HRA-Secure Homomorphic Lattice-Based Proxy Re-Encryption with Tight Security*

Aloni Cohen[1], David Bruce Cousins[2,**], Nicholas Genise[3,***], Erik Kline[4], Yuriy Polyakov[2,**], and Saraswathy RV***

[1] University of Chicago, USA
[2] Duality Technologies Inc., USA
[3] Apple, USA
[4] University of Southern California/Information Sciences Institute, USA

Abstract. We construct an efficient proxy re-encryption (PRE) scheme secure against honest re-encryption attacks (HRA-secure) with precise concrete security estimates. To get these precise concrete security estimates, we introduce the tight, fine-grained noise-flooding techniques of Li et al. (CRYPTO'22) to RLWE-based (homomorphic) PRE schemes, as well as a mixed statistical-computational security to HRA security analysis. Our solution also supports homomorphic operations on the ciphertexts. Such homomorphism allows for advanced applications, e.g., encrypted computation of network statistics across networks and unlimited hops, in the case of full homomorphism, i.e., bootstrapping.

We implement our PRE scheme in the OpenFHE software library and apply it to a problem of secure multi-hop data distribution in the context of 5G virtual network slices. We also experimentally evaluate the performance of our scheme, demonstrating that the implementation is practical.

In addition, we compare our PRE method with other lattice-based PRE schemes and approaches to achieve HRA security. These achieve HRA security, but not in a tight, practical scheme such as our work. Further, we present an attack on the PRE scheme proposed in Davidson et al.'s (ACISP'19), which was claimed to achieve HRA security without noise flooding.

Keywords: Lattice-Based Proxy Re-Encryption · Homomorphic Encryption · Distributed Networking.

---

* Distribution Statement "A" (Approved for Public Release, Distribution Unlimited). This work is supported by DARPA through HR001120C0157. The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government

** Corresponding authors: Dave Bruce Cousins (dcousins@dualitytech.com) and Yuriy Polyakov (ypolyakov@dualitytech.com)

*** This work was done while this author was at Duality Technologies.

1 Introduction

Proxy re-encryption (PRE), introduced by Blaze, Bleumer, and Strauss [5], allows re-encrypting ciphertexts encrypted under a secret key to a new encryption of the same message under a different secret key without ever having to decrypt the ciphertext. That is, PRE schemes allow for local delegation of keys. Such schemes have been studied for a wide variety of applications such as encrypted email forwarding, key escrow [14], encrypted file storage [4], secure payment system for credit cards [23], sharing patient medical records with emergency care providers [47, 6], and access control for data sharing in IoT [17, 50]. Multi-hop PRE is a chain of multiple re-encryptions from a source to a destination where a hop refers to a re-encryption. For example, multi-hop PRE solves such problems associated with distributing sensitive information payloads within and across trust boundaries while limiting distribution of encryption keys to within the boundaries of a trust zone, or to pairwise interactions between trusted agents across trust zone boundaries.

On the security side, many PRE schemes are cryptographically secure from users outside the network (without secret keys) under chosen plaintext attacks (IND-CPA), akin to many public-key encryption schemes. However, most applications necessitate security from adversaries *within the network* since otherwise all users in a network would simply share a single symmetric key. One prominent example of the need for security against internal adversaries is in 5G virtual network slices [45], where a virtual network operator's (VNO) leased hardware can leak intermediate ciphertexts via side-channel attacks. Then, an adversary can see the intermediate ciphertext, before re-encryption, as well as the re-encrypted ciphertext under their secret key. Despite sounding harmless, this simple attack can lead to secret key recovery attacks between users in the network. Cohen [14] showed IND-CPA security does not suffice in this setting and developed honest re-encryption (HRA) security for PRE to be robust against honest-but-curious users within the network. Notably, Cohen showed that all prior PRE schemes based on the (Ring)-Learning-With-Errors problem [48, 41], (R)LWE, notably [47], suffer from honest-but-curious adversaries being able to recover the ciphertext's RLWE error which then allows for learning the secret key by solving a linear system of equations.

Shortly after Cohen's work [14], Li and Micciancio [39] applied a very similar RLWE attack to an approximate FHE scheme, namely, the Cheon-Kim-Kim-Song (CKKS) scheme [13], where the adversary gets a somewhat restricted decryption oracle, called IND-CPA^D security [39, Definition 2]. The underlying implication of this connection is that the (R)LWE schemes for PRE are deeply connected to the (R)LWE schemes for (approximate) FHE. Both Cohen's fix for (R)LWE PRE schemes and the fix for CKKS require some form of noise flooding [40], but the latter introduced a fine-grained flooding technique for optimal parameters mixing both statistical and computational security whereas the former relied on crude, theoretical noise flooding bounds [3]. The same fine-grained noise flooding technique was recently used in threshold FHE as well [37]. Therefore, there is currently a significant gap in the state of the art in concrete security for approximate and threshold FHE compared to the state of the art in concrete security for lattice-based PRE schemes.

Lattice-based PRE schemes must be practical since they are the only class of PRE schemes resistant to quantum attacks. For example, lattice-based schemes were recently chosen by the National Institute of Standards (NIST) for standardized digital signatures and key-exchange mechanisms[5]. A simple quantum attack is the "harvest now, decrypt later" attack, where an adversary stores ciphertexts now and decrypts them once they have access to a quantum computer. Post-quantum, hence lattice-based, schemes are cryptographic schemes robust against these attacks.

In addition, a notable feature of RLWE schemes is that they support homomorphism and the popular fully-homomorphic encryption (FHE) schemes are based on RLWE. These include the Brakerski/Fan-Vercauteren (BFV) [7, 21] and Brakerski-Gentry-Vaikuntanathan (BGV) [8] schemes, two schemes in the simultaneous-instruction-multiple-data (SIMD) paradigm with the same plaintext spaces. FHE in the context of proxy-re-encryption enables delegating computation and key responsibilities to the cloud. FHE-based PRE schemes also enable an unlimited number of hops in the multi-hop setting since one can bootstrap a ciphertext en-route whenever the noise budget diminishes after so many hops.

1.1 Our Contributions

We introduce the tight, rigorously secure noise flooding technique recently proposed by Li et al. [40] for approximate homomorphic encryption to lattice-based PRE schemes with HRA security. This fine-grained noise flooding yields a procedure used for erasing the information about the previous secret key after re-encryption in PRE schemes. We propose an efficient, provably-secure, HRA-secure PRE scheme with precise security estimates by introducing a mixed, statistical-computational security definition and analysis. We build our system on top of the BGV FHE scheme, enabling PRE schemes with full homomorphism and unlimited re-encryptions. The same underlying ideas can be extended to BFV and CKKS FHE schemes as well.

We provide an efficient implementation of the PRE scheme using the OpenFHE library, which implements all common FHE schemes [1]. We also implement a networking application system (motivated by a use case in 5G virtual network slice security) based on the PRE functionality with Google's RPC framework [29] for multiple hops where an AES symmetric key is the data payload. We perform network simulation using the open-source RAVEN framework [34]. For example, the timing for a 13-hop parameter set in OpenFHE on an Intel® Core™ i7-9700 CPU with 64 GB RAM, a commodity desktop machine, is at most 32.68 milliseconds for re-encrypting a 3.25MB ciphertext, and decreases by 1.48 milliseconds after each hop. Our PRE scheme implementation is publicly available as part of

[5] `https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022`

the OpenFHE library [1]. Our networking application system implementation is also publicly available in a separate OpenFHE project repository [19].

In addition, we explore lattice-based alternatives to our approach for achieving HRA security. In particular, we examine the divide-and-round technique of de Castro et al. [12] used to achieve circuit privacy in homomorphic encryption schemes. We conclude this technique does not allow a more efficient multi-hop HRA-secure scheme. Furthermore, we show that the scheme presented in [16], which uses simple ciphertext re-randomization without noise flooding, is not HRA-secure despite their claims. (See Appendices A and B for more details.)

Connections to threshold and approximate FHE. Our work is closely related to the state of the art in threshold FHE [3, 37] and approximate FHE [40] since (R)LWE-based PRE, approximate FHE, and threshold FHE all compute some form of approximate decryption, or the decryption function without rounding. In PRE, this is achieved through key switching, enabled by (R)LWE's key homomorphism. Because the decryption error is not rounded away during re-encryption, as in the full RLWE decryption algorithm, the new ciphertext carries the old ciphertext's error. We construct an optimal scheme based on the state of the art in concrete security of this approximate decryption phenomenon. One could, however, round away the error at each hop, but this requires the inefficient bootstrapping procedure in FHE. (See Gentry's thesis [25] for more information on PRE using bootstrapping.) Our work shows how these three areas, RLWE-based PRE, approximate FHE, and threshold FHE are deeply connected. In short, an advancement in one of these areas yields an advance in the others.

1.2 Related work

Our work improves upon the Polyakov-Rohloff-Sahu-Vaikuntanathan (PRSV) [6, 47] system whose underlying PRE scheme does not provide HRA security. We fix this by applying the fine-grained noise flooding technique of [40] (used in the context of approximate FHE) to RLWE-based PRE schemes. This technique breaks any correlations among ciphertexts and former secret keys (as part of re-encryption) and provides a tight security reduction. The resulting scheme is multi-hop, uni-directional (re-encryption is one-way), and the initial ciphertext grows with the number of hops due to the noise flooding technique, while the re-encrypted ciphertext size drops at every hop due to modulus switching.

Attribute-based encryption (ABE) is another possible solution to building an encrypted, distributed-trust system in a network. ABE is a generalization of identity-based encryption (IBE) where the public key for encryption is created using a set of attributes defined by an access policy. The access policy determines which consumers can access data published by a producer. ABE is more appropriate for cloud systems where many users try to decrypt the same ciphertext rather than for point-to-point communication. Many ABE schemes based on bilinear pairings have been proposed in the literature [46] but are not post-quantum. Lattice-based ABE schemes are not efficient [30, 24] but offer richer access policies than PRE.

Fine-grained PRE, first constructed by Zhou et al. [52] in the single-hop CPA-secure setting and later improved to the multi-hop HRA setting [51], are PRE schemes where the message, m, gets transformed to a known function, $f(\mathsf{m})$. The constructions in [52, 51] are based on lattice trapdoors [28, 42], similar to the state-of-the-art ABE schemes. Therefore, these schemes are interesting from a theoretical point of view but suffer the same practical efficiency issues faced by lattice-based ABE schemes. Neither [52] nor [51] provide an implementation or give practical parameters[6]. Our work improves these schemes on three fronts: 1) we offer arbitrary homomorphism, 2) a tight security reduction and optimized parameters, and 3) practical implementation and simplicity of design. Practical deployments of PRE must be constant-time and making our scheme constant-time (as we sample a discrete gaussian on $\mathbb{Z}^N$) is much simpler than making discrete gaussian sampling constant time in the trapdoor-lattice regime [43] since the lattice in the latter setting is described by secret key, unlike $\mathbb{Z}^N$.

