirable. One may ask, since Alice has agreed to transfer the decryption rights to Bob, is there any additional risk in revealing the secret key? Yes, if Alice wants to give all ciphertext decryption rights to Bob, in this application scenario, directly providing the private key to Bob will not impact Alice's interests. However, this scenario eliminates the possibility of fine-grained access control because Bob can access all ciphertexts without

restriction. Therefore, the secret key leakage is undesirable if Alice only allows Bob to read a specific subset of the data. Then, Cohen built the indistinguishability against honest re-encryption attacks (HRA[1]) security on top of chosen plaintext attacks (CPA) model to fill this security gap. In the HRA security model, the adversary can query an oracle that outputs an honest, re-encrypted ciphertext; therefore, HRA is strictly stronger than CPA security.

HRA is becoming a new standard for PRE schemes [7]–[9]. Recall that PRE is some kind of "access control", where only Bob can access the data but not the proxy. More specifically, Alice imposes this access control by generating a re-encryption key from her secret key and Bob's public key pk_B. This re-encryption key can convert her ciphertexts to ones decryptable by the party with the corresponding secret key sk_B of pk_B. Therefore, the proxy, which has no secret key sk_B, cannot access the data. In the scenario where HRA matters, there is a second "access control" against Bob and is imposed by the proxy, who will re-encrypt the specified ciphertext for Bob according to Alice's instructions. The first "access control" specifies who can decrypt, which is controlled by Alice, the delegator; the second "access control" specifies which ciphertexts can be re-encrypted, which is controlled by the *proxy*. This interesting difference introduces our main research question:

> *Can Alice control both the delegation and the re-encryption authorities?*

This is not only a theoretical interest. From a practical point of view, if Alice can control both authorities, the security will still be maintained when the proxy and Bob collude. Collusion attacks are considered in literature [4], [10], but HRA is an orthogonal notion to collusion resistance. We will discuss more details about the collusion and HRA security in the related work.

B. Full-authority data sharing

We first consider a simple situation for building our full-authority data sharing paradigm. Bob wants a copy of Alice's data, which is stored in a cloud server and encrypted by the symmetric key sek_A chosen by Alice. Alice can use public key encryption to encrypt sek_A under Bob's public key and send the *key ciphertext* to Bob. After decrypting this key ciphertext, Bob can decrypt the *data ciphertext* from the cloud server. It seems perfect if Bob requests only one dataset. However, if Bob wants k datasets, the problem becomes complex. If Alice encrypts all datasets using the same symmetric key, she only needs to delegate one key ciphertext to Bob, but this is the *all-or-nothing* approach, losing the fine-grained access control. This situation is similar to the concatenation scheme [6] (an example to explain the insufficiency of CPA security) where the re-encryption key $\mathsf{rk} \leftarrow \mathsf{Enc}(\mathsf{pk}_B, \mathsf{sk}_A)$ and the re-encryption algorithm concatenates the ciphertext and the re-encryption key together, except that the re-encryption key is an encryption of a symmetric key ($\mathsf{rk} \leftarrow \mathsf{Enc}(\mathsf{pk}_B, \mathsf{sek}_A)$) in our discussion. So, what if Alice uses different symmetric keys to encrypt each dataset? In this case, the computation complexity (of re-encryption key generation and re-encryption) and bandwidth consumption are linear to the number of datasets, i.e., $\Theta(k)$, if the re-encryption key generation is trivially encrypting each symmetric key under Bob's public key. The proxy concatenates the data ciphertext with the corresponding key ciphertext. It isn't pleasant, but Alice *does* control both the delegation and the re-encryption authorities.

[1]In this paper, we only consider indistinguishable security, so we use HRA to indicate the security notion of indistinguishability against HRA for short, instead of IND-HRA.

C. Contribution

We currently see no promising future in the above trivial scheme since, in most scenarios, Bob will request a large amount of data or face ongoing demand for the datasets. But what if the symmetric keys are *dependent*? Suppose in a pub/sub scenario, Alice, as a publisher, publishes datasets for her subscribers in each *epoch*. Bob subscribes to the data service provided by Alice on the current epoch i. For future datasets $\geq i$, Bob should be able to access the data, but not the previous ones, since he starts subscription on epoch i. Inspired by the double ratchet algorithm [11], we employ a key derivation function (KDF) chain to implement this access control. The KDF generates the symmetric key for the current epoch with input from the derivation key generated from the previous epoch, i.e., all symmetric keys are chained chronologically in epochs.

Although the KDF chain is helpful in pub/sub scenarios, if we need more refined access control, the chain structure will force us to fork it, returning to the trivial solution's efficiency problem (Subsection I-B). To enable efficient fine-grained ciphertext control, inspired by unique-path identity-based encryption [12], we employ the KDF tree to provide *adaptive* control. Note that in this situation, we need all datasets to be labelled with a unique identifier, representing that the corresponding symmetric key of the leaf node in the tree encrypts it. We call the key generation from the KDF chain or tree *dynamic* key generation (dKG).

Another key management issue arises: if Alice needs to encrypt symmetric keys for Bob, Alice must have the key! So, if Alice needs to store every symmetric key as plaintext locally, this increases the leakage threat. To provide a reasonable framework, we still need the re-encryption functionality, but a *ciphertext-dependent* one to control the delegation of ciphertexts. Proxy can only use the re-encryption key to re-encrypt one *specified* ciphertext. In addition, traditional cryptosystems are insecure due to Shor's algorithm, which cracks the discrete logarithm and factoring problems using a quantum computer. Even though we don't know when quantum computers will appear, a "harvest now, decrypt later" strategy could threaten current data privacy. We construct our ciphertext-dependent PRE (cdPRE) based on Kyber [13], an NIST standardised public key encryption scheme, which is lattice-based, post-quantum secure, and efficient.

We summarise the three-fold contribution:

- Define the full-authority data sharing systems from the cdPRE scheme with dKG;
- Construct, prove the security of, and implement the KDF chain and tree-based dKG algorithms.

- Construct, prove the security of, implement, and test the parameters of a post-quantum secure cdPRE scheme;

D. Organization

This paper is organised as follows. Next, Section II summarises the related work; necessary notations, cryptographic primitives, and lattice assumption are introduced in Section III; Section IV shows a general data sharing system, the component (cdPRE) of which is constructed by the following Section V, where the security is also proved; Section VI provides computational and communication costs evaluation and experimental comparison with the benchmark; finally, Section VII summarises the future work.

II. Related work

Polyakov, Rohloff, Sahu, and Vaikuntanathan [2] proposed two fast proxy re-encryption schemes for pub/sub systems. One is based on the NTRU encryption scheme [14] (with security reduction from Stehlé and Steinfeld [15], provably secure at the cost of the performance advantage of NTRU), the other is based on the BV homomorphic encryption scheme [16]. Unfortunately, this scheme has been proven not HRA secure by Cohen [6] since Bob will obtain Alice's secret key given an honest re-encrypted ciphertext. From our point of view, the schemes from [2] eliminate the possibility of access control to the ciphertext. The follow-up work [3] is stuck in the same problem.

As we mentioned in the introduction, the HRA-secure PRE scheme relies on access control by the proxy; it is reasonable to consider that all ciphertexts are decryptable to Bob if he colludes with the proxy, i.e., obtains the re-encryption key. Therefore, all ciphertexts are rendered unchallengable in the security proof. This situation reminds us of another critical security notion, master key security [4], capturing the situation when Bob colludes with the proxy to get Alice's secret (master) key. This notion is further formalised as a decisional version of the ciphertext by Zhou, Liu, Han, and Zhang [10]. No matter how this collusion resistance is formalised, the key point to fighting against collusion is to determine whether the adversary can obtain a plaintext other than the plaintext that can be obtained through legal re-encryption through collusion. This work provides a new paradigm to tackle this problem.

We found that those collusion-resistant PRE schemes [4], [10] are all *single-hop*, i.e., the ciphertext can be re-encrypted only once. Their strategy is mapping the *fresh* ciphertext (generated from the encryption algorithm) to another ciphertext space via re-encryption. Therefore, the adversary cannot crack the re-encrypted ciphertext since there is no efficient function reversing the mapping. This work defines the *ciphertext-dependent* PRE schemes, providing reasonable collusion-resistant and *multi-hop* PRE in this framework. We also found many PRE schemes with various functionalities [17], [18]. Still, their security is based on the intractability of bilinear pairing problems, which do not hold against quantum computers. A blockchain-based data trading solution [19] had similar goals as ours, but their ciphertext-dependent PRE is also bilinear pairing-based, and their solution can only work for one-to-one ciphertext transformation, which is quite restricted in most applications. To our knowledge, our scheme is the first post-quantum secure and ciphertext-dependent PRE.

III. Preliminaries

This section introduces some necessary notations and cryptographic notions.

A. Notations

For $n \in \mathbb{N}$, let $[n] := \{0, 1, \ldots, n-1\}$, and let $\lambda \in \mathbb{N}$ be the security parameter. For a finite set $\mathcal{S}$, we denote the process of sampling s uniformly from $\mathcal{S}$ by $s \leftarrow \mathcal{S}$. We use $\perp$ to indicate that an algorithm terminates with an error and A^{B} when A has access to call B, where B may return $\top$ as a distinguished special symbol. We say that an algorithm A runs in probabilistic polynomial time (PPT) if the running time of A is polynomial in the security parameter λ (sometimes implicitly). Define a negligible function $\mathsf{negl}(\cdot)$: there exists a positive integer N such that for all $n > N$, $|\mathsf{negl}(n)| < 1/\mathsf{poly}(n)$ for every positive polynomial $\mathsf{poly}(\cdot)$. The empty string is specified as ε.

B. Cryptographic Primitives

1) Symmetric key encryption: A symmetric key encryption SE compromises of three algorithms: $\mathsf{KG}(\lambda) \to \mathsf{sek}$: on taking the security parameter as an input, it outputs a key sek; $\mathsf{Enc}(\mathsf{sek}, m) \to \mathsf{ct}$: on taking the key sek and message m as input, it outputs a ciphertext ct; $\mathsf{Dec}(\mathsf{sek}, \mathsf{ct}) \to m$: on taking the key sek and ciphertext ct as input, it outputs the message m.

The correctness of SE requires that for all security parameters λ it holds that

$$\Pr\left[m = \mathsf{Dec}(\mathsf{sek}, \mathsf{ct}) : \begin{array}{c} \mathsf{sek} \leftarrow \mathsf{KG}(\lambda), \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{sek}, m) \end{array}\right] = 1.$$

A symmetric encryption scheme SE is said to be indistinguishable-secure under Chosen-Plaintext Attacks (CPA) if any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ such that

$$\Pr\left[b = b' : \begin{array}{c} \mathsf{sek} \leftarrow \mathsf{KG}(\lambda), b \leftarrow \{0,1\} \\ (m_{i0}, m_{i1}) \leftarrow \mathcal{A}_1(\lambda), \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{sek}, m_{ib}), b' \leftarrow \mathcal{A}_2(\mathsf{ct}_i) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

We will construct the dynamic key generation algorithm dKG in Section IV, which will replace the algorithm $\mathsf{SE.KG}$.

2) Proxy re-encryption: Next, we define the *ciphertext-dependent* PRE scheme.

A cdPRE scheme cdPRE compromises five algorithms: $\mathsf{KG}(\lambda) \to (\mathsf{sk}, \mathsf{pk})$: on taking the security parameter as an input, it outputs a key pair $(\mathsf{sk}, \mathsf{pk})$; $\mathsf{Enc}(\mathsf{pk}, m) \to \mathsf{ct}$: on taking a public key pk and message m as input, it outputs a ciphertext ct; $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to m/\perp$: on taking the key sk and ciphertext ct as input, it outputs the message m if decryption success, otherwise $\perp$; $\mathsf{RKG}(\mathsf{sk}_i, \mathsf{pk}_j, \mathsf{ct}_i) \to \mathsf{rk}_{i \xrightarrow{\mathsf{ct}_j} j}$: on taking the private key sk_i of i, the public key pk_j of j, and a ciphertext ct_i encrypted under key of i, it outputs a re-encryption key $\mathsf{rk}_{i \xrightarrow{\mathsf{ct}_j} j}$; $\mathsf{REnc}(\mathsf{rk}_{i \xrightarrow{\mathsf{ct}_j} j}, \mathsf{ct}_i) \to \mathsf{ct}_j/\perp$: on taking

the re-encryption key rk and a ciphertext ct as input, it outputs a ciphertext ct decryptable under the secret key sk_j, or the error symbol $\perp$.

The correctness of cdPRE requires that

$$\Pr\left[m = \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) : \begin{matrix}(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KG}(\lambda), \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m)\end{matrix}\right] = 1,$$

$$\Pr\left[m = \mathsf{Dec}(\mathsf{sk}_j, \mathsf{ct}_j) : \begin{matrix}(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KG}(\lambda), \\ (\mathsf{sk}_j, \mathsf{pk}_j) \leftarrow \mathsf{KG}(\lambda), \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, m), \\ \mathsf{rk}_{i \xrightarrow{\mathsf{ct}_i} j} \leftarrow \mathsf{RKG}(\mathsf{sk}_i, \mathsf{pk}_j, \mathsf{ct}_i), \\ \mathsf{ct}_j \leftarrow \mathsf{REnc}(\mathsf{rk}_{i \xrightarrow{\mathsf{ct}_i} j}, \mathsf{ct}_i)\end{matrix}\right] = 1.$$

The above correctness can be easily extended to a multi-hop setting.

Is the CPA secure enough for data sharing? Unfortunately, CPA-secure PRE might enable Bob to recover Alice's secret key [6], which is undesirable. However, Chosen Ciphertext Attacks (CCA) security is also unnecessary since Alice will not *directly* decrypt anything for anyone in the data sharing application, i.e., there is no need to provide a decryption oracle for adversaries in the security model. To provide an adequate security guarantee, we need the receiver to gain nothing besides the honest re-encrypted ciphertext, which is just the security against honest re-encryption attacks (HRA) [6] (ciphertext-independent version). We defer the formal notion of HRA to Section V-D, where it is presented together with the security proof.

3) Key derivation function: Key derivation function (KDF) is usually based on cryptographic hash functions (SHA-256) or HMAC, constructs a pseudo-random function, and extends the input key material to the required length. This paper uses the KDF to extend the input key material to double its length. Let $\mathcal{K}$ be the input key material space, $\mathcal{S}$ be the salt space, and λ be the length of key $k \in \mathcal{K}$. We need a KDF $\mathsf{KDF} : \mathcal{K} \times \mathcal{S} \to \{0,1\}^{2\lambda}$ to be pseudo-random, i.e., computationally indistinguishable from a uniformly random distribution $U = \{u\}_{u \in \{0,1\}^{2\lambda}}$. Let $D = \{\mathsf{KDF}(k, s)\}_{k \in \mathcal{K}, s \in \mathcal{S}}$, for any PPT distinguisher $\mathcal{A}$:

$$|\Pr[\mathcal{A}(D) = 1] - \Pr[\mathcal{A}(U) = 1]| \leq \mathsf{negl}(\lambda).$$

C. Lattice assumption

Definition 1 (Module-LWE)**.** *For an algorithm $\mathcal{A}$ with running time polynomial in λ, we define $\mathsf{Adv}^{mlwe}_{\mathcal{A},m,k,\eta}(\lambda)$ be the absolute value of*

$$\Pr\left[b' = 1 : \begin{matrix}\mathbf{A} \leftarrow R_q^{m \times k}, (\mathbf{s}, \mathbf{e}) \leftarrow \beta_\eta^k \times \beta_\eta^m, \\ \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}, b' \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{b})\end{matrix}\right] -$$
$$\Pr\left[b' = 1 : \mathbf{A}, \mathbf{b} \leftarrow R_q^{m \times k} \times R_q^m, b' \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{b})\right].$$

Our scheme is based on the hardness assumption of module-LWE (module-learning with errors, MLWE) [13]. That is, $\mathsf{Adv}^{\mathrm{mlwe}}_{\mathcal{A},m,k,\eta}(\lambda)$ is negligible.

IV. Full-Authority Data Sharing Systems

This section introduces three full-authority frameworks for data sharing.

A. Entities

We define four main entities in our system:

- **Delegator**: Delegator is an individual or organisation that owns or generates data. It has ownership and control over the data and can decide how to share it.
- **Delegatee**: Delegatee is an individual or organisation that wants to acquire data to meet specific needs. It may need data for analysis, research, product development, or market decision-making.
- **Proxy**: A remote server provides data storage and management services. It allows users (delegators and delegatees) to store and access data.

Let Bob be a delegatee, and Alice be a delegator, as in the introduction section. In the last parts, we will use these two names to refer to their identities.

B. Naive system

In this naive sharing system, the delegatee encrypts all its data with the same symmetric key.

Let $(\mathsf{pk}_A, \mathsf{sk}_A), (\mathsf{pk}_B, \mathsf{sk}_B) \leftarrow \mathsf{cdPRE.KG}(\lambda)$ be the key-pairs of Alice and Bob, respectively. Alice encrypts all her data via one symmetric key, $\mathsf{sek} \leftarrow \mathsf{SE.KG}(\lambda), \mathsf{ct}_{mi} \leftarrow \mathsf{SE.Enc}(\mathsf{sek}, m_i)$. Due to the key management issues, Alice encrypts this symmetric key with her public key, $\mathsf{ct}_k \leftarrow \mathsf{cdPRE.Enc}(\mathsf{pk}_A, \mathsf{sek})$, and uploads the key ciphertext and the data ciphertext to the proxy. After Bob requests data access, Alice generates a re-encryption key, $\mathsf{rk}_{A \xrightarrow{\mathsf{ct}_k} B} \leftarrow \mathsf{cdPRE.RKG}(\mathsf{sk}_A, \mathsf{pk}_B, \mathsf{ct}_k)$, and sends the re-encryption key to the proxy. The proxy re-encrypts the key ciphertext for Bob, $\mathsf{ct}'_k \leftarrow \mathsf{REnc}(\mathsf{rk}_{A \xrightarrow{\mathsf{ct}_k} B}, \mathsf{ct}_k)$. Bob can decrypt the key ciphertext first, obtain the symmetric encryption key sek, then Bob uses this key sek to decrypt the data ciphertext ct_{mi} to obtain the plaintext.

C. Chain system

The main idea is to chain the symmetric keys via KDF, see Fig. 2.