HRA security is now the standard in PRE schemes. The work in [16] presents a PRE scheme as an extension of the scheme in [47] to achieve HRA security and strong IND-post-compromise security (PCS). PCS ensures an adversary cannot distinguish a re-encrypted ciphertext from random uniform assuming the re-encryption key is known to the adversary and corruption of the producer's (sender's) secret. The re-encryption from [47] is extended with a re-randomization of the ciphertext, but it does not use an error distribution with sufficiently large standard deviation to flood traces of the previous secret key from the ciphertext, making it prone to an averaging attack. This is because the noise in the ciphertext is correlated to the sender's secret. Refer to Appendix B for an outlined HRA attack on [16] using binary matrix (R)LWE attacks [32] together with an averaging attack.

Fuchsbauer et al. [22] achieve adaptively secure PRE, where the adversary can corrupt any party throughout the security game, with a general reduction which is exponential in a parameter which depends on the adversary's corruptions, $n^{O(\log n)}$ for a binary tree of corruptions and $2^{O(n)}$ loss in general corruptions, where n is the number of parties. Asymptotically, this super-polynomial loss in security makes the scheme impractical for our use-case with many clients. As for concrete efficiency, their scheme appears to be much slower than ours because the former uses ciphertext sanitation, i.e., multiple FHE bootstrappings [20], in addition to noise flooding, to achieve this for lattice-based PRE schemes. They did not implement their scheme. Therefore, efficient, adaptively secure, post-quantum PRE with a tight security reduction is an open problem.

An even more powerful PRE scheme is universal PRE, where re-encryption is done between *any* public key scheme. Dottling and Nishimaki [18] achieve this by using either (probabilistic) indistinguishability obfuscation or garbled circuits over the PKE schemes (not practically efficient).

[6] The parameter suggestions for security parameter $\lambda = 128$, lattice dimension 128 and ciphertext modulus $\sim 2^{70}$ do not meet the lattice cryptography security estimates in the Homomorphic Encryption Standard `https://homomorphicencryption.org/standard/`, for example. Therefore, these parameter estimates are asymptotic.

Susilo et al. [49] show a lattice-based construction of attribute-based PRE. Their construction used lattice-based ABE (lattice trapdoors) and is not implemented. We expect their solution to be similar in computational and storage complexity to the state of the art in lattice-based ABE.

PRE schemes based on decisional bilinear Diffie-Hellmann (DBDH) problem were presented in [4, 11, 35]. The scheme in [4] is IND-CPA secure and provides low performance run-times for 256 and 512 bits of classical security.

1.3 Organization

PRE and other background are reviewed in Section 2. Our PRE scheme is presented in Section 3 with correctness and security analysis. Section 4 describes our network application. The experimental results are presented in Section 5, followed by concluding remarks in Section 6. Appendix A shows the necessity of noise flooding in RLWE schemes based off of the PRSV scheme. We explore alternatives to noise flooding in Appendix B. The rest of the appendices discusses the details of our implementation and networking use case.

2 Preliminaries

We use λ to denote some underlying computational security parameter. A function, f, is negligible in λ if it asymptotically satisfies $f(\lambda) = \lambda^{-\omega(1)}$. We say a probabilistic event happens with high probability if its complement happens with negligible probability. All algorithms are probabilistic polynomial time (PPT) in λ unless stated otherwise. For a PPT algorithm A with some input b, we denote its randomized output as $c \leftarrow A(b)$.

2.1 Security under Honest Re-Encryption Attacks (HRA)

The IND-CPA security definition for PRE is adapted from the IND-CPA security definition for encryption schemes. On a high level, it shows indistinguishability of re-encrypted ciphertexts when the adversary is given access to a re-encryption key generation oracle from corrupt to honest parties and corrupt to corrupt parties. (A party is corrupt if the adversary knows this party's secret key.) Cohen showed IND-CPA security is not strong enough for most applications and introduced HRA-security[14], a stronger security definition modeled against an honest-but-curious adversary corrupting parties with re-encryption keys. HRA security allows the adversary to query for re-encryption on non-challenge ciphertexts from an honest key to a corrupted key as well, in addition to the access allowed in the IND-CPA model.

Definition 1 (Proxy Re-Encryption (PRE) Scheme). *A* proxy re-encryption scheme *(PRE) for a message space $\mathcal{M}$ is a tuple of algorithms* $(\mathsf{ParamGen}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$*:*

$\mathsf{pp} \leftarrow \mathsf{ParamGen}(1^\lambda)$*: Given a security parameter* λ*, the setup algorithm outputs the public parameters* pp*.*

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(pp)$*: Given public parameters, the* KeyGen *algorithm outputs a public key* pk *and a secret key* sk[7]*.*

$\mathsf{rk}_{i\to j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$*: Given a secret key* sk_i *and a public key* pk_j*, where* $i \neq j$*, the re-encryption key generation algorithm outputs a re-encryption key* $\mathsf{rk}_{i\to j}$*.*

$\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{m})$*: Given a public key* pk_i *and a message* $\mathsf{m} \in \mathcal{M}$*, the encryption algorithm outputs a ciphertext* ct_i*.*

$\mathsf{ct}_j \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct}_i)$*: Given a re-encryption key from* i *to* j $\mathsf{rk}_{i\to j}$ *and a ciphertext* ct_i*, the re-encryption algorithm outputs a ciphertext* ct_j *or the error symbol* $\perp$*.*

$\mathsf{m} \leftarrow \mathsf{Dec}(\mathsf{sk}_j, \mathsf{ct}_j)$*: Given a secret key* sk_j *and a ciphertext* ct_j*, the (deterministic) decryption algorithm outputs a message* $\mathsf{m} \in \mathcal{M}$ *or the error symbol* $\perp$*.*

Definition 2 (PRE Correctness). *A proxy re-encryption scheme PRE is correct with respect to message space* $\mathcal{M}$*, if for all possible* $\mathsf{pp} \leftarrow \mathsf{ParamGen}(1^\lambda)$ *and* $\mathsf{m} \in \mathcal{M}$*:*

1. *with high probability over* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$*:*
$$\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, \mathsf{m})) = \mathsf{m}$$
2. *with high probability over* $(\mathsf{pk}_i, \mathsf{sk}_i), (\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$*, and* $\mathsf{rk}_{i\to j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$*:*
$$\mathsf{Dec}(\mathsf{sk}_j, \mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{Enc}(\mathsf{pk}_i, \mathsf{m}))) = \mathsf{m}$$

Note that our PRE scheme is based on RLWE and has a decryption failure rate that can be determined by the parameters chosen. Refer to Section 5 for discussion on decryption failure rate of our PRE scheme. Now we define HRA security.

Definition 3 (HRA Security Game, Definition 5 in [14]). *Fix some* λ *and let* $\mathcal{A}$ *denote some PPT adversary. The* HRA security game *consists of running* $\mathcal{A}$ *with the following oracles, in order.*
Phase 1***:***

- ◇ *Setup: The public parameters* $\mathsf{pp} \leftarrow \mathsf{ParamGen}(1^\lambda)$ *are generated and given to* $\mathcal{A}$*. A counter* $\mathsf{numKeys}$ *is initialized to* 0*, and sets* $\mathsf{Hon} \leftarrow \emptyset$ *and* $\mathsf{Cor} \leftarrow \emptyset$ *representing honest and corrupt parties, respectively, are initialized. Additionally the following are initialized:* numCt *to* 0*, sets* $\mathsf{C} \leftarrow \emptyset$ *and* $\mathsf{Deriv} \leftarrow \emptyset$*. This oracle is executed first and only once.*

[7] For ease of notation, we assume that both pk and sk include pp and refrain from including pp as an input to the other algorithms in a PRE scheme.

- ⋄ *Uncorrupted Key Generation: Sample* $(\mathsf{pk}_{\mathsf{numKeys}}, \mathsf{sk}_{\mathsf{numKeys}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ *and give* $\mathsf{pk}_{\mathsf{numKeys}}$ *to* $\mathcal{A}$*. The current value of* $\mathsf{numKeys}$ *is added to* Hon *and* $\mathsf{numKeys}$ *is incremented.*
- ⋄ *Corrupted Key Generation: Sample* $(\mathsf{pk}_{\mathsf{numKeys}}, \mathsf{sk}_{\mathsf{numKeys}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ *and give both keys to* $\mathcal{A}$*. The current value of* $\mathsf{numKeys}$ *is added to* Cor *and* $\mathsf{numKeys}$ *is incremented.*

Phase* 2*: *For each pair* $i, j \leq \mathsf{numKeys}$*, compute the re-encryption key* $\mathsf{rk}_{i\to j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$*.*

- ⋄ *Re-encryption Key Generation* $\mathcal{O}_{\mathsf{ReKeyGen}}$*: On input* (i, j) *where* $i, j \leq \mathsf{numKeys}$*, if* $i = j$ *or if* $i \in \mathsf{Hon}$ *and* $j \in \mathsf{Cor}$*, output* $\perp$*. Otherwise return* $\mathsf{rk}_{i\to j}$*.*
- ⋄ *Encryption* $\mathcal{O}_{\mathsf{Enc}}$*: On input* (i, m)*, where* $i \leq \mathsf{numKeys}$*, compute* $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{m})$ *and increment* numCt*. Store* ct *in* C *with key* (i, numCt)*. Return* $(\mathsf{numCt}, \mathsf{ct})$*.*
- ⋄ *Challenge Oracle: On input* $(i, \mathsf{m}_0, \mathsf{m}_1)$ *where* $i \in \mathsf{Hon}$ *and* $\mathsf{m}_0, \mathsf{m}_1 \in \mathcal{M}$*, sample a bit* $\mathsf{b} \leftarrow \{0, 1\}$ *uniformly at random, compute the challenge ciphertext* $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{m}_b)$*, and increment* numCt*. Add* numCt *to the set* Deriv*. Store the value* ct^* *in* C *with key* (i, numCt)*. Return* $(\mathsf{numCt}, \mathsf{ct}^*)$*. This oracle is queried once.*
- ⋄ *Re-encryption* $\mathcal{O}_{\mathsf{ReEnc}}$*: On input* (i, j, k) *where* $i, j \leq \mathsf{numKeys}$ *and* $k \leq \mathsf{numCt}$*, if* $j \in \mathsf{Cor}$ *and* $k \in \mathsf{Deriv}$ *return* $\perp$*. If there is no value in* C *with key* (i, k)*, return* $\perp$*. Otherwise, let* $\mathsf{ct_i}$ *be that value in* C*, let* $\mathsf{ct_j} \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct}_i)$*, and increment* numCt*. Store the value* ct_j *in* C *with key* (j, numCt)*. If* $k \in \mathsf{Deriv}$*, add* numCt *to the set* Deriv*. Return* $(\mathsf{numCt}, \mathsf{ct}_j)$*.*

Phase* 3*:

- ⋄ *Decision: on input bit* b'*, return* 1 *iff* $b' = b$ *and* 0 *otherwise.*

The HRA advantage of $\mathcal{A}$ *is defined as* $\mathsf{Adv}^{\mathcal{A}}(\lambda) = Pr(b' = b)$*, where the probability is over the randomness of* $\mathcal{A}$ *and the oracles in HRA game. Given a security parameter* λ*, a proxy re-encryption scheme is HRA-secure if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that* $\mathsf{Adv}^{\mathcal{A}}_{hra}(\lambda) < \frac{1}{2} + \mathsf{negl}(\lambda)$*.*

The formal definition of IND-CPA differs from HRA in $\mathcal{O}_{\mathsf{ReEnc}}$ of Phase 2 where it outputs $\perp$ if $i \in \mathsf{Hon}$ and $j \in \mathsf{Cor}$.