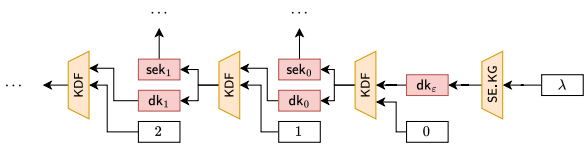

Fig. 2. KDF chain-based dKG

Let $(\mathsf{pk}_A, \mathsf{sk}_A), (\mathsf{pk}_B, \mathsf{sk}_B) \leftarrow \mathsf{cdPRE.KG}(\lambda)$ be the key-pairs of Alice and Bob, respectively, and n be the number of total datasets. Alice labels her data sequentially with epoch identifier $i \in [n]$. For the first data with $i = 0$, Alice generates a fresh symmetric key $\mathsf{dk}_\varepsilon \leftarrow \mathsf{SE.KG}(\lambda)$, generates the symmetric encryption key and derivation key $(\mathsf{sek}_0, \mathsf{dk}_0) \leftarrow \mathsf{KDF}(\mathsf{dk}_\varepsilon, 0)$, and encrypts the data $\mathsf{ct}_{m0} \leftarrow \mathsf{SE.Enc}(\mathsf{sek}_0, m_0)$. Alice encrypts this symmetric key $\mathsf{ct}_{k0} \leftarrow \mathsf{cdPRE.Enc}(\mathsf{pk}_A, \mathsf{k})$ and stores the tuple $(i = 0, \mathsf{ct}_{k0}, \mathsf{ct}_{m0})$ on the proxy. For all future data storage, Alice generates

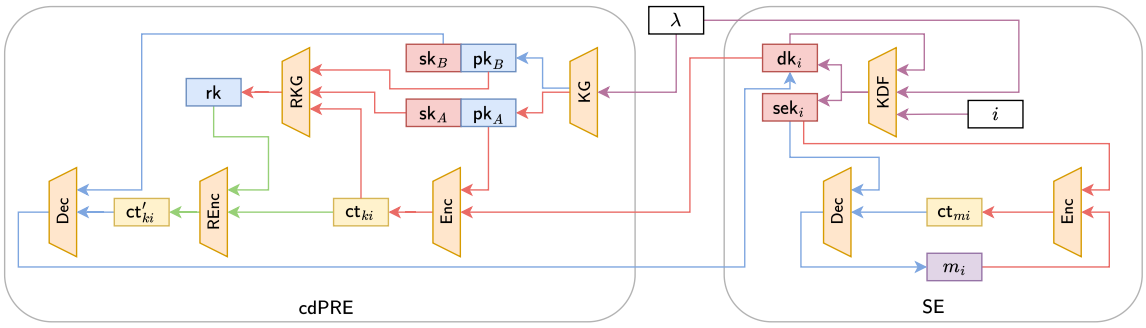

Fig. 3. A key schedule diagram of the chain system for $i \geq 1$. The red lines refer to the operation of Alice, the blue lines refer to the operation of Bob, the purple lines refer to the operation of both Alice and Bob, and the green lines refer to the operation of the proxy

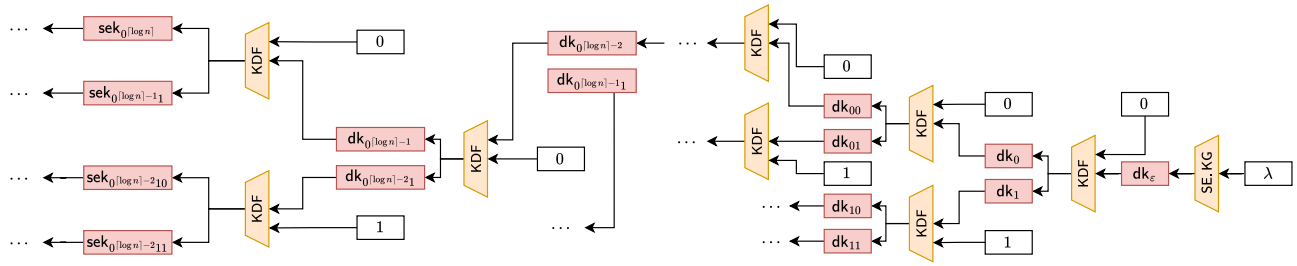

Fig. 4. KDF tree-based dKG

a symmetric encryption key and a derivation key from the previous derivation key $(\mathsf{sek}_i, \mathsf{dk}_i) \leftarrow \mathsf{KDF}(\mathsf{dk}_{i-1}, i)$, encrypts the data $\mathsf{ct}_{mi} \leftarrow \mathsf{SE.Enc}(\mathsf{sek}_i, m_i)$, encrypts the derivation key $\mathsf{ct}_{ki} \leftarrow \mathsf{cdPRE.Enc}(\mathsf{pk}_A, \mathsf{dk}_{i-1})$, and uploads the tuple $(i, \mathsf{ct}_{ki}, \mathsf{ct}_{mi})$ to the proxy. Compared with the naive system, this system enables Alice to control how many plaintexts Bob can access by a pre-set sequence. This structure is sufficient for pub/sub systems. Bob needs to specify the starting epoch he wants to access, say, i'. Alice computes the re-encryption key $\mathsf{rk}_{A \xrightarrow{\mathsf{ct}_{ki'}} B} \leftarrow \mathsf{cdPRE.RKG}(\mathsf{sk}_A, \mathsf{pk}_B, \mathsf{ct}_{ki'})$ and sends $(i', \mathsf{rk}_{A \xrightarrow{\mathsf{ct}_{ki'}} B})$ to the proxy. The proxy finds the tuple $(i', \mathsf{ct}_{ki'}, \mathsf{ct}_{mi'})$ according to identifier i' and re-encrypts $\mathsf{ct}'_{ki'} \leftarrow \mathsf{REnc}(\mathsf{rk}_{A \xrightarrow{\mathsf{ct}_{ki'}} B}, \mathsf{ct}_{ki'})$ for Bob. Bob can generate all symmetric encryption keys in this system with identifier $i \geq i'$ by himself. The key schedule diagram for $i \geq 1$ is shown in Fig. 3.

D. Tree system

The main idea is to organise the symmetric encryption keys in a tree via the KDF, see Fig. 4 for an illustration. This structure is a lying tree for a better display of the algorithm, with the root node on the far right. We will explain the structure in a traditional tree structure, that is, the root node is at the top, and the leaf nodes are generated at the bottom.

Let $(\mathsf{pk}_A, \mathsf{sk}_A), (\mathsf{pk}_B, \mathsf{sk}_B) \leftarrow \mathsf{cdPRE.KG}(\lambda)$ be the key-pairs of Alice and Bob, respectively, and n be the number of total datasets. We build a binary KDF tree from the root node (the rightmost node in Fig. 4) to the bottom. Each leaf node is an encryption key for the epoch, and we label them from left to right (from top to bottom in Fig. 4) as the symmetric encryption keys with identifiers from 0 (in binary $0^{\lceil \log n \rceil}$) to $n-1$. Therefore, the tree's height (or "width" of tree in Fig. 4) is $\lceil \log n \rceil$. The entire tree is recursively generated from the root node in depth-first order (i.e., left first, then right; or, top first, then bottom in Fig. 4). Alice sequences her data with identifier $i \in [n]$. For each leaf node, Alice uses it as a symmetric encryption key to encrypt the dataset with the corresponding identifier $\mathsf{ct}_{mi} \leftarrow \mathsf{SE.Enc}(\mathsf{sek}_i, m_i)$. To enable efficient delegation, Alice encrypts each node (internal and leaf) by its public key

$$\mathsf{ct}_{ki} \leftarrow \mathsf{cdPRE.Enc}(\mathsf{pk}_A, \mathsf{sek}_i), \mathsf{ct}_{kj} \leftarrow \mathsf{cdPRE.Enc}(\mathsf{pk}_A, \mathsf{dk}_j),$$

where $i \in [n], j \in \{\varepsilon, 0, 1, 00, 01, \dots\}$. Let $\mathcal{S} := \{\varepsilon, 0, 1, 00, 01, \dots\}$, there are at most $n-1$ internal nodes in total, i.e., $|\mathcal{S}| \leq n-1$, when the tree is a full binary tree, take the equality. Alice uploads the data ciphertext tuples $\{(i, \mathsf{ct}_{mi})\}_{i \in [n]}$ and the key ciphertext tuples $\{(i, \mathsf{ct}_{ki})\}_{i \in [n]} \cup \{(j, \mathsf{ct}_{kj})\}_{j \in \mathcal{S}}$ to the proxy.

Suppose Bob requests some datasets. If Bob requests *all* datasets and Alice approves it, Alice generates the re-encryption key for the root node $\mathsf{ct}_{A \xrightarrow{\mathsf{ct}_{k\varepsilon}} B} \leftarrow \mathsf{PRE.RKG}(\mathsf{sk}_A, \mathsf{pk}_B, \mathsf{ct}_{k\varepsilon})$ and sends the re-encryption key to the proxy. Upon receiving the re-encryption key, the proxy re-encrypts the key ciphertext $\mathsf{ct}'_{k\varepsilon} \leftarrow \mathsf{PRE.REnc}(\mathsf{ct}_{A \xrightarrow{\mathsf{ct}_{k\varepsilon}} B}, \mathsf{ct}_{k\varepsilon})$ for Bob. Bob can decrypt it and generate the same KDF tree. If Bob requests access to a proper subset $\mathcal{R}$ of datasets $[n]$, Alice finds the cover set of this set $\mathcal{R}$. More specifically, starting from the leaf node in $\mathcal{R}$, merge sub-trees from bottom to top by traversing the tree in post-order: if both sibling nodes are covered, replace them with their parent nodes. Let the

final node set be $\mathcal{R}'$. Alice needs to delegate all nodes in set $\mathcal{R}'$, and the proxy accordingly re-encrypts the (derivation or encryption) key ciphertexts. The complexity of delegation depends on the distribution of selected nodes. In the worst case, $|\mathcal{R}'| = |\mathcal{R}|$, there are no nodes merged. In the best case, $|\mathcal{R}'| = 1$, all nodes are merged into one node, i.e., belonging to the same sub-tree (not necessarily the root node).

1) Example: Suppose Alice builds a KDF tree for eight datasets. If Bob wants to access all datasets, i.e., $\mathcal{R} = \{0, 1, 2, \ldots, 7\}$, Alice generates the re-encryption key associated with the root node. If $\mathcal{R} = \{0, 1, 2\}$, then two keys with identifier 00 and 010 will be re-encrypted since node 000 and node 001 can be generated by their parent node 00. But if $\mathcal{R} = \{0, 1, 2, 3\}$, only one key with identifier 0 will be proxy re-encrypted because $000, 001, 010, 011$ share a common ancestor 0 and this ancestor entirely generated these leaves. Note that we cannot re-encrypt 0 for leaves $\{000, 001, 010\}$ (binaries of $\{0, 1, 2\}$) because this node will lead to the derivation of 011, which Alice does not want to reveal. The reader can access our Python code on GitHub[2] for the demo implementing chain and tree systems (using cdPRE constructed in Section V as a re-encryption scheme).

E. Security analysis

We use block cipher AES and HMAC to implement SE and KDF.