Now we define a re-encryption simulator from Cohen's work [14]. We use the term "statistically close" in the following definition as a place holder for arbitrary closeness metrics or divergences. One such closeness measure is KL-divergence, defined in Definition 5.

Definition 4 (Re-Encryption Simulatable, Definition 7 in [14]). *A* PRE *scheme is* re-encryption simulatable *if there exists a simulator* $\mathsf{ReEncSim}$ *such that for all* $\mathsf{m} \in \mathcal{M}$*, the distribution*

$$\{(\mathsf{ReEnc}(\mathsf{rk}_{a\to b}, \mathsf{ct}_a), \mathsf{aux})\}$$

is statistically close to

$$\{(\mathsf{ReEncSim}(\mathsf{aux}), \mathsf{aux})\}$$

where $\mathsf{aux} = (\mathsf{pp},\ \mathsf{pk}_a,\ \mathsf{pk}_b,\ \mathsf{sk}_b,\ \mathsf{ct}_a,\ \mathsf{m})$*. The strings in* aux *are honestly generated:* $\mathsf{pp} \leftarrow \mathsf{ParamGen}(1^\lambda)$, $(\mathsf{pk}_a, \mathsf{sk}_a) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, $(\mathsf{pk}_b, \mathsf{sk}_b) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, $\mathsf{rk}_{a\to b} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_a, \mathsf{pk}_b)$, $\mathsf{ct}_a \leftarrow \mathsf{Enc}(\mathsf{pk}_a, \mathsf{m})$.

Our main technique in our HRA-secure construction will be leveraging the following theorem, but we do this in a more fine-grained setting.

Theorem 1 (Theorem 5 in [14]). *Let* PRE *be a IND-CPA-secure, re-encryption simulatable PRE scheme. Then,* PRE *is HRA-secure.*

The main idea in the theorem is that $\mathsf{ct}_b \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{a\to b}, \mathsf{ct}_a)$ breaks ct_b's correlation to sk_a when the scheme is re-encryption simulatable. In (R)LWE schemes, the error in the ciphertexts can be used to recover the secret key, which is why Cohen's attack [14] on PICADOR [6] is nearly the same attack as Li and Micciancio's attack on the CKKS scheme [39]. Breaking this correlation is crucial to HRA security.

2.2 Concrete Security

Our main statistical measure for the concrete security of our PRE scheme is KL divergence.

Definition 5. *Let $\mathcal{P}, \mathcal{Q}$ be two discrete distributions with common support X. The* Kullback-Leibler Divergence *(from $\mathcal{Q}$ to $\mathcal{P}$) is defined as*
$D(\mathcal{P}||\mathcal{Q}) = \sum_{x\in X} \mathcal{P}(x) \ln \frac{\mathcal{P}(x)}{\mathcal{Q}(x)}$.

Next, we define the adversary's distinguishing advantage in state-of-the-art concrete security measures and reductions via Micciancio and Walter's work [44]. First, we define a generic distinguishing game, encompassing CPA and HRA security for PRE.

Definition 6 (Indistinguishability Game [44]). *Let $\{\mathcal{D}^0_\theta\}_\theta, \{\mathcal{D}^1_\theta\}_\theta$ be two distribution ensembles. The indistinguishability game for these ensembles between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ is as follows: $\mathcal{C}$ picks a secret bit $b \leftarrow \{0, 1\}$ at random. Then, the adversary (adaptively) sends query strings θ_i to $\mathcal{C}$ which returns a sample $\mathsf{c}_i \leftarrow \mathcal{D}^b_{\theta_i}$. Finally, the adversary returns a guess bit b' and wins if $b' = b$. An adversary is allowed to output $\perp$ as an "I do not know" symbol.*

We write $\mathcal{G}(\{\mathcal{D}^0_\theta\}_\theta, \{\mathcal{D}^1_\theta\}_\theta)$ for the security game above, and $\mathcal{G}$ when the distributions are clear from context. We define the adversary's distinguishing advantage and the scheme's resulting bit security below in Definition 7 .

Definition 7 (Advantage and Bit Security [44]). *We define an adversary $\mathcal{A}$'s output probability in game $\mathcal{G}$ as $\alpha^{\mathcal{A}} = \Pr[\mathcal{A} \neq \perp]$ and its conditional success probability as $\beta^{\mathcal{A}} = \Pr[b' = b|\mathcal{A} \neq \perp]$ where the probability is taken over the randomness in the game $\mathcal{G}$ and the adversary's internal randomness. An adversary's conditional success probability is defined as $\delta^{\mathcal{A}} = 2\beta^{\mathcal{A}} - 1$ and its advantage is* $\mathsf{Adv}^{\mathcal{A}} = \alpha^{\mathcal{A}}(\delta^{\mathcal{A}})^2$.

Cryptographic schemes or protocols often rely on a mixture of computational security (e.g., RLWE or DDH) and statistical security (noise-flooding or secret-sharing). Li et al. [40] captured this intuition in their definition of (c, s) security where c is a computational security parameter (often $128 - 256$) and s is a statistical security parameter (often 40-64 [15]). We abbreviate the time of an adversary as $T(\mathcal{A})$.

Definition 8 ((c, s) security [40]). *Let Π be a cryptographic primitive and $\mathcal{G}$ be a security game based on Π. Then, we say Π has (c, s) security for $c, s > 0$ if for any adversary $\mathcal{A}$, either $\log_2 \frac{T(\mathcal{A})}{\mathsf{Adv}^{\mathcal{A}}} \geq c$ or $\log_2 \frac{1}{\mathsf{Adv}^{\mathcal{A}}} \geq s$.*

We use (λ, ν)-security to denote the security in Definition 8 throughout the rest of the paper since we reserve λ to denote some computational security parameter and ν to denote some statistical security parameter. Now we give a lemma relating the loss in security with the number of queries an adversary has with respect to the KL divergence.

Lemma 1 (Lemma 5 in [40]). *Let $\mathcal{G}$ be an indistinguishability game (Definition 6) with distribution ensembles $\{\mathcal{X}_\theta\}_\theta$ and $\{\mathcal{Y}_\theta\}_\theta$ and $\tau > 0$. Then, for any adversary $\mathcal{A}$ making at most τ queries in game $\mathcal{G}$, $\mathsf{Adv}^{\mathcal{A}} \leq \frac{\tau}{2} \max_\theta D(\mathcal{X}_\theta || \mathcal{Y}_\theta)$.*

We use the following generalized hybrid lemma.

Lemma 2 (Lemma 2 in [44]). *Let $\{\mathcal{H}_i\}_{i=1}^{k}$ be k distribution ensembles (θ is implicit/suppressed) and let $\mathcal{G}_{i,j}$ be the indistinguisability game for $\mathcal{H}_i$ and $\mathcal{H}_j$. Let C be some (large) fixed constant, and let $\epsilon_{i,j} = \max_{\mathcal{A}} \mathsf{Adv}^{\mathcal{A}}$ with $T(\mathcal{A}) \leq C$. Then, $\epsilon_{1,k} \leq 3k \sum_{i=1}^{k-1} \epsilon_{i,i+1}$.*

2.3 RLWE Algorithms

Bold letters denote vectors. For $\mathbf{a}, \mathbf{b} \in R_Q^K$, $a[i] \in R_q$ denotes the ith entry and $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^{K} a[i] \cdot b[i]$. Let $[a]_p$ denote reducing a polynomial a's coefficients modulo p. Our noise flooding distribution is the (standard) discrete Gaussian, denoted as $D_{\sigma_{fl}}$.

Here we describe the necessary RLWE-related technical background needed to understand our PRE scheme. We use the standard RLWE setting: $R_Q = \mathbb{Z}_Q[X]/(X^N + 1)$ is a polynomial ring of dimension N where N is a power of 2 and Q is an NTT[8]-friendly modulus, $Q = 1 \bmod 2N$. Let U_Q be the uniform distribution over R_Q, χ_k denote the distribution of the secret and D_{σ_e} be the distribution of the noise. An RLWE secret key is $\mathsf{sk} = s$ for $s \leftarrow \chi_k$ and a (BGV) public key under s is $\mathsf{pk} = (\mathsf{pk}_0, \mathsf{pk}_1) = (as + pe, -a)$ where $a \leftarrow U_Q$, p is a positive integer that is the plaintext modulus such that $p \ll Q$ and is coprime to Q. Further, $e \leftarrow D_{\sigma_e}$ is the RLWE noise. The secret distribution χ_k is assumed

[8] NTT stands for the "Number Theoretic Transform". Polynomials in NTT form can be multiplied in linear time. However, Q being NTT-friendly allows us to switch representations in $O(N \log N)$ modular multiplications and additions via the NTT.

to be the distribution of polynomials in $R = \mathbb{Z}[X]/(X^N + 1)$ with coefficients in $\{0, \pm 1\}$ chosen uniformly at random. The noise distribution D_{σ_e} is a discrete Gaussian (Definition 9) of width 3.19 [2]. RLWE public key encryption for a given message $\mathsf{m} \in R_p$ and a public key pk is done by $v \leftarrow \chi_k, e, e' \leftarrow D_{\sigma_e}$, and returning $\mathsf{ct} = (v \cdot \mathsf{pk}_0 + pe', v \cdot \mathsf{pk}_1 + \mathsf{m} + pe) = (c_0, c_1)$. Decryption of ct given sk is given by $\mathsf{m}' \leftarrow [[c_0 + c_1 s]_Q]_p$, where $[\cdot]_p$ denotes reduction modulo p into the range $(-p/2, p/2]$. Note, we do not use SIMD plaintext packing for our scheme, but we can use it without changing the scheme's parameters.

BGV works with a chain of distinct NTT-friendly moduli $Q_1, \dots, Q_l, \dots, Q_L$, where $Q_l | Q_{l+1}$ for $l = 1, \dots, L-1$. The index l denotes the ciphertext level. The largest modulus is a product of NTT-friendly machine-sized primes $Q = Q_L = \prod q_i,\ i = 1, \dots, L$ where each $Q_i = \prod q_j,\ j = 1, \dots, i$. This allows us to use the Residue Number System (RNS) representation (called the "double-CRT" optimization elsewhere [26]).

Definition 9. *The discrete Gaussian (over R represented as $\mathbb{Z}^n$) with parameter $\sigma > 0$ is the probability distribution over $\mathbb{Z}^n$ given by the probability mass function* $\Pr\{\mathbf{z}\} = e^{-\|\mathbf{z}\|_2^2/2\sigma^2} / (\sum_{\mathbf{y} \in \mathbb{Z}^n} e^{-\|\mathbf{y}\|_2^2/2\sigma^2})$. *We abbreviate sampling from this distribution as $\mathbf{z} \leftarrow D_\sigma$. Note, σ is approximately the standard deviation.*

Discrete Gaussians can be efficiently sampled for relatively small σ's, and for the parameters we need for noise flooding, in constant time [43].

Modulus switching. The main noise-control method in BGV encryption is modulus switching.

Definition 10. *Let ct be a BGV ciphertext and $Q = Q'D$ be a positive integer coprime with p, and $Q \bmod p = Q' \bmod p = 1 \bmod p$. Then, the BGV* modulus-switching *operation is $\mathsf{ct}' \leftarrow (Q'/Q) \cdot (\mathsf{ct} + \delta) \in R^2_{Q'}$, where $\delta = p \cdot ([-c_0/p]_D, [-c_1/p]_D) \in R^2$.*

Brakerski et al. [8], showed that if $\mathsf{ct} = (c_0, c_1)$ was a BGV ciphertext encrypting $m \in R_p$ with $\|c_0 + c_1 s \bmod Q\|_\infty = \|m + pe\|_\infty \leq \frac{Q}{2} - \frac{pD(1+N)}{2}$, then the output ct' is a ciphertext encrypting $m/D \bmod p$ with noise $\|e'\| \leq \|e\|_\infty / D + \frac{1+N}{2}$. Halevi et al. [31] showed that noise is heuristically closer to $\|e'\| \leq \|e\|_\infty / D + \frac{1+\sqrt{N}}{2}$. We denote the algorithm in Definition 10 as: $\mathsf{ct}' \leftarrow \mathsf{ModSwitch}_Q^{Q'}(\mathsf{ct})$.

Digit decomposition. Let $k = \lceil \log_2 Q_l \rceil$ be the bit-length of the current ciphertext level. For a polynomial $a(X) = \sum_i a_i X^i \in R_{Q_l}$, we denote its binary decomposition as the vector of binary polynomials $\mathbf{a} = \sum \mathbf{a}_i X^i$ where each $\mathbf{a}_i \in \{0,1\}^k \subset R^k_{Q_l}$ is the binary decomposition of a_i: $\sum_j a_i[j] 2^j = a_i$. Let $\mathbf{2} = (1, 2, \dots, 2^{k-1})$ denote the power of two vector in $R^k_{Q_l}$, then we have $a(X) = \sum a_i X^i = \sum \langle \mathbf{a}_i, \mathbf{2} \rangle X^i = \langle \mathbf{a}, \mathbf{2} \rangle$ by linearity. Often in practice we use a larger radix base, $\omega = 2^r$, instead of 2, and the decomposition is with respect to ω. The parameter r is the digit size. If we let $\mathsf{dnum} := \lceil k/r \rceil$ be the number of digits in our decomposition and $\boldsymbol{\omega} = (1, \omega, \omega^2, \dots, \omega^{\mathsf{dnum}-1})$, then we have $a(X) = \sum a_i X^i = \sum \langle \mathbf{a}_i, \boldsymbol{\omega} \rangle X^i = \langle \mathbf{a}, \boldsymbol{\omega} \rangle$ where $\mathbf{a}_i$ is now the base-ω decomposition. We use the following notation for these decompositions in the rest of the paper:

$$\mathcal{WD}_\omega(a_i) := ([a_i]_\omega, [\lfloor a_i/\omega \rceil]_\omega, \ldots, [\lfloor a_i/\omega^{\mathsf{dnum}-1} \rceil]_\omega)$$
$$\mathcal{PW}_\omega(s) := ([s]_{Q_l}, [s\omega]_{Q_l}, \ldots, [s\omega^{\mathsf{dnum}-1}]_{Q_l}) = s \cdot \boldsymbol{\omega}$$

where s is a polynomial in R_{Q_l}. We abuse notation for a polynomial $a = \sum a_i X^i$:

$$\mathcal{WD}_\omega(a) = \sum_i \mathcal{WD}_\omega(a_i) X^i \in R_{Q_l}^{\mathsf{dnum}}.$$

Importantly, we have $\langle \mathcal{WD}_\omega(a), \mathcal{PW}_\omega(s) \rangle = a \cdot s \in R_{Q_l}$ for all polynomials $a, s \in R_{Q_l}$ and that the norm of $\mathcal{WD}_\omega(a)$ is relatively small since its coefficients are no larger than ω. This allows us to perform homomorphic inner products in RLWE-based cryptosystems while keeping the noise in control.

We use the RNS digit decomposition where we partition the current level modulus' factors into dnum' digits $\{\tilde{Q}_j\}_{j=1}^{\mathsf{dnum}'}$, $Q_L = \prod_{j=1}^{\mathsf{dnum}'} \tilde{Q}_j$, where each $\tilde{Q}_j$ is approximately the same bit-length as the others. Then,

$$\mathcal{WD}_l(a) := \left(\left[a \frac{\tilde{Q}_1}{Q_l} \right]_{\tilde{Q}_1}, \ldots, \left[a \frac{\tilde{Q}_{\mathsf{dnum}'}}{Q_l} \right]_{\tilde{Q}_{\mathsf{dnum}'}} \right), \tag{1}$$
$$\mathcal{PW}_l(s) := \left(\left[s \frac{Q_l}{\tilde{Q}_1} \right]_{Q_l}, \ldots, \left[s \frac{Q_l}{\tilde{Q}_{\mathsf{dnum}'}} \right]_{Q_l} \right). \tag{2}$$

Just as above, we have $\langle \mathcal{WD}_l(a), \mathcal{PW}_l(s) \rangle = a \cdot s \in R_{Q_l}$ for all polynomials $a, s \in R_{Q_l}$ and $\mathcal{WD}_l(a)$ has a relatively small norm. Note, we can do a base ω decomposition of $[a]_{\tilde{Q}_1}$ in Equations (1)-(2) as long as $\omega < \tilde{Q}_i$ for all i.

Key switching. The main algorithm enabling our PRE scheme is key switching. Given a ciphertext $\mathsf{ct} = (c_0, c_1)$ encrypted under a secret sk, key switching allows us to convert ct into a ciphertext $\mathsf{ct}' = (c_0', c_1')$ under a different secret sk' with the same message without knowing either secret key. It is generally used in FHE schemes since many homomorphic operations change the underlying secret key to a known function of the key.

BV key switching. The BV key-switching [9] method relies on digit decomposition to control the magnitude of the noise in ct'. The key-switching key, swk, in this case is a vector of encryptions of the secret $\mathsf{sk} = s$ multiplied by powers of the radix base ω, $\mathcal{PW}_\omega(s)$. In more detail, $\mathsf{swk} = (-\mathbf{a}s^* + p\mathbf{e} + \mathcal{PW}_\omega(s), \mathbf{a}) \in R_Q^{\mathsf{dnum}}$ is a switching key from s to s^*. Key-switching a ciphertext $\mathsf{ct} = (c_0, c_1)$ where $c_0 + c_1 s = pe + \mathsf{m}$ is given by $(\langle \mathsf{swk}_0, \mathcal{WD}_\omega(c_1) \rangle, \langle \mathsf{swk}_1, \mathcal{WD}_\omega(c_1) \rangle) + (c_0, 0)$ $= (-a's^* + c_0 + c_1 s + pe', -a') = (-a's^* + m + p(e' + e), -a')$ for $a' := \langle \mathbf{a}, \mathcal{WD}_\omega(c_1) \rangle$ and $e' := \langle \mathbf{e}, \mathcal{WD}_\omega(c_1) \rangle$. Hence, the resulting noise in BV key switching is from the inner product $\langle \mathbf{e}, \mathcal{WD}_\omega(c_1) \rangle$ modulo q where e is noise in the key-switching key. Note, key switching in RLWE schemes always results in *additive* noise growth.

Noise growth in BV key switching. Here we briefly discuss the noise growth from BV key-switching in the RNS setting. See the appendix of [36] for more details. We use the RNS version of BV key switching for our HRA-secure PRE scheme. Therefore, if the input ciphertext has noise e and the key-switching key has error coefficients at most B_{err}, then the output ciphertext has output noise at most $\|e\|_\infty + \frac{\mathsf{dnum}'\tilde{Q}B_{err}\sqrt{N}}{2}$ if we use the decomposition in Equations (1)-(2) and $\tilde{Q} = \max_i \tilde{Q}_i$. Further, the noise magnitude is at most $\|e\|_\infty + \frac{\lceil \log_\omega(\tilde{Q})\rceil \mathsf{dnum}'\omega B_{err}\sqrt{N}}{2}$ if a base-ω decomposition is done for each $\tilde{Q}_i$.

If we are not in the RNS setting, then the added noise growth from BV key switching is no more than $\frac{\lceil \log_\omega(Q)\rceil \omega B_{err}\sqrt{N}}{2}$.

Hybrid RNS key switching. We also use the hybrid key-switching technique that is commonly used in practice for improved performance in the RNS setting. This performance gain is due to the the linear growth of the number of NTTs with the number of RNS limbs in hybrid switching as compared to the quadratic growth in BV. It combines the GHS [27] technique and the original digit-decomposition-based (BV) [10] technique.