The CPA security of the chain system follows the security of block cipher [20] for $i = 0$ and the pseudo-randomness of KDF [21] for $i \in [n] \setminus 0$ (the encryption key of SE is derived from the KDF). If a decryptor (Bob) has dk_i, it cannot derive keys for epoch $i' \in [i]$ because of the KDF's one-wayness (implied by pseudo-randomness). The security claim of the tree system is analogous.

F. Comparison

Given n datasets in total, k datasets to be shared, the functionalities and re-encryption complexities are shown in Table I. Recall that "all-or-nothing" refers to the functionality that Bob can access all datasets if his request is satisfied by Alice; "pub/sub" refers to the functionality that Bob can access a series of data sets starting with a specific identifier (if his request is satisfied by Alice) not the previous ones; "adaptive" refers to the functionality that Bob can access the datasets he specified (if his request is satisfied by Alice) but not others. In this table, the asymptotic symbol represents the complexity (number of ciphertexts, number of re-encrypted keys, computations, and bandwidths) required to satisfy the functionality given k datasets to be shared; $\times$ suggests that the system cannot meet the functionality; (trust proxy) indicates the delegator in the system needs to trust the proxy to enact access control (non collusion-resistance). "Pure PRE" column indicates the traditional ciphertext-*in*dependent PRE solutions for data sharing, where a trusted proxy uses a re-encryption key to re-encrypt ciphertexts from Alice to Bob according to the specification.

[2]https://github.com/coro-afk/cdpre-from-kyber/tree/main/demo

TABLE I
COMPARISON BETWEEN DIFFERENT SHARING SYSTEMS

Functionalities	Naive	Chain	Tree	Pure PRE
All-or-nothing	$O(1)$	$O(1)$	$O(1)$	$O(1)$
Pub/sub	$\times$	$O(1)$	$O(\log k)$	$O(1)$ (trust proxy)
Adaptive	$\times$	$\times$	$O(1) - O(k)$	$O(1)$ (trust proxy)

Note that in a tree system, pub/sub functionality can be implemented with complexity upper bounded by $\log k$, because a subscription specifies a *continuous* nodes for delegation, which can be merged to sub-trees; it comes to $O(1)$, when the nodes share on common ancestor. Adaptive functionality depends on the distribution of selected nodes, as explained in the construction of the tree system. Even in the worst case, where the complexity is upper bounded by $O(k)$, only key ciphertexts are stored in Alice's local storage, but not the whole data ciphertext (sufficiently larger than the key ciphertext), nor the plaintexts of encryption keys (providing reasonable security against leakage). Recall that for the traditional use of PRE, Alice needs to trust that the proxy will enact access control that she specified for a delegation. This promise fails when Bob colludes with the proxy, but our systems resist this threat since Alice masters the access control rights to her ciphertext. All-or-nothing functionality has no such concern since all ciphertexts are accessible to Bob after delegation.

V. HRA-SECURE CDPRE FROM LATTICES

This section first introduces the notion of ciphertext-dependent CPA (cdCPA) security of PRE. Then, we construct a ciphertext-dependent PRE cdPRE from the Kyber encryption scheme from [13]. Finally, we prove our scheme cdPRE is cdCPA-secure and cdHRA-secure.

A. cdCPA security of PRE

Definition 2. *(cdCPA security of PRE) A proxy re-encryption scheme* PRE *is cdCPA-secure if for all PPT adversaries* $\mathcal{A}$ *such that*

$$\mathsf{Adv}^{\mathsf{cd\text{-}cpa}}_{\mathsf{PRE},\mathcal{A}}(\lambda) := \Pr[1 \leftarrow \mathsf{Exp}^{\mathsf{cd\text{-}cpa}}_{\mathsf{PRE},\mathcal{A}}(\lambda)] < \frac{1}{2} + \mathsf{negl}(\lambda),$$

where $\mathsf{Exp}^{\mathsf{cd\text{-}cpa}}_{\mathsf{PRE},\mathcal{A}}(\lambda)$ *is defined in Fig. 5.*

B. cdPRE from Kyber

We construct the PRE scheme cdPRE based on Kyber [13], which has been standardised by NIST. Current PRE construction [22] from Kyber is ciphertext-*in*dependent and single-hop.

The compression/decompression functions used in Kyber:

- Comp_q: For $x \in \mathbb{Z}_q$ and $d \in \mathbb{Z}$, it is defined as $\mathsf{Comp}_q(x) := \lceil (2^d/q) \cdot x \rfloor \bmod 2^d$ with parameter $q \in \mathbb{Z}$.
- Decomp_q: For $x \in \mathbb{Z}_q$ and $d \in \mathbb{Z}$, it is defined as $\mathsf{Decomp}_q(x) := \lceil (q/2^d) \cdot x \rfloor$ with parameter $q \in \mathbb{Z}$.
- The bit-decomposition algorithm $\mathsf{BitDecomp}$ given a vector $\mathbf{x} \in \mathbb{Z}_q^N$ decomposes $\mathbf{x}$ into its bit representation.

Experiment $\mathsf{Exp}^{\mathsf{cd\text{-}cpa}}_{\mathsf{PRE},\mathcal{A}}(\lambda)$:

$\mathsf{Cor} := \emptyset$ // record corrupted keys
$i := 0$ // user identifier
$i^* :=\perp$ // record challenge user
$\mathcal{O} := \{\mathcal{O}_{\mathsf{HKG}}, \mathcal{O}_{\mathsf{CKG}}, \mathcal{O}_{\mathsf{RKG}}\}$
$(i^*, m_0, m_1, st) \leftarrow \mathcal{A}_1^{\mathcal{O}}(\{\mathsf{pk}_i\}_{i\in[n]}, \{\mathsf{sk}_i\}_{i\in\mathsf{Cor}})$
$b \leftarrow \{0,1\}$
If $i^* \in \mathsf{Cor}$: // avoid trivial attacks
  Return $\perp$
$\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}_{i^*}, m_b)$
$b' \leftarrow \mathcal{A}_2^{\mathcal{O}}(st, \mathsf{ct}^*)$
Return $b = b'$

$\mathcal{O}_{\mathsf{HKG}}$:

$(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KG}(\lambda)$
Return (i, pk_i)
Increment i

$\mathcal{O}_{\mathsf{CKG}}$:

$(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KG}(\lambda)$
$\mathsf{Cor} \leftarrow \mathsf{Cor} \cup \{i\}$ // record corrupted users
Return $(i, \mathsf{pk}_i, \mathsf{sk}_i)$
Increment i

$\mathcal{O}_{\mathsf{RKG}}(i, j, \mathsf{ct}_i)$:

If ($i \notin \mathsf{Cor}$ and $j \in \mathsf{Cor}$): // avoid trivial re-encryption key generation
  Return $\perp$
Return $\mathsf{RKG}(\mathsf{sk}_i, \mathsf{pk}_j, \mathsf{ct}_i)$

Fig. 5. The cdCPA security experiment

- The powers-of-two algorithm Powersof2 with $\ell = \lceil \log q \rfloor$, on input a (column) vector $\mathbf{s} \in \mathbb{Z}_q^N$, outputs $(1, 2, \ldots, 2^\ell)^\top \otimes \mathbf{s} = (\mathbf{s}, 2\mathbf{s}, \ldots, 2^{\ell-1}\mathbf{s}) \in \mathbb{Z}_q^{N\ell}$, where $\otimes$ is the standard tensor product.

We know that

$$|\mathsf{Decomp}_q(\mathsf{Comp}_q(x, d), d) - x \bmod^{\pm} q| \leq B_q := \left\lceil \frac{q}{2^{d+1}} \right\rfloor. \tag{1}$$

$\mathsf{cdPRE} = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{RKG}, \mathsf{ReEnc})$ is shown in Fig. 6. Note that in the algorithm RKG, we compress the first re-encryption key $\mathbf{u}_{A \xrightarrow{\mathsf{ct}_A} B}$ just as much as the first ciphertext since $\mathbf{u}_{A \xrightarrow{\mathsf{ct}_A} B}$ is assigned directly to the re-encrypted ciphertext $\mathbf{u}_B$. This operation reduces the re-encryption key size but does not introduce additional error terms.

The main idea of re-encryption key generation is to re-randomise the ciphertext with the knowledge of Alice's secret key. The first element of ciphertext $\mathbf{u}_A$ is used to compute a part of the re-encryption key. Therefore, in the re-encryption algorithm, this promises that the generated re-encryption key is only valid to transform the ciphertext with specific randomness in $\mathbf{u}_A$. This randomness serves as an identifier for the ciphertext. The re-encryption algorithms are secure if the attacker cannot arrange the transformation between the ciphertexts with different identifiers.

$\mathsf{Setup}(\lambda)$:

$\mathcal{M} = \{0,1\}^\mu$ be the message space
prime $q, \ell := \lceil \log q \rceil, R := \mathbb{Z}[X]/(X^n + 1), R_q = \mathbb{Z}_q[X]/(X^n + 1)$.
ψ_η is a distribution where each coefficient of a sample is generated from B_η, where η is a positive integer
Let k, d_t, d_u, d_v be positive integers
Sample $\mathbf{A} \leftarrow R_q^{k\times k}$
Return $\mathsf{pp} = (\lambda, \mu, n, q, k, \ell, \eta_1, \eta_2, d_u, d_v, \mathbf{A})$

$\mathsf{KG}(\mathsf{pp}) \rightarrow (\mathsf{pk}, \mathsf{sk})$:

Sample $\mathbf{s} \leftarrow \psi_{\eta_1}^k$
$\mathbf{t} \leftarrow \mathbf{A}\mathbf{s} + \mathbf{e}$
$\mathsf{pk} := (\mathbf{t}, \mathbf{A}), \mathsf{sk} := \mathbf{s}$.
Return $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{Enc}(\mathsf{pk}, m) \rightarrow \mathsf{ct}$:

Parse $\mathsf{pk} = (\mathbf{t}, \mathbf{A})$
Sample $(\mathbf{r}, \mathbf{e}_1, e_2) \leftarrow \psi_{\eta_1}^k \times \psi_{\eta_2}^k \times \psi_{\eta_2}$
$\mathbf{u} \leftarrow \mathsf{Comp}_q(\mathbf{A}^\top \mathbf{r} + \mathbf{e}_1^\top, d_u)$
$v \leftarrow \mathsf{Comp}_q(\mathbf{t}^\top \mathbf{r} + e_2 + \lceil \frac{q}{2} \rfloor \cdot m, d_v)$
Return $\mathsf{ct} = (\mathbf{u}, v)$

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \rightarrow m$:

Parse $\mathsf{sk} \rightarrow \mathbf{s}, \mathsf{ct} \rightarrow (\mathbf{u}, v)$
$\mathbf{u} \leftarrow \mathsf{Decomp}_q(\mathbf{u}, d_u), v \leftarrow \mathsf{Decomp}_q(v, d_v)$
Return $m \leftarrow \mathsf{Comp}_q(v - \mathbf{s}^\top \mathbf{u}, 1)$.