Definition 11. *Let R_{Q_l} be a power of two cyclotomic ring where $Q_l = \prod q_i$ is a product of machine-sized NTT-friendly primes, P be another NTT-friendly prime, p a BGV plaintext modulus, together with* dnum*, $\mathcal{PW}_l(\cdot)$, and $\mathcal{WD}_l(\cdot)$ defined above. For two RLWE secret keys s and s^*, a hybrid BGV key-switching key is* $\mathsf{swk} = (\mathsf{swk}_0, \mathsf{swk}_1) \in R_{PQ_l}^{2\times \mathsf{dnum}'}$ *where* $\mathsf{swk}_1 \leftarrow U_{PQ_l}^{\mathsf{dnum}'}$ *and* $\mathsf{swk}_0 = -s^*\mathsf{swk}_1 + pe + P\cdot \mathcal{PW}_l(s)$. *Then, the* key-switching *operation from s to s^* on input ciphertext* $\mathsf{ct} = (c_0, c_1)$ *encrypted under s is given by*

$$c_0^* \leftarrow c_0 + \mathsf{ModSwitch}_{PQ_l}^{Q_l}(\langle \mathcal{WD}_l(c_1), \mathsf{swk}_0\rangle), c_1^* \quad \leftarrow \mathsf{ModSwitch}_{PQ_l}^{Q_l}(\langle \mathcal{WD}_l(c_1), \mathsf{swk}_1\rangle).$$

and $\mathsf{ct}^* = (c_0^*, c_1^*)$ *is the outputted ciphertext under s^*. We denote this operation as* $\mathsf{ct}^* \leftarrow \mathsf{KeySwitch}(\mathsf{ct}, \mathsf{swk})$.

Note that the key-switching key from s to s^* can be generated with a public key for s^* and the secret key s since swk is just an encryption of $P\cdot\mathcal{PW}_l(s)$ under s^*. We use public key encryption in $\mathsf{ReKeyGen}$ for security against an adversary with access to secret s^* and the key-switching/re-encryption key from s to s^*.

Noise growth in hybrid key switching. Our implementation chooses $P \approx \tilde{Q} = \max_i \tilde{Q}_i$ which is standard. Therefore, the noise added from hybrid RNS key switching is no more than $\frac{\zeta_{num}\mathsf{dnum}'\sqrt{N}B_{err}}{2} + \zeta_{num}\frac{1+\sqrt{N}}{2}$, where ζ_{num} is the number of RNS moduli divided by the number of $\tilde{Q}_i$'s, $\zeta_{num} = \lceil (L+1)/\mathsf{dnum}'\rceil$ where $Q = \prod_{i=0}^{L} q_i = \prod_j \tilde{Q}_j$. The additive noise is at most $\frac{\zeta_{num}\mathsf{dnum}'\omega\sqrt{N}B_{err}}{2P} + \zeta_{num}\frac{1+\sqrt{N}}{2}$ if a base-ω is used in addition to the RNS decomposition in Equations (1)-(2). See the appendix of [36] for a detailed analysis.

3 Our HRA-Secure PRE Scheme

Our proposed PRE scheme is an HRA-secure extension of the scheme in [47]. We rely on the tight noise-flooding analysis of [40] for precise security estimates. This yields an efficient PRE scheme with HRA security. We show that our scheme and its tight security analysis is HRA-secure for our target application, both with a single hop and multiple hops. Our implementation supports both BV and hybrid key switching but uses hybrid for larger modulus in the RNS setting.

Although we describe and implement the scheme based on the BGV homomorphic encryption scheme [8], the same underlying ideas can be used to construct an efficient, HRA-secure PRE scheme with BFV (Brakerski, Fan, Vercauteren [7]) or CKKS (Cheon, Kim, Kim, Song [13]) encryption.

The main challenge in constructing HRA-secure RLWE-based PRE schemes is balancing the noise flooding needed to generate securely re-encrypted ciphertexts together with achieving a high level of performance. In CPA-secure, but not HRA-secure, schemes, users can fix a relatively small ciphertext modulus due to the additive noise resulting from key switching. This gives CPA-secure PRE schemes essentially the same performance as CPA-secure public-key encryption. However, these re-encrypted ciphertexts are highly correlated to the secret key under whose public key they were originally encrypted [14]. Noise flooding [3] is a well-known technique to break such correlations.

Up until recently, it was believed that one needed λ bits of noise, e.g., 2^{λ}-wide discrete Gaussian or uniformly random vector, to achieve λ bits of concrete security. This is a significant efficiency issue since any realistic λ is at least 128 to hedge against advances in cryptanalysis. Recent works changed this understanding [43, 44, 40]. The conclusion derived by Li et al. [40] is that we can flood with a significantly narrower discrete Gaussian while achieving an acceptable level of *statistical* security, nearly independent of the computational hardness of the underlying RLWE parameters. Let τ be the number of ciphertext queries allowed by the application, usually between 2^{10} and 2^{20}, t be the size of the value we are trying to flood, and ν being some statistical security parameter ($\nu \geq 40$ is often used in practice [15]). Then, a discrete Gaussian standard deviation of $\sigma = \sqrt{12\tau}2^{\nu/2}t$ is used to achieve ν-bits of *statistical* security together with λ bits of *computational* security where the latter is determined by the RLWE ring dimension and modulus [40].

Our scheme is presented in Algorithms 1–6. Recall, a PRE scheme consists of the algorithms ($\mathsf{ParamGen}$, KeyGen, Enc, Dec, $\mathsf{ReKeyGen}$, ReEnc) (Definition 1). Our $\mathsf{ParamGen}$, KeyGen, Enc, Dec algorithms are the same as in the IND-CPA secure scheme in [47], i.e., they correspond to standard BGV public key encryption, but we modify the $\mathsf{ReKeyGen}$ and ReEnc algorithms for HRA security. Our scheme achieves HRA security with tight parameters via the refined noise flooding technique of Li et al. [40]. We denote encrypting a vector of messages, $\mathbf{m} \in R^k$, or the k-repeated public-key encryption algorithm, as $\mathbf{ct} = (\mathbf{ct}_0, \mathbf{ct}_1) \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, \mathbf{m})$ where each ct_i is a fresh public key encryption of m_i ($\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, m_i)$).

Note that the ReEnc described in Algorithm 6 is key switching with a re-encryption key rk (generated using $\mathsf{ReKeyGen}$) together with a specialized re-randomization process: adding an encryption of 0 and noise flooding. This specialized re-randomization process is needed to achieve HRA security (and in more detail, re-encryption simulability [14]). In short, this re-randomization breaks the

Algorithm 1 $\mathsf{ParamGen}(1^\lambda, \nu, \mathsf{h})$

Input: computational security parameter $\lambda > 0$, a statistical security parameter $\lambda \geq \nu > 0$, and the number of hops $\mathsf{h} > 0$.
Output: pp is a multi-hop PRE parameter set with (λ, ν) HRA-security with at least h number of hops in the network.
1: **return** a (λ, ν)-HRA-secure RLWE parameter set $\mathsf{pp} = (Q_L, N, p, \chi_k, D_{\sigma_e}, D_{\sigma_{fl}})$ given in Appendix D with h hops.

Algorithm 2 $\mathsf{KeyGen}(\mathsf{pp})$

Input: pp is a multi-hop PRE parameter set.
Output: $(\mathsf{pk}, \mathsf{sk})$ is a valid public-key secret-key pair.
1: Sample $a \leftarrow U_{Q_L}$, $s \leftarrow \chi_k$, $e \leftarrow D_{\sigma_e}$.
2: Set $\mathsf{pk}_0 := as + pe$, $\mathsf{pk}_1 := -a$, $\mathsf{pk} := (\mathsf{pk}_0, \mathsf{pk}_1)$, and $\mathsf{sk} := s$.
3: **return** $(\mathsf{pk}, \mathsf{sk})$.

Algorithm 3 $\mathsf{Enc}(\mathsf{pk}, \mathsf{m})$

Input: An RLWE public key $\mathsf{pk} \in R_Q^2$, and $\mathsf{m} \in R_p$.
Output: Ciphertext ct, an encryption of m under $(\mathsf{pk}, \mathsf{sk})$.
1: Sample $v \leftarrow \chi_k, e_\beta, e_\alpha \leftarrow D_{\sigma_e}$.
2: Compute $c_0 = \mathsf{pk}_0 v + pe_\beta + \mathsf{m}$ and $c_1 = \mathsf{pk}_1 v + pe_\alpha$.
3: **return** $\mathsf{ct} = (c_0, c_1)$.

Algorithm 4 $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$

Input: RLWE secret key sk, and an RLWE ciphertext $\mathsf{ct} \in R_{Q_l}^2$.
Output: $\mathsf{m}' \in R_p$.
1: Compute $\mathsf{m}' = [[c_0 + s \cdot c_1]_{Q_l}]_p$.
2: **return** m'.

Algorithm 5 $\mathsf{ReKeyGen}(\mathsf{sk}, \mathsf{pk})$

Input: A source $\mathsf{sk} = s$ and a target pk^*.
Output: A re-encryption key $\mathsf{rk}_{s \mapsto s^*}$.
1: $\mathsf{rk}_{s \mapsto s^*} = (\boldsymbol{rk}_0, \boldsymbol{rk}_1) \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{pk}^*, \mathcal{PW}_l(s))$
2: **return** $\mathsf{rk}_{s \mapsto s^*}$

Algorithm 6 HRA-Secure $\mathsf{ReEnc}(\mathsf{ct}, \mathsf{rk}_{s \mapsto s^*})$

Input: A ciphertext $\mathsf{ct} \in R_{Q_l}^2$ encrypted under s and a re-encryption key $\mathsf{rk}_{s \mapsto s^*}$ as described in $\mathsf{ReKeyGen}$, and a public key for s^*, pk^*. Further, $D_{\sigma_{fl}}$ is a discrete Gaussian over R with width $\sigma_{fl} = \sqrt{12\tau} t 2^{\nu/2}$ where τ is the number of adversarial queries allowed by the application, t is an upper bound on the ciphertext noise, and ν is a statistical security parameter.
Output: A ciphertext ct^* encrypting the same message as ct under s^*.
1: Rerandomize: $\mathsf{ct}^{(0)} \leftarrow \mathsf{ct} + \mathsf{Enc}(\mathsf{pk}^*, 0)$.
2: Generate the flooding noise $e_{re} \leftarrow D_{\sigma_{fl}}$.
3: Flood the input $\mathsf{ct}^{(1)} \leftarrow \mathsf{ct}^{(0)} + (pe_{re}, 0)$.
4: $\mathsf{ct}^{(2)} \leftarrow \mathsf{KeySwitch}(\mathsf{ct}^{(1)}, \mathsf{rk})$.
5: Modulus switch: $\mathsf{ct}^{(3)} \leftarrow \mathsf{ModSwitch}_{Q_l}^{Q_{l-1}}(\mathsf{ct}^{(2)})$.
6: **return** $\mathsf{ct}^* = \mathsf{ct}^{(3)}$.

output ciphertext's correlation with the input ciphertext's secret key. This correlation is why the scheme from [47] does not achieve Cohen's HRA security. Further, this correlation to the secret key is nearly the same correlation observed by Li and Micciancio in their CKKS attack [39]. Analogously, we use the refined flooding technique from Li et al. [40], together with plain ciphertext re-randomization by adding an encryption of 0, as a way to break this correlation.

Correctness and noise analysis The correctness of Algorithm 6 follows immediately from the correctness of $\mathsf{KeySwitch}(\cdot,\cdot)$ and the correctness of $\mathsf{ModSwitch}_{Q_l}^{Q_{l-1}}(\cdot)$. Let e_{ks} be the additive noise from key switching, $e_{\mathsf{ks}} = \frac{\zeta_{num}\mathsf{dnum}'\omega\sqrt{N}B_{err}}{2P} + \zeta_{num}\frac{1+\sqrt{N}}{2}$ if we use a base-ω decomposition in addition to an RNS decomposition (Section 2). If the input ciphertext's noise is e, then the output of Algorithm 6 ciphertext's noise is at most $\frac{Q_{l-1}}{Q_l}(\|e\|_\infty + \|e_{re}\|_\infty + \|e_{\mathsf{ks}}\|_\infty) + \frac{1+\sqrt{N}}{2}$, where e_{re} is re-randomization noise in Algorithm 6.

3.1 The Concrete Security of Our HRA PRE Schemes

Here we give a tight reduction tracking the concrete security of our HRA-secure PRE scheme. We will use KL divergence in our proofs as a measure of statistical closeness between two distributions. We first state our main theorem relating concrete security in HRA-secure PRE schemes with the KL divergence of a re-encryption simulator.

Theorem 2. *Let Π be a λ-bit secure PRE CPA scheme. If Π has a re-encryption simulator (Definition 4) with KL divergence $\leq \rho$ between the two distributions in Definition 4, then the same scheme is $(\lambda - \log_2 24, \log_2(1/\rho) - \log_2(\tau) - \log_2 24)$ HRA secure against any semi-honest adversary with at most τ queries.*

Proof. Cohen [14] (restated in Theorem 1) first goes through re-encryption simulability, but we adopt his proof to the fine-grained setting (Definition 8). The main idea is that there they transform a CPA scheme to a stronger security notion, IND-CPA$^{\mathrm{D}}$, where the oracle added to the CPA game is simulatable. This is the same for HRA-security in PRE.

Let $\mathcal{G}_0$ be the actual HRA security game, $\mathcal{G}_1$ be the HRA security game with the simulator $\mathsf{ReEncSim}$ in place of the re-encryption oracle, and let $\mathcal{G}_2$ be the original CPA game. Similar to Theorems 2 and 5 in [40], any adversary winning in game $\mathcal{G}_2$ automatically wins in $\mathcal{G}_0$ since the oracle queries in $\mathcal{G}_2$ are a strict subset of those in $\mathcal{G}_0$.

Now we fix the following distributions: $\mathcal{H}_1 = \mathcal{G}_0^0$, $\mathcal{H}_2 = \mathcal{G}_1^0$, $\mathcal{H}_3 = \mathcal{G}_1^1$, and $\mathcal{H}_4 = \mathcal{G}_0^1$, where $\mathcal{G}_j^b$ is game $\mathcal{G}_j$ with secret bit b. Let $\epsilon_{i,j}$ be the maximum advantage of all adversaries distinguishing games $\mathcal{H}_i, \mathcal{H}_j$ (with time complexity at most 2^λ). From Lemma 2, we have $\epsilon_{1,4} \leq 12(\epsilon_{1,2}+\epsilon_{2,3}+\epsilon_{3,4})$. From Lemma 1, we have $\epsilon_{1,2}+\epsilon_{3,4} \leq \tau\rho$. (Note that these two epsilons are where we move between the actual game and the simulated query game.)

Therefore, we have $\max_{\mathcal{A}} \mathsf{adv}_{\mathcal{G}_0}^{\mathcal{A}} \leq 12(\max_{\mathcal{B}} \mathsf{adv}_{\mathcal{G}_1}^{\mathcal{B}} + \tau\rho) \leq 24\max(\max_{\mathcal{B}} \mathsf{adv}_{\mathcal{G}_1}^{\mathcal{B}}, \tau\rho)$. Now we consider both cases. If $\tau\rho \geq \max_{\mathcal{B}} \mathsf{adv}_{\mathcal{G}_1}^{\mathcal{B}}$, then $\min_{\mathcal{A}} \log_2(1/\mathsf{adv}_{\mathcal{G}_0}^{\mathcal{A}}) \geq$

$\log_2(1/\rho) - \log_2(24\tau)$. On the other hand, $\min_{\mathcal{B}}(1/\mathsf{adv}^{\mathcal{B}}_{\mathcal{G}_1}) \leq \min_{\mathcal{A}}(24/\mathsf{adv}^{\mathcal{A}}_{\mathcal{G}_0})$ if $\tau\rho < \max_{\mathcal{B}} \mathsf{adv}^{\mathcal{B}}_{\mathcal{G}_1}$ and we have $\min_{\mathcal{C}} 1/\mathsf{adv}^{\mathcal{C}}_{\mathcal{G}_2} \leq \min_{\mathcal{B}} 1/\mathsf{adv}^{\mathcal{B}}_{\mathcal{G}_1}$ since the simulator, $\mathsf{ReEncSim}$, in $\mathcal{G}_1$ is perfectly simulatable within the CPA game.

Noise Flooding According to Corollary 2 of [40][9], we must add a discrete Gaussian with standard deviation $\sigma = \sqrt{12\tau}t2^{\nu/2}$ to flood an error polynomial with absolute value at most $t > 0$, allowing for τ adversary queries, and with a statistical security parameter ν.

Lemma 3 (Lemma 6 in [40]). *For any two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^n$ with euclidean distance at most t, $\|\mathbf{x}-\mathbf{y}\|_2 \leq t$, the KL divergence between the following smudged distributions is at most ρ:*

$$D(\mathbf{x} + D_{\mathbb{Z}^n, \frac{t}{\sqrt{2\rho}}} || \mathbf{y} + D_{\mathbb{Z}^n, \frac{t}{\sqrt{2\rho}}}) \leq \rho.$$

Algorithm 7 $\mathsf{ReEncSim}$ for Algorithm 6.

Input: A ciphertext encrypted under s, $\mathsf{ct}_s \in R^2_{Q_l}$, a public key under s denoted pk_s, a public key under s^* denoted pk_{s^*}, the secret key s^*, a message m.
Output: A simulated ciphertext $\mathsf{ct}^* \in R^2_{Q_{l-1}}$ encrypting the same message as ct under s^* with a noise distribution close to the output of Algorithm 6.
1: $e \leftarrow D_{R,\sigma}$ for $\sigma = \sqrt{12\tau}2^{\nu/2}\mathsf{ct}.t$ where $\mathsf{ct}.t$ is an upperbound on the key-switching noise.
2: $\mathsf{ct}' \leftarrow \mathsf{Enc}(\mathsf{pk}_{s^*}, m)$
3: $\mathsf{ct}^* \leftarrow \mathsf{ModSwitch}^{Q_{l-1}}_{Q_l}(\mathsf{ct}') + (pe, 0)$
4: **return** ct.

Note that the real noise in the output of Algorithm 6 is $\frac{e_{\mathsf{flood}}+e_{\mathsf{KS}}}{q_{\mathsf{drop}}} + \tau'_0 + \tau'_1 s^*$ whereas the noise in the output of the simulator, Algorithm 7, is $\frac{e_{\mathsf{flood}}+e_{\mathsf{fresh}}}{q_{\mathsf{drop}}} + \tau_0 + \tau_1 s^*$ where $\tau_0, \tau_1, \tau'_0, \tau'_1$ are all identically distributed since they are the output of the rounding function applied to (unseen, re-randomized) RLWE samples.

Theorem 3. *The output of the re-encryption simulator, Algorithm 7, is within a KL divergence of $(24\tau 2^\nu)^{-1}$ from Algorithm 6. Furthermore, the re-encryption algorithm, Algorithm 6, gives a $(\lambda - \log_2 24, \nu)$-secure HRA PRE scheme if the scheme uses RLWE with λ bits of computational security.*

Proof. The KL divergence follows from plugging in $\sigma^2 = t^2/2\rho$ in Lemma 3: $\frac{1}{\rho} = \frac{2\sigma^2}{t^2} = 2(\sqrt{12\tau 2^{\nu/2}})^2 = 24\tau 2^\nu$. Furthermore, Theorem 2 boils down to plugging in $\rho = (24\tau 2^\nu)^{-1}$ into $\nu = \log_2(1/\rho) - \log_2(\tau) - \log_2(24)$ to get $(\lambda - \log_2(24), \nu)$ security.

[9] We corrected this formula to remove an unnecessary factor of $\sqrt{2}$, as we show in the proof of Theorem 3, and we removed the $\sqrt{N}$ factor since our security game is played in the coefficient domain and not the canonical embedding.

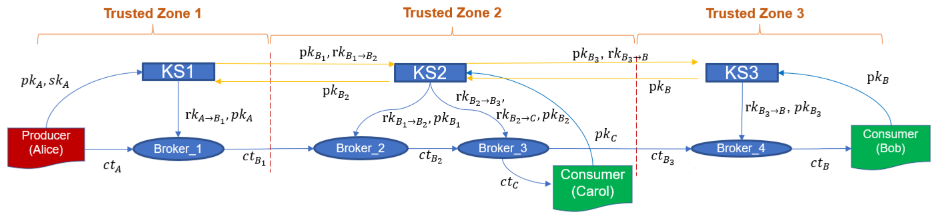

Fig. 1. Example PRE network with three trust zones, one Key Server for each zone, one producer, two consumers, and four brokers. Public, Secret, Re-encryption Keys and ciphertext exchanges are shown as connecting arrows. Secure exchange of keys between trust zones is shown in yellow.