$\mathsf{RKG}(\mathsf{sk}_A, \mathsf{pk}_B, \mathsf{ct}_A) \rightarrow (rk_{A \xrightarrow{\mathsf{ct}_A} B})$:

Parse $\mathsf{sk}_A = \mathbf{s}_A, \mathsf{pk}_B = (\mathbf{t}_B, \mathbf{A}), \mathsf{ct}_A = (\mathbf{u}_A, v_A)$
$\mathbf{u}_A \leftarrow \mathsf{Decomp}_q(\mathbf{u}_A, d_u)$
Sample $(\mathbf{r}', \mathbf{e}') \leftarrow \psi_{\eta_1}^k \times \psi_{\eta_2}^k$
$\mathbf{u}_{A \xrightarrow{\mathsf{ct}_A} B} \leftarrow \mathsf{Comp}_q(\mathbf{A}^\top \mathbf{r}' + \mathbf{e}' \in R_q^k, d_u)$
$v_{A \xrightarrow{\mathsf{ct}_A} B} \leftarrow \mathsf{Comp}_q(\mathbf{t}_B^\top \mathbf{r}' - \mathbf{s}_A^\top \mathbf{u}_A, d_{rv})$
Return $\mathsf{rk}_{A \xrightarrow{\mathsf{ct}_A} B} := (\mathbf{u}_{A \xrightarrow{\mathsf{ct}_A} B}, v_{A \xrightarrow{\mathsf{ct}_A} B})$

$\mathsf{REnc}(\mathsf{rk}_{A \xrightarrow{\mathsf{ct}_A} B}, \mathsf{ct}_A) \rightarrow \mathsf{ct}_B$:

Parse $\mathsf{rk}_{A \xrightarrow{\mathsf{ct}_A} B} \rightarrow (\mathbf{u}_{A \xrightarrow{\mathsf{ct}_A} B}, v_{A \xrightarrow{\mathsf{ct}_A} B}), \mathsf{ct}_A \rightarrow (\mathbf{u}_A, v_A)$
$v_A \leftarrow \mathsf{Decomp}_q(v_A, d_v)$
$v_{A \xrightarrow{\mathsf{ct}_A} B} \leftarrow \mathsf{Decomp}_q(v_{A \xrightarrow{\mathsf{ct}_A} B}, d_{rv})$
$\mathbf{u}_B \leftarrow \mathbf{u}_{A \xrightarrow{\mathsf{ct}_A} B}, v_B \leftarrow v_A + v_{A \xrightarrow{\mathsf{ct}_A} B}$
$\mathbf{v}_B \leftarrow \mathsf{Comp}_q(v_B, d_v)$
Return $\mathsf{ct}_B = (\mathbf{u}_B, v_B)$

Fig. 6. cdPRE scheme

C. Decryption Failure Rate

The correctness of Kyber encryption/decryption is analysed in Theorem 1 of [13]. Here, we prove the correctness of re-encryption. The concrete noise bound is computed in Section VI-A.

Proposition 1 (Correctness of re-encryption)**.** *Let* $\mathsf{pp} = (\lambda, \mu, n, q, k, \ell, \eta_1, \eta_2, d_u, d_v, d_{rv}, \mathbf{A})$ *be the public parameters output by* $\mathsf{Setup}(\lambda)$*. Let* A, B *be distinct users, their key pairs be* $(\mathsf{pk}_A, \mathsf{sk}_A), (\mathsf{pk}_B, \mathsf{sk}_B) \leftarrow \mathsf{KG}(\mathsf{pp})$*, and a ciphertext* $\mathsf{ct}_A \leftarrow \mathsf{Enc}(\mathsf{pk}_A, m)$*. Let* $\mathbf{c}_u \leftarrow \psi_{d_u}^k, c_v \leftarrow \psi_{d_v}, c_{rv} \leftarrow \psi_{d_{rv}}$ *be distributed according to the following distribution* ψ_d^k *over* R*:*

1) *Choose* $\mathbf{y} \leftarrow R^k$ *uniformly at random.*
2) *Return* $(\mathbf{y} - \mathsf{Decomp}_q(\mathsf{Comp}_q(\mathbf{y}, d)), d) \bmod^{\pm} q$.

Denote

$$\begin{aligned} w :=& \mathbf{e}_A^\top \mathbf{r} + e_{A,2} + c_{v,A} + \mathbf{e}_B^\top \mathbf{r}' + c_{rv} \\ & - \mathbf{s}_A^\top (\mathbf{e}_{A,1} + \mathbf{c}_{u,A}) + c_{v,B} - \mathbf{s}_B^\top (\mathbf{e}' + \mathbf{c}_{u,B}), \\ \delta :=& \Pr\left[\|w\|_\infty \geq \lceil q/4 \rfloor\right]. \end{aligned}$$

Then cdPRE *is* $(1-\delta)$*-correct.*

Proof. The public keys $\mathbf{t}_A, \mathbf{t}_B$ is represented as

$$\begin{aligned} \mathbf{t}_A &= \mathbf{A}\mathbf{s}_A + \mathbf{e}_A, \\ \mathbf{t}_B &= \mathbf{A}\mathbf{s}_B + \mathbf{e}_B \end{aligned}$$

The value $\mathbf{u}_A$ of the ciphertext $\mathsf{ct}_A = (\mathbf{u}_A, v_A)$ under the public key pk is

$$\begin{aligned} \mathbf{u}_A &= \mathsf{Decomp}_q(\mathsf{Comp}_q(\mathbf{A}^\top \mathbf{r}_A + \mathbf{e}_{A,1}, d_u), d_u) \\ &= \mathbf{A}^\top \mathbf{r}_A + \mathbf{e}_{A,1} + \mathbf{c}_{u,A} \end{aligned}$$

for some $\mathbf{c}_{u,A} \in R^k$. And the value v_A is

$$\begin{aligned} v_A &= \mathsf{Decomp}_q(\mathsf{Comp}_q(\mathbf{t}_A^\top \mathbf{r} + e_{A,2} + \left\lceil \frac{q}{2} \right\rfloor \cdot m, d_v), d_v) \\ &= (\mathbf{A}\mathbf{s}_A + \mathbf{e}_A)^\top \mathbf{r} + e_{A,2} + c_{v,A} + \left\lceil \frac{q}{2} \right\rfloor \cdot m, \end{aligned}$$

for some $c_{v,A} \in R$. A re-encryption key $\mathsf{rk}_{A \xrightarrow{\mathsf{ct}_A} B} = (\mathbf{u}_{A \xrightarrow{\mathsf{ct}_A} B}, v_{A \xrightarrow{\mathsf{ct}_A} B})$ is generated by $\mathsf{RKG}(\mathsf{sk}_A, \mathsf{pk}_B, \mathsf{ct}_A)$, and the values of $\mathsf{rk}_{A \xrightarrow{c_A} B}$ are

$$\begin{aligned} \mathbf{u}_{A \xrightarrow{c_A} B} &= \mathbf{A}^\top \mathbf{r}' + \mathbf{e}' \in R_q^k, \\ v_{A \xrightarrow{c_A} B} &= \mathbf{t}_B^\top \mathbf{r}' - \mathbf{s}_A^\top \mathbf{u}_A. \end{aligned}$$

The re-encrypted ciphertext $\mathsf{ct}_B = (\mathbf{u}_B, v_B)$ is generated by $\mathsf{REnc}(\mathsf{rk}_{A \xrightarrow{\mathsf{ct}_A} B}, \mathsf{ct}_A)$, and the values of ct_B are

$$\begin{aligned} \mathbf{u}_B &= \mathsf{Decomp}_q(\mathsf{Comp}_q(\mathbf{u}_B, d_u), d_u) \\ &= \mathsf{Decomp}_q(\mathsf{Comp}_q(\mathbf{u}_{A \xrightarrow{c_A} B}, d_u), d_u) \\ &= \mathbf{A}^\top \mathbf{r}' + \mathbf{e}' + \mathbf{c}_{u,B}, \\ v_B &= \mathsf{Decomp}_q(\mathsf{Comp}_q(v_B, d_v), d_v) \\ &= v_A + \mathsf{Decomp}_q(\mathsf{Comp}_q(v_{A \xrightarrow{c_A} B}, d_{rv}), d_{rv}) + c_{v,B} \\ &= v_A + \mathbf{t}_B^\top \mathbf{r}' - \mathbf{s}_A^\top \mathbf{u}_A + c_{rv} + c_{v,B}, \end{aligned}$$

for some $(\mathbf{c}_{u,B}, c_{v,B}) \in R^k \times R$. Substituting the value of $v_A, \mathbf{t}_B, \mathbf{u}_A$, we have

$$\begin{aligned} v_B =& (\mathbf{A}\mathbf{s}_A + \mathbf{e}_A)^\top \mathbf{r} + e_{A,2} + c_{v,A} + \left\lceil \frac{q}{2} \right\rfloor \cdot m \\ & + (\mathbf{A}\mathbf{s}_B + \mathbf{e}_B)^\top \mathbf{r}' - \mathbf{s}_A^\top (\mathbf{A}^\top \mathbf{r} + \mathbf{e}_{A,1}) \\ & - \mathbf{s}_A^\top \mathbf{c}_{u,A} + c_{rv} + c_{v,B}, \\ \mathbf{s}_B^\top \mathbf{u}_B =& \mathbf{s}_B^\top (\mathbf{A}^\top \mathbf{r}' + \mathbf{e}') + \mathbf{s}_B^\top \mathbf{c}_{u,B}. \end{aligned}$$