4 Secure Multi-Hop Data Distribution System

As a motivating application for multi-hop PRE, we consider the design of a system for secure multi-hop information (AES key) distribution for 5G virtual network slices consisting of publishers and consumers with multiple trust zones. In Figure 1, we show an example network with three trust zones, four brokers, and a producer sharing content with two consumers in different trust zones. Ciphertexts are re-encrypted through a chain of brokers as they pass through multiple trust zones. Broker keys and their distribution are managed exclusively by key servers running on trusted hardware. We label these key servers as KS. They generate all keys for encryption, decryption, and re-encryption for the brokers.

In the context of 5G virtul network slices, the orchestrator in the 5G slicing architecture is trusted, so KS can be assumed to be in the same trust level as the Orchestrator for security considerations. ("key authority" is another name for an orchestrator in more general contexts.) Re-encryption keys are passed down to the brokers from the KS. Brokers are not trusted with the ability to decrypt since they can be deployed on untrusted hardware. Only consumers are trusted to decrypt. This is possible because the KS generate re-encryption keys for brokers but do not share secret keys with brokers. Note that brokers re-encrypt for the next broker down-stream, whether or not they are in the same zone. This allows trees of brokers to service a very large number of consumers. For example, a binary tree of depth d could service 2^d brokers. While multiple options exist for moving keys between entities, the approach shown in Figure 1 limits cross-zone key interactions to adjacent trusted key authorities. This keeps brokers from possessing any keys required for decryption, minimizes secure communication, and eliminates the need for a single central KS across trust zones.

Security considerations require careful design of allowed interactions among producers, brokers, consumers, and key servers. Producers and consumers generate their own keys. For example, we implement a simple whitelist of authorized consumers within the KS to limit generation of re-encryption keys to the brokers with access control. More details are provided in Appendix C. There could also

	N	$\log Q_L$	Max hops	pk	rk	ReEnc ct	ct reduction
Bounded HRA*	2048	54	+	32.63 KB	96.92 KB	32.81 KB	-
HRA-Secure	16384	438	13	4.01 MB	12.01 MB	3.25 MB	0.25 MB/hop

Table 1. Parameters and resulting key sizes for a minimum of 13 hops for both security options, plaintext modulus $p = 2$ and a digit size of 3. We use + to denote practically unlimited (over a million). ReEnc ct stands for the re-encrypted ct largest size.

	KeyGen (KS)	ReKeyGen (KS)	Enc (Producer)	First ReEnc (Broker)	Dec (Consumer)
Bounded HRA*	0.38	1.15	0.41	0.76	0.08
HRA-Secure	24.16	57.79	20.09	32.68	1.57

Table 2. Runtime performance (in milliseconds) for HRA-secure PRE schemes (average over 10 runs). Each algorithm is labeled with the network node name in parentheses (ReEnc is done by Brokers, etc.). KeyGen, ReKeyGen, Enc, Dec were benchmarked on a single-threaded commodity desktop machine while ReEnc was benchmarked on the same machine with 8 threads since ReEnc is run on a server-grade machine in the 5G secure slice use case. We expect the runtime for ReEnc to be even faster on a server-grade machine. The ReEnc runtime reduces by 1.48 milliseconds after each hop (32.68 milliseconds is the initial runtime for the first hop) due to the modulus reduction (step 5 of Algorithm 6) after each re-encryption.

be a setting where the key authorities generate keys for producers and consumers as well, depending on the application.

5 Experimental Results

We implemented our PRE scheme from Section 3 in OpenFHE by extending its BGV scheme implementation. We measured and compared the runtimes and key sizes of two variants we propose for PRE: fixed-noise (bounded-query) HRA*[10] and provably secure HRA with hybrid key switching, each described below. In short, they differ in their noise-flood/statistical security against HRA attacks. All parameters are chosen to allow a minimum of 13 hops with 128 bits of computational security. The bound $B = \alpha\sigma$ for noise from distribution D_σ determines the decryption failure rates. Since $erfc(6) \approx 2^{-55}$, the probability that the norm of a random variable (noise) sampled from D_σ is greater than B is 2^{-55}. The same probability is at most 2^{-41} while using a union bound with ring dimension up to 2^{14}. Hence, we choose $\alpha = 6$ in our implementation to target a decryption failure rate of at most 2^{-40} [27]. As mentioned in Appendix D, the number of queries and the statistical security level can be set according to the application. We fix the plaintext modulus $p = 2$ for our application of key encapsulation that transfers 256 bits AES keys from producers to consumers. Note that a hexadecimal AES key can be converted to a vector of bits to be processed by the scheme. The quality of the discrete Gaussian samples for noise flooding was verified using the GLITCH framework [33].

Bounded-Query HRA*-Secure To achieve a trade-off between performance and security, we implement this mode that adds a fixed 20-bit noise in

[10] Bounded HRA*-secure mode provides better efficiency but limited protection against re-encryption attacks

Step 3 of the ReEnc Algorithm 6 at every hop instead of full noise flooding (no modulus switching is performed). For example, the concrete security of Section 3.1 means that the (λ, ν)-HRA-security is about $(128, 20)$ if the adversary gets $2^9 = 512$ adversarial queries, minimal number of re-encryption queries and the key-switching noise is about 5.5 ($\approx \sqrt{N}$ for $N = 2048$) bits in absolute value and we start with a computational security of at least 132 bits. This mode allows for smaller parameters, allowing more hops and better performance.

HRA-Secure This mode is IND-HRA secure and implements Algorithm 6 as described with noise flooding. It supports both BV and hybrid key switching. For the concrete security of the scheme in Section 3.1, the noise flooding parameter needs to factor in the number of adversarial queries and the desired statistical security, in addition to the noise bound for key switching. The exact equation for this noise flooding distribution is a discrete Gaussian over R with width $\sigma_{fl} = \sqrt{12\tau}t2^{\nu/2}$ for $\nu \geq 48$ and 2^{18} queries.

Table 1 shows the parameters for the different security modes, along with the maximum number of hops supported for each mode, the public key size, re-encryption key size and the initial re-encrypted ciphertext size. The re-encryption key size is influenced by the digit size – the larger the digit size, the smaller the resulting re-encryption key size. However, changing the digit size also affects the number of hops and might increase the required modulus size and the ring dimension for the desired number of hops. In the case of our provable HRA-secure mode with hybrid key switching, the re-encrypted ciphertext size reduces linearly with every hop due to the modulus switching operation at every hop. That is, every hop reduces the ciphertext modulus by one machine-sized modulus. To reflect this, we show the initial re-encrypted ciphertext size and the reduction in the size at every hop.

Table 2 shows the runtime performance (as an average of 10 runs) of all PRE scheme operations using OpenFHE v1.0.3. The runtime for ReEnc for the HRA-secure mode is measured for an 8-threaded execution on an Intel® Core™ i7-9700 CPU with 64 GB RAM. We used multiple threads for ReEnc because we expect this algorithm to be run on a server-grade machine in 5G use cases. The runtimes for KeyGen, ReKeyGen, Enc, Dec are measured for a single-threaded execution on the same machine. We measure KeyGen, Enc, Dec runtimes on a less powerful instantiation since these functions are called by end users (producers and consumers), potentially on IoT devices. The KeyGen algorithm is also called by the key server to generate keys for brokers in its trust zone. Server performance can be projected using the KeyGen runtime provided in the table, and will likely be faster. Note that the key sizes, ciphertext sizes and runtimes in Tables 1 and 2 are larger for the provably-secure HRA option. This is due to the larger ring dimension and modulus Q size needed to allow for noise flooding. Since the ciphertext modulus reduces at every hop with modulus switching for the provably-secure HRA mode, the runtime for re-encryption reduces as well. Table 1 shows the initial re-encryption time, and there is an average of 1.48 milliseconds reduction in the runtime per hop.

Extensions. Conceptually, multihop PRE resembles the leveled BGV setup; here, for each hop we add an extra level. In a way, our proposal extends the (leveled) FHE model, where a new "computation" is added called re-encryption (or key switching which hides the previous key). Therefore, our solution can be easily extended to support both access delegation and homomorphic computations. For example, BGV bootstrapping could be beneficial to keep the parameters smaller if a large number of hops (say, more than 30) is required by an application.

6 Concluding remarks

We advance the state of the art in lattice-based HRA-secure PRE schemes by proposing and implementing an HRA-secure PRE scheme with tight security. Our implemented system is motivated by security issues in 5G virtual network slices, which are segmented over multiple substrate networks, resulting in multiple trust zones. Such a system can also be used for securely transferring any data payload in many other types of networks. The performance runtime, key sizes and ciphertext sizes using the implementation in OpenFHE are reported for different security modes.

Adding homomorphic computations at the broker level and further optimizing for performance will be considered for future work. Furthermore, the maliciously-secure setting is clearly of importance in the 5G virtual slice setting since there may be scenarios where the untrusted hardware acts maliciously. This interesting direction is left to future work. We believe the technical challenges here are similar to those encountered in constructing actively secure (threshold) FHE. Another interesting research direction is to find a lower bound on the number of noise-flooding bits one needs to add in order to hide all information about the secret keys used throughout the network. Our work shows that $\Omega(\nu/2)$ noise-flooding bits suffices for ν bits of statistical security for HRA security.