Therefore, we have

$$\begin{aligned} v_B - \mathbf{s}_B^\top \mathbf{u}_B =& \mathbf{e}_A^\top \mathbf{r} + e_{A,2} + c_{v,A} + \left\lceil \frac{q}{2} \right\rfloor \cdot m + \mathbf{e}_B^\top \mathbf{r}' + c_{rv} \\ & - \mathbf{s}_A^\top (\mathbf{e}_{A,1} + \mathbf{c}_{u,A}) + c_{v,B} - \mathbf{s}_B^\top (\mathbf{e}' + \mathbf{c}_{u,B}). \end{aligned}$$

Let

$$\begin{aligned} w &:= v_B - \mathbf{s}_B^\top \mathbf{u}_B - \left\lceil \frac{q}{2} \right\rfloor \cdot m, \\ m' &:= \mathsf{Comp}_q(v_B - \mathbf{s}_B^\top \mathbf{u}_B, 1). \end{aligned}$$

Hence, if $\|w\|_\infty < \left\lceil \frac{q}{4} \right\rfloor$, by inequality 1, we have $\left\lceil \frac{q}{4} \right\rfloor \geq \|v_B - \mathbf{s}_B^\top \mathbf{u}_B - \left\lceil \frac{q}{2} \right\rfloor \cdot m'\|_\infty = \|w + \left\lceil \frac{q}{2} \right\rfloor \cdot m - \left\lceil \frac{q}{2} \right\rfloor \cdot m'\|_\infty$. By the triangle inequality and the assumption $\|w\|_\infty < \left\lceil \frac{q}{4} \right\rfloor$, we have

$$\left\| \left\lceil \frac{q}{2} \right\rfloor \cdot (m - m') \right\|_\infty < 2 \cdot \left\lceil \frac{q}{2} \right\rfloor,$$

which indicates[3] $m = m'$. Therefore, the correctness of re-encryption follows. □

Our scheme supports multi-hop re-encryption since the re-encrypted ciphertext has the same distribution as the non-re-encrypted ciphertext, except that the noise keeps increasing. If the noise goes wild, the recipient cannot decrypt the ciphertext correctly. One may use the bootstrap strategy [23] to refresh the ciphertext. Our scheme is sufficient for a data-sharing application, so we leave this problem untouched.

D. Security analysis

We prove that the above scheme is cdCPA-secure, re-encryption simulatable, and cdHRA-secure.

Theorem 1. *If the MLWE problem is intractable, given any PPT adversary* $\mathcal{A}$*,* cdPRE *is cdCPA-secure.*

Proof. (Outline) Let $\mathcal{A}$ be an adversary executed in the cdCPA security experiment shown in Fig. 5. Suppose there is a Module-LWE solver $\mathcal{B}$ with input $(\mathbf{A}^*, \mathbf{b}^*)$, and we pick an honest user as the challenge user and set its public key to be $\mathbf{b}^*$ (we do not have the secret key). On simulating $\mathcal{O}_{\mathsf{RKG}}$, all directions of re-encryption keys are simulatable according to the scheme except the one from i^* to an honest user i since we do not have the secret key of i^*. We generate the re-encryption key for some ciphertext ct by uniformly random $\mathbf{u}_{i^* \xrightarrow{\mathsf{ct}} i}, v_{i^* \xrightarrow{\mathsf{ct}} i} \leftarrow R_q^k \times R_q$. This change is indistinguishable because we can build a CPA adversary to crack Kyber: in the real scheme, rk is a kind of encryption of $-\mathbf{s}_i^\top \mathbf{u}_i$ (as a plaintext) by i's public key $\mathbf{t}_i$; therefore, the indistinguishability stems from the CPA security of Kyber; we have known that MLWE can be reduced to the security of Kyber [13].

[3] One can verify it by two cases: $q \equiv 1 \bmod 4$ and $q \equiv 3 \bmod 4$.

If the adversary outputs the challenger user identifier i^*, we output a uniformly random chosen ciphertext $(\mathbf{u}^*, v^*) \leftarrow R_q^k \times R_q$ and pick a uniformly random coin b. Similarly, if the adversary can win the security game with non-negligible probability, we can build a Kyber CPA adversary on it. Therefore, cdPRE is cdCPA-secure. □

Next, we show that cdPRE is re-encryption simulatable.

Definition 3 (Re-encryption Simulatability)**.** *A PRE scheme* PRE *is re-encryption simulatable if there exists a probabilistic, polynomial-time algorithm* RES *such that with high probability over auxiliary information* aux*, for all* $m \in \mathcal{M}$*:*

$$(\mathsf{RES}(\mathsf{aux}), \mathsf{aux}) \approx_c (\mathsf{REnc}(\mathsf{rk}_{A \xrightarrow{\mathsf{ct}_A} B}, \mathsf{ct}_A), \mathsf{aux}),$$

where $\approx_c$ *denotes computationally indistinguishable, and* ct_A *and* aux *are sampled according to* $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda), (\mathsf{pk}_A, \mathsf{sk}_A), (\mathsf{pk}_B, \mathsf{sk}_B) \leftarrow \mathsf{KG}(\mathsf{pp}), \mathsf{ct}_A \leftarrow \mathsf{Enc}(\mathsf{pk}_A, m), \mathsf{rk}_{A \xrightarrow{\mathsf{ct}_A} B} \leftarrow \mathsf{RKG}(\mathsf{sk}_A, \mathsf{pk}_B, \mathsf{ct}_A), \mathsf{aux} = (\mathsf{pp}, \mathsf{pk}_A, \mathsf{pk}_B, \mathsf{sk}_B, \mathsf{ct}_A, m)$.

Proposition 2 (Re-encryption Simulatability of cdPRE)**.** *Scheme* cdPRE *is re-encryption simulatable.*

Intuitively, the distribution of the re-encrypted ciphertext and the freshly generated one is indistinguishable.

Proof. (Outline) Let $\mathsf{ct}_A = (\mathbf{u}_A, v_A)$ be a ciphertext generated by Enc, and $\mathsf{ct}_B = (\mathbf{u}_B, v_B)$ be a ciphertext generated by REnc. For $\mathbf{u}_A$ and $\mathbf{u}_B$, we know that $\mathbf{u}_B$ is generated the same as in $\mathsf{Enc}(\mathsf{pk}_B, m)$. Therefore, $\mathbf{u}_A \approx_c \mathbf{u}_B$.

For v_B, let

$$v'_B = (\mathbf{A}\mathbf{s}_B + \mathbf{e}_B)^\top \mathbf{r}' + e_{A,2} + c_{v,A} + \lceil \frac{q}{2} \rfloor m.$$

We know that $v'_B \approx_c v_A$ because they are distributed as a normal ciphertext. We also have

$$v_B = v'_B + \mathbf{e}_A^\top \mathbf{r} - \mathbf{s}_A^\top \mathbf{e}_{A,1} - \mathbf{s}_A^\top \mathbf{c}_{u,A} + c_{rv} + c_{v,B}.$$

Those small terms are normally or evenly distributed around the zero and small enough, therefore $v_B \approx_c v_A$. □

We are ready to claim the cdHRA security of cdPRE.

Definition 4 (cdHRA Security)**.** *Let* λ *be the security parameter,* PRE *be a PRE scheme, and* $\mathcal{A}$ *be an adversary. The HRA experiment is shown in Fig. 7. A PRE scheme is said to be HRA-secure if*

$$\Pr[1 \leftarrow \mathsf{Exp}^{\mathsf{cd\text{-}hra}}_{\mathsf{PRE},\mathcal{A}}(\lambda)] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where negl *is some negligible function.*

cdHRA security is stronger than cdCPA security, since it provides an *honest re-encryption* oracle for the adversary. This new oracle provides the adversary with additional re-encrypted ciphertexts, which cannot be calculated from valid re-encryption keys. That is, if the re-encryption key from user i to user j is not allowed by $\mathcal{O}_{\mathsf{RKG}}$, the adversary can query $\mathcal{O}_{\mathsf{REnc}}$ to get some re-encrypted ciphertext except for the derivative of the challenge.

Experiment $\mathsf{Exp}^{\mathsf{cd\text{-}hra}}_{\mathsf{PRE},\mathcal{A}}(\lambda)$**:**

$\mathsf{Cor} := \emptyset, i := 0, i^* := \perp$
$\mathcal{O} := \{\mathcal{O}_{\mathsf{HKG}}, \mathcal{O}_{\mathsf{CKG}}, \mathcal{O}_{\mathsf{RKG}}, \mathcal{O}_{\mathsf{REnc}}\}$
$\mathsf{CtNum} := 0$ // a counter for ciphertexts
$\mathcal{C} := \emptyset$ // a key-value store for indices of keys and ciphertexts and the value of ciphertexts
$(i^*, m_0, m_1, st) \leftarrow \mathcal{A}_1^{\mathcal{O}}(\{\mathsf{pk}_i\}_{i \in [n]}, \{\mathsf{sk}_i\}_{i \in \mathsf{Cor}})$
$b \leftarrow \{0, 1\}$
If $i^* \in \mathsf{Cor}$: // avoid trivial attacks
  Return $\perp$
$\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}_{i^*}, m_b)$
Increment CtNum
$\mathsf{Deriv} \leftarrow \mathsf{CtNum} \cup \mathsf{Deriv}$ // record derivative of challenge
$\mathcal{C} \leftarrow \{(i^*, \mathsf{CtNum}) : \mathsf{ct}^*\} \cup \mathcal{C}$
$b' \leftarrow \mathcal{A}_2^{\mathcal{O}}(st, \mathsf{ct}^*)$
Return $b = b'$

$\underline{\mathcal{O}_{\mathsf{RKG}}(i, j, k)}$:

If ($i \notin \mathsf{Cor}$ and $j \in \mathsf{Cor}$): // avoid trivial re-encryption key generation
  Return $\perp$
If $((i, k), \cdot) \notin \mathcal{C}$: // only response recorded requests
  Return $\perp$
Return $\mathsf{RKG}(\mathsf{sk}_i, \mathsf{pk}_j, \mathsf{ct}_i)$

$\underline{\mathcal{O}_{\mathsf{Enc}}(i, m)}$:

$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}_i, m)$
Increment CtNum
$\mathcal{C} \leftarrow \{(i, \mathsf{CtNum}) : \mathsf{ct}\} \cup \mathcal{C}$
Return $(\mathsf{CtNum}, \mathsf{ct})$

$\underline{\mathcal{O}_{\mathsf{REnc}}(i, j, k)}$:

If ($j \in \mathsf{Cor}$ and $k \in \mathsf{Deriv}$): // avoid trivial re-encryption
  Return $\perp$
If $((i, k), \cdot) \notin \mathcal{C}$: // only response recorded requests
  Return $\perp$
Find ct_i by key (i, k) in $\mathcal{C}$
$\mathsf{rk}_{i \xrightarrow{\mathsf{ct}_i} j} \leftarrow \mathsf{RKG}(\mathsf{sk}_i, \mathsf{pk}_j, \mathsf{ct}_i)$
$\mathsf{ct}_j \leftarrow \mathsf{REnc}(\mathsf{rk}_{i \xrightarrow{\mathsf{ct}_i} j}, \mathsf{ct}_i)$
Increment CtNum
$\mathcal{C} \leftarrow \{(j, \mathsf{CtNum}) : \mathsf{ct}_j\} \cup \mathcal{C}$
If $k \in \mathsf{Deriv}$: // avoid trivial attacks
  $\mathsf{Deriv} \leftarrow \mathsf{CtNum} \cup \mathsf{Deriv}$
Return $(\mathsf{CtNum}, \mathsf{ct}_j)$

Fig. 7. The cdHRA security experiment, where we omitted the definition of $\mathcal{O}_{\mathsf{HKG}}$ and $\mathcal{O}_{\mathsf{CKG}}$ since they are the same as the cdCPA security experiment

Theorem 2 ([6])**.** *Let* PRE *be a CPA secure, re-encryption simulatable, proxy re-encryption scheme. Then* PRE *is HRA secure.*

Theorem 2 can be easily extended to a ciphertext-dependent version. Therefore, we can conclude that:

TABLE II
PARAMETER SETS FOR cdPRE

	Mode	n	q	k	η_1	η_2	d_u	d_v	d_{rv}	δ	Public key	Ciphertext	Re-encryption key
cdPRE512	Untuned	256	3329	2	3	2	10	4	12	$2^{-57.6}$	800B	768B	1024B
cdPRE768	Untuned	256	3329	3	2	2	10	4	12	$2^{-67.4}$	1184B	1088B	1344B
cdPRE1024	Untuned	256	3329	4	2	2	11	5	12	$2^{-81.4}$	1568B	1568B	1792B
cdPRE512	Conservative	256	3329	2	3	2	10	4	7	$2^{-57.0}$	800B	768B	864B
cdPRE768	Conservative	256	3329	3	2	2	10	4	7	$2^{-66.6}$	1184B	1088B	1184B
cdPRE1024	Conservative	256	3329	4	2	2	11	5	7	$2^{-80.6}$	1568B	1568B	1632B
cdPRE512	Optimistic	256	3329	2	3	2	10	4	4	$2^{-45.3}$	800B	768B	768B
cdPRE768	Optimistic	256	3329	3	2	2	10	4	4	$2^{-52.4}$	1184B	1088B	1088B
cdPRE1024	Optimistic	256	3329	4	2	2	11	5	5	$2^{-74.3}$	1568B	1568B	1568B

Claim 1. cdPRE *is cdHRA secure.*

VI. IMPLEMENTATION AND EVALUATION

This section tests the parameters and performance of cdPRE. To provide a practical scheme, we must ensure that the decryption error of re-encrypted messages is sufficiently small. We use Python scripts[4] to calculate the error based on the analysis of Section V-C. The parameters are listed in Table II, where the security of the three schemes cdPRE512, cdPRE768, and cdPRE1024 are approximately equivalent to that of AES128, AES192, and AES256; the Mode column stands for three different d_{rv} explained in the following subsections; n is set to 256 for encapsulate 256-bit keys; q is a small prime such that $n \mid (q-1)$ for fast Number Theoretic Transforms multiplication; k is the dimension of the ring; $\eta_1, \eta_2, d_u, d_v, d_{rv}$ are chosen to balance between security, ciphertext size, and failure rate δ; the size is in B, bytes. The other columns are explained in the following subsections.

We uploaded the Python code on GitHub[5] for a demo system implementing chain system and tree system with cdPRE. We use the cffi module to load the shared library of the Kyber encryption and our cdPRE. AES128 is used to implement SE with standard PKCS7 padding strategy, so cdPRE512 is used for re-encrypting the keys accordingly.

A. Noise Bound

The decryption error (re-encrypted ciphertext) is higher than Kyber (e.g., for Kyber512, the failure rate is $2^{-139.2}$); we claim this decryption error is acceptable. The PRE scheme is not as frequently used as a PKE scheme. More specifically, the failure rate δ of untuned cdPRE512 is $2^{-57.6}$, which takes about $1.05 \times 10^{17} \approx \ln(2)/2^{\delta}$ proxy re-encryptions to have a probability of 1/2 of at least one error. If there are one million proxy re-encryptions per day, it will take about 4.1×10^{8} years to have a probability of 1/2 of at least one error. This error increment is tolerable and acceptable.

One may be concerned that the decryption failure attack will leak the private key [24]. Note that decryption is only performed on the delegatee's side in a data-sharing scenario. In other words, we do not provide a decryption oracle in our security model. Therefore, the adversary cannot generate an arbitrary ciphertext to exploit the decryption failure.

[4]https://github.com/coro-afk/pre-cost-failure-test
[5]https://github.com/coro-afk/cdpre-from-kyber/tree/main/demo

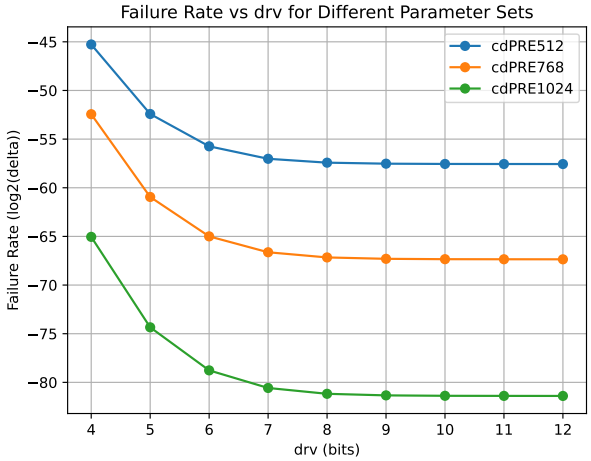

Fig. 8. Failure rate δ vs. d_{rv}

B. Communication Cost

The re-encryption key size is the largest in the first three rows (untuned mode) of Table II. As mentioned in Section V, we can compress the re-encryption key just like Kyber in algorithm Enc, and the first re-encryption key $\mathbf{u}_{A \xrightarrow{\text{ct}_A} B}$ can be cost-freely compressed. Next, we use the same strategy to adjust d_{rv} to find a satisfactory failure rate and re-encryption key length.

C. Tuning the Compression Size of Re-Encryption Key

We experiment with the relation between the compression parameter d_{ru} (it controls the size of compressed $v_{A \xrightarrow{\text{ct}_A} B}$) and the failure rate δ, the result of which is shown in Fig. 8. The size of the re-encryption key grows as the compression parameter d_{ru} grows, the result of which is shown in Fig. 9. The re-encryption key length grows linearly, and the failure rate barely decreases when $d_{rv} > 7$. Hence, we pick $d_{rv} = 4$ (for cdPRE1024, $d_{rv} = 5 = d_v$, which preserves the security inherited from Kyber) as an optimistic setting and $d_{rv} = 7$ as a conservative setting.

D. Communication Cost Comparison

We compare the Kyber-based CPA PRE scheme [22] with our cdPRE in re-encryption key and public key length, and

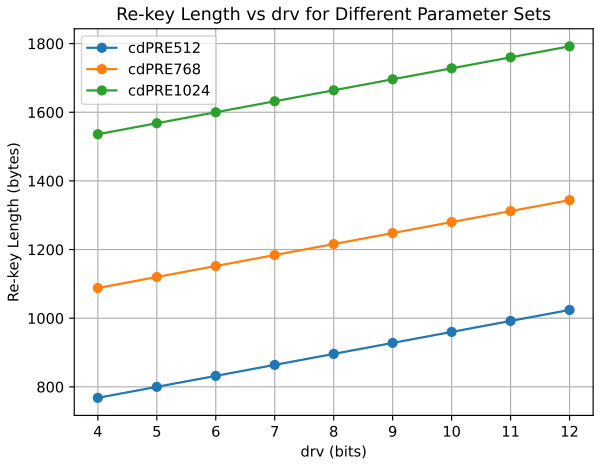

Fig. 9. Re-encryption key length vs. d_{rv}

TABLE III
RE-ENCRYPTION KEY AND PUBLIC KEY LENGTH COMPARED WITH BENCHMARK

Scheme	Item	512	768	1024
CPA PRE [22]	Re-encryption key	27648B	55296B	92160B
Untuned cdPRE	Re-encryption key	1024B	1344B	1792B
CPA PRE [22]	Public key	1568B	2336B	3104B
Untuned cdPRE	Public key	800B	1184B	1568B

the results are shown in Table III. For ciphertext, they are the same as Kyber's ciphertext.

The reason for that huge length of re-encryption key in [22] is that it uses a large re-encryption key from $R_q^{k\times k\ell} \times R_q^{k\ell}$. In the setting of Kyber512, $k = 2$ and $\ell = \lceil \log 3329 \rceil = 12$. They use two key pairs in their key generation: one for encryption and one for re-encryption key generation. That's why the public key length is nearly twice Kyber's and cdPRE's.

E. Computation Costs and Comparison

Table IV reports[6] the performance results obtained on an Intel Core i5-12500 (Alder Lake) processor clocked at 2995 MHz. The benchmarking machine has 3.7GB of RAM and is running on Ubuntu 24.04.2 LTS with Linux Kernel version 5.15.167.4-microsoft-standard-WSL2, which is a subsystem (WSL, Windows Subsystem for Linux) on Windows 10 22H2. Implementations were compiled with gcc version 13.3.0. All cycle counts reported are the median of the cycle counts of 10000 executions of the respective function. All implementation uses AVX2 vector instructions.

TABLE IV
COMPUTATION COSTS (ALDER LAKE CYCLES)

Scheme	KG	Enc	Dec	RKG	REnc
cdPRE512	8948	9572	802	11858	406
cdPRE768	16190	15790	1066	20200	550
cdPRE1024	23094	24318	1374	29972	872

[6]https://github.com/coro-afk/cdpre-from-kyber

The computation costs compared with (simulated) CPA-secure PRE in [22] are shown in Fig. 10. As explained in Section VI-D, the CPA PRE's key generation costs twice as much as cdPRE's since two key-pairs are generated. Both schemes use the same Kyber encryption and decryption algorithm with similar consumption. Re-encryption key generation and re-encryption operation multiply a polynomial matrix by a vector, rendering about ten times the computation cost than cdPRE. Overall, our cdPRE performs better than the CPA PRE.

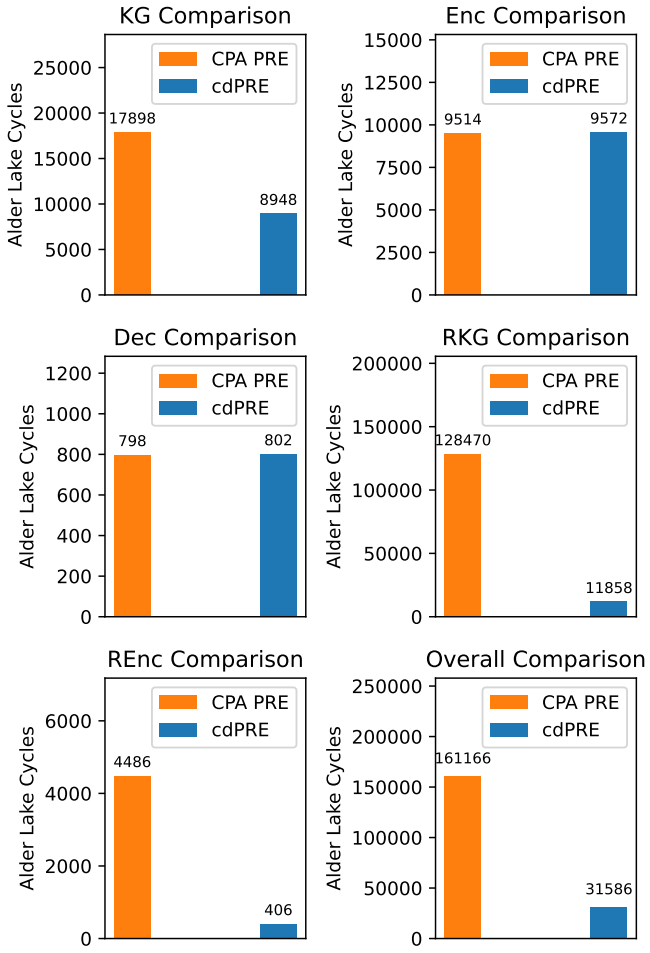

Fig. 10. Computational cost comparison: cdPRE512 vs. CPA PRE [22]

VII. FUTURE WORK

In this work, we defined a new paradigm for data sharing systems, i.e., ciphertext-dependent proxy re-encryption (cdPRE) with dynamic key generation (dKG). Although cdPRE seems limited at first glance, combined with dKG, our solution performs well in many common scenarios, e.g., pub/sub systems and general data sharing systems. Note that the key generation must be handled on the delegator side, which is different from some PRE applications; some other ciphertext sender (encryptor) may exist. For example, in an email

forwarding system, Alice delegates her decryption right to Bob, and the proxy will re-encrypt Alice's ciphertext for Bob. Suppose an email sender, say Claire, sends an email to Alice. Claire will encrypt her email with Alice's public key. Suppose we are still using a hybrid encryption paradigm. Claire picks the symmetric encryption key, but Alice has no idea about it since she stays offline. Ciphertext-dependent re-encryption becomes an obstacle to email forwarding. But anyway, in data sharing scenarios, where Alice takes control of the encryption of the ciphertext, our paradigm works perfectly. In summary, the ciphertext-dependent property of PRE keeps the control of re-encryptability in Alice's hands, and the dependent property of symmetric keys enables more efficient multi-data sharing.

Universal Composability (UC) [25] could be helpful since the data sharing system (and PRE [26]) is usually called by other protocols. We will explore the possibility of providing security in the UC framework.

REFERENCES

[1] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *International conference on the theory and applications of cryptographic techniques*. Springer, 1998, pp. 127–144.

[2] Y. Polyakov, K. Rohloff, G. Sahu, and V. Vaikuntanathan, "Fast proxy re-encryption for publish/subscribe systems," *ACM Transactions on Privacy and Security (TOPS)*, vol. 20, no. 4, pp. 1–31, 2017.

[3] C. Borcea, Y. Polyakov, K. Rohloff, G. Ryan *et al.*, "Picador: End-to-end encrypted publish–subscribe information distribution with proxy re-encryption," *Future Generation Computer Systems*, vol. 71, pp. 177–191, 2017.

[4] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Transactions on Information and System Security (TISSEC)*, vol. 9, no. 1, pp. 1–30, 2006.

[5] K. Moriarty, B. Kaliski, J. Jonsson, and A. Rusch, "Pkcs# 1: Rsa cryptography specifications version 2.2," Tech. Rep., 2016.

[6] A. Cohen, "What about bob? the inadequacy of cpa security for proxy reencryption," in *IACR International Workshop on Public Key Cryptography*. Springer, 2019, pp. 287–316.

[7] G. Fuchsbauer, C. Kamath, K. Klein, and K. Pietrzak, "Adaptively secure proxy re-encryption," in *IACR International Workshop on Public Key Cryptography*. Springer, 2019, pp. 317–346.

[8] A. Cohen, D. B. Cousins, N. Genise, E. Kline, Y. Polyakov, and S. RV, "Hra-secure homomorphic lattice-based proxy re-encryption with tight security," *Cryptology ePrint Archive*, 2024.

[9] W. Susilo, P. Dutta, D. H. Duong, and P. S. Roy, "Lattice-based hra-secure attribute-based proxy re-encryption in standard model," in *European Symposium on Research in Computer Security*. Springer, 2021, pp. 169–191.

[10] Y. Zhou, S. Liu, S. Han, and H. Zhang, "Fine-grained proxy re-encryption: Definitions and constructions from lwe," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2023, pp. 199–231.

[11] T. Perrin and M. Marlinspike, "The double ratchet algorithm," *GitHub wiki*, vol. 112, 2016.

[12] P. Rösler, D. Slamanig, and C. Striecks, "Unique-path identity based encryption with applications to strongly secure messaging," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2023, pp. 3–34.

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

[14] J. Hoffstein, J. Pipher, and J. H. Silverman, "Ntru: A ring-based public key cryptosystem," in *International algorithmic number theory symposium*. Springer, 1998, pp. 267–288.

[15] D. Stehlé and R. Steinfeld, "Making ntru as secure as worst-case problems over ideal lattices," in *Advances in Cryptology–EUROCRYPT 2011: 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings 30*. Springer, 2011, pp. 27–47.

[16] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) lwe," *SIAM Journal on computing*, vol. 43, no. 2, pp. 831–871, 2014.

[17] Y. Zhou, L. Zhao, and F. Li, "Exfiltration-resistant proxy re-encryption for iot data sharing in unreliable clouds," *IEEE Transactions on Dependable and Secure Computing*, 2025.

[18] C. Ge, W. Susilo, Z. Liu, J. Baek, X. Luo, and L. Fang, "Attribute-based proxy re-encryption with direct revocation mechanism for data sharing in clouds," in *Proceedings of the ACM Turing Award Celebration Conference-China 2023*, 2023, pp. 164–165.

[19] Y. J. Galteland and S. Wu, "Blockchain-based privacy-preserving fair data trading protocol," *Cryptology ePrint Archive*, 2021.

[20] J. Daemen and V. Rijmen, "Aes proposal: Rijndael," 1999.

[21] H. Krawczyk and P. Eronen, "Hmac-based extract-and-expand key derivation function (hkdf)," Tech. Rep., 2010.

[22] S. Sato and J. Shikata, "Bounded cca secure proxy re-encryption based on kyber," *Cryptology ePrint Archive*, 2024.

[23] C. Gentry, *A fully homomorphic encryption scheme*. Stanford university, 2009.

[24] J.-P. D'Anvers, Q. Guo, T. Johansson, A. Nilsson, F. Vercauteren, and I. Verbauwhede, "Decryption failure attacks on ind-cca secure lattice-based schemes," in *Public-Key Cryptography–PKC 2019: 22nd IACR International Conference on Practice and Theory of Public-Key Cryptography, Beijing, China, April 14-17, 2019, Proceedings, Part II 22*. Springer, 2019, pp. 565–598.

[25] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*. IEEE, 2001, pp. 136–145.

[26] N. Döttling and R. Nishimaki, "Universal proxy re-encryption," in *IACR International Conference on Public-Key Cryptography*. Springer, 2021, pp. 512–542.