References

1. Al Badawi, A., Alexandru, A., Bates, J., Bergamaschi, F., Cousins, D.B., Erabelli, S., Genise, N., Halevi, S., Hunt, H., Kim, A., Lee, Y., Liu, Z., Micciancio, D., Pascoe, C., Polyakov, Y., Quah, I., R.V., S., Rohloff, K., Saylor, J., Suponitsky, D., Triplett, M., Vaikuntanathan, V., Zucca, V.: Openfhe: Open-source fully homomorphic encryption library. Cryptology ePrint Archive, Paper 2022/915 (2022), `https://eprint.iacr.org/2022/915`, `https://eprint.iacr.org/2022/915`
2. Albrecht, M.R., Chase, M., Chen, H., Ding, J., Goldwasser, S., Gorbunov, S., Halevi, S., Hoffstein, J., Laine, K., Lauter, K.E., Lokam, S., Micciancio, D., Moody, D., Morrison, T., Sahai, A., Vaikuntanathan, V.: Homomorphic encryption standard. IACR Cryptol. ePrint Arch. p. 939 (2019)
3. Asharov, G., Jain, A., López-Alt, A., Tromer, E., Vaikuntanathan, V., Wichs, D.: Multiparty computation with low communication, computation and interaction via threshold FHE. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 7237, pp. 483–501. Springer (2012)

4. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans. Inf. Syst. Secur. **9**(1), 1–30 (2006)
5. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 1403, pp. 127–144. Springer (1998)
6. Borcea, C., Gupta, A.D., Polyakov, Y., Rohloff, K., Ryan, G.W.: PICADOR: end-to-end encrypted publish-subscribe information distribution with proxy re-encryption. Future Gener. Comput. Syst. **71**, 177–191 (2017)
7. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical gapsvp. In: CRYPTO. Lecture Notes in Computer Science, vol. 7417, pp. 868–886. Springer (2012)
8. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. ACM Trans. Comput. Theory **6**(3), 13:1–13:36 (2014)
9. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: FOCS. pp. 97–106. IEEE Computer Society (2011)
10. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-lwe and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO 2011. pp. 505–524. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
11. Canetti, R., Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption. In: CCS. pp. 185–194. ACM (2007)
12. de Castro, L., Juvekar, C., Vaikuntanathan, V.: Fast vector oblivious linear evaluation from ring learning with errors. In: WAHC@CCS. pp. 29–41. WAHC@ACM (2021)
13. Cheon, J.H., Kim, A., Kim, M., Song, Y.S.: Homomorphic encryption for arithmetic of approximate numbers. In: ASIACRYPT (1). Lecture Notes in Computer Science, vol. 10624, pp. 409–437. Springer (2017)
14. Cohen, A.: What about bob? the inadequacy of CPA security for proxy reencryption. In: Public Key Cryptography (2). Lecture Notes in Computer Science, vol. 11443, pp. 287–316. Springer (2019)
15. Damgård, I., Escudero, D., Frederiksen, T.K., Keller, M., Scholl, P., Volgushev, N.: New primitives for actively-secure MPC over rings with applications to private machine learning. In: IEEE Symposium on Security and Privacy. pp. 1102–1120. IEEE (2019)
16. Davidson, A., Deo, A., Lee, E., Martin, K.: Strong post-compromise secure proxy re-encryption. In: ACISP. Lecture Notes in Computer Science, vol. 11547, pp. 58–77. Springer (2019)
17. Diro, A.A., Chilamkurti, N.K., Nam, Y.: Analysis of lightweight encryption scheme for fog-to-things communication. IEEE Access **6**, 26820–26830 (2018)
18. Döttling, N., Nishimaki, R.: Universal proxy re-encryption. In: Public Key Cryptography (1). Lecture Notes in Computer Science, vol. 12710, pp. 512–542. Springer (2021)
19. Duality Technologies: Duality Labs OpenFHE experiments for encrypted network measurement/control and secure data distribution with proxy re-encryption, `https://github.com/openfheorg/openfhe-network-examples`
20. Ducas, L., Stehlé, D.: Sanitization of FHE ciphertexts. In: EUROCRYPT (1). Lecture Notes in Computer Science, vol. 9665, pp. 294–310. Springer (2016)
21. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. IACR Cryptol. ePrint Arch. p. 144 (2012)

22. Fuchsbauer, G., Kamath, C., Klein, K., Pietrzak, K.: Adaptively secure proxy re-encryption. In: Public Key Cryptography (2). Lecture Notes in Computer Science, vol. 11443, pp. 317–346. Springer (2019)
23. Gaddam, S., Luykx, A., Sinha, R., Watson, G.J.: Reducing HSM reliance in payments through proxy re-encryption. In: USENIX Security Symposium. pp. 4061–4078. USENIX Association (2021)
24. Genise, N., Micciancio, D., Polyakov, Y.: Building an efficient lattice gadget toolkit: Subgaussian sampling and more. In: EUROCRYPT (2). Lecture Notes in Computer Science, vol. 11477, pp. 655–684. Springer (2019)
25. Gentry, C.: A fully homomorphic encryption scheme. Ph.D. thesis, Stanford University, USA (2009)
26. Gentry, C., Halevi, S., Smart, N.P.: Fully homomorphic encryption with polylog overhead. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 7237, pp. 465–482. Springer (2012)
27. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: CRYPTO. Lecture Notes in Computer Science, vol. 7417, pp. 850–867. Springer (2012)
28. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC. pp. 197–206. ACM (2008)
29. Google.com: grpc: Google remote procedure call framework, `github.com/grpc/grpc/`
30. Gür, K.D., Polyakov, Y., Rohloff, K., Ryan, G.W., Sajjadpour, H., Savas, E.: Practical applications of improved gaussian sampling for trapdoor lattices. IEEE Trans. Computers **68**(4), 570–584 (2019)
31. Halevi, S., Polyakov, Y., Shoup, V.: An improved rns variant of the bfv homomorphic encryption scheme. In: Matsui, M. (ed.) CT-RSA 2019. pp. 83–105. Springer International Publishing, Cham (2019)
32. Herold, G., May, A.: LP solutions of vectorial integer subset sums - cryptanalysis of galbraith's binary matrix LWE. In: Public Key Cryptography (1). Lecture Notes in Computer Science, vol. 10174, pp. 3–15. Springer (2017)
33. Howe, J., O'Neill, M.: Glitch: A discrete gaussian testing suite for lattice-based cryptography. Cryptology ePrint Archive, Report 2017/438 (2017), `ia.cr/2017/438`
34. Institute, U.I.S.: Raven: Ry's apparatus for virtual encodable networks, `gitlab.com/mergetb/tech/raven`
35. Ivan, A., Dodis, Y.: Proxy cryptography revisited. In: NDSS. The Internet Society (2003)
36. Kim, A., Polyakov, Y., Zucca, V.: Revisiting homomorphic encryption schemes for finite fields. In: Advances in Cryptology – ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part III. p. 608–639. Springer-Verlag, Berlin, Heidelberg (2021). https://doi.org/10.1007/978-3-030-92078-4_21
37. Kluczniak, K., Santato, G.: On circuit private, multikey and threshold approximate homomorphic encryption. IACR Cryptol. ePrint Arch. p. 301 (2023)
38. Lee, J.: Lecture notes in sublinear algorithms for big data (2020), `cs.brown.edu/courses/csci1951-w/lec/lec%203%20notes.pdf`
39. Li, B., Micciancio, D.: On the security of homomorphic encryption on approximate numbers. In: EUROCRYPT (1). Lecture Notes in Computer Science, vol. 12696, pp. 648–677. Springer (2021)

40. Li, B., Micciancio, D., Schultz, M., Sorrell, J.: Securing approximate homomorphic encryption using differential privacy. In: CRYPTO (1). Lecture Notes in Computer Science, vol. 13507, pp. 560–589. Springer (2022)
41. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 6110, pp. 1–23. Springer (2010)
42. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 7237, pp. 700–718. Springer (2012)
43. Micciancio, D., Walter, M.: Gaussian sampling over the integers: Efficient, generic, constant-time. In: CRYPTO (2). Lecture Notes in Computer Science, vol. 10402, pp. 455–485. Springer (2017)
44. Micciancio, D., Walter, M.: On the bit security of cryptographic primitives. In: EUROCRYPT (1). Lecture Notes in Computer Science, vol. 10820, pp. 3–28. Springer (2018)
45. Olimid, R.F., Nencioni, G.: 5g network slicing: A security overview. IEEE Access **8**, 99999–100009 (2020)
46. de la Piedra, A., Venema, M., Alpár, G.: ABE squared: Accurately benchmarking efficiency of attribute-based encryption. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2022**(2), 192–239 (2022)
47. Polyakov, Y., Rohloff, K., Sahu, G., Vaikuntanathan, V.: Fast proxy re-encryption for publish/subscribe systems. ACM Trans. Priv. Secur. **20**(4), 14:1–14:31 (2017)
48. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC. pp. 84–93. ACM (2005)
49. Susilo, W., Dutta, P., Duong, D.H., Roy, P.S.: Lattice-based hra-secure attribute-based proxy re-encryption in standard model. In: ESORICS (2). Lecture Notes in Computer Science, vol. 12973, pp. 169–191. Springer (2021)
50. Zach, H., Peinsold, P., Winter, J., Danner, P., Hatzl, J.: Using proxy re-encryption for secure data management in an ambient assisted living application. In: Hühnlein, D., Roßnagel, H., Kuhlisch, R., Ziesing, J. (eds.) Open Identity Summit 2015. GI (2015)
51. Zhou, Y., Liu, S., Han, S.: Multi-hop fine-grained proxy re-encryption. IACR Cryptol. ePrint Arch. p. 55 (2024)
52. Zhou, Y., Liu, S., Han, S., Zhang, H.: Fine-grained proxy re-encryption: Definitions and constructions from LWE. In: ASIACRYPT (6). Lecture Notes in Computer Science, vol. 14443, pp. 199–231. Springer (2023)

Appendices

A Why Simple Rerandomization is Not HRA Secure

Here we sketch a simple HRA attack showing that simple rerandomization, or just adding fresh encryptions of 0 locally after key-switching, is not an HRA-secure PRE scheme. This method was used in the work of Davidson et al. [16] where they claimed this method to satisfy HRA security assuming RLWE. Below, we show an attack with (R)LWE with a binary matrix (not binary error or binary secret) and a simple averaging argument. The former was shown to be insecure by Herold and May [32]. We leave an in-depth, experimental cryptanalysis to future work since our goal is simply to demonstrate a lack of security in the rerandomization approach without noise flooding. Note that the attack we sketch may not be the most effective as there may be more efficient breaks on this scheme in the HRA security model.

PRE without noise flooding. Recall, an HRA adversary gets access to a honest-to-corrupt re-encryption oracle without ever seeing the associated re-encryption key. Therefore, the adversary is going to query this oracle, decrypt, and use the RLWE error to learn information about the honest secret key. The main point is that the RLWE error is highly correlated to the honest secret key.

Here we review the algorithms in [16]. User A's public-secret key pair is generated as a standard RLWE sample $a \leftarrow R_q,\ e_{\mathsf{pk}} \leftarrow D_{R,\sigma_f}, s_A \leftarrow \{0, \pm 1\}^n, b \leftarrow as_A + pe_{\mathsf{pk}}$ and $\mathsf{pk}_A := (b, a)$ and $\mathsf{sk}_A = s_A \in R$ where σ_f is small (often 3.2 in applications). Furthermore, public key encryption of $m \in R_p$ is given by $v, e', e'' \leftarrow D_{R,\sigma_f}$ and the output is $\mathsf{ct} = (vb + pe' + m, va + pe'')$. Note that the fresh encryption error is given by $e_{\mathsf{fresh}} = e_{\mathsf{pk}} + e' - s_A e''$. The re-encryption key $\mathsf{rk}_{s \mapsto s^*}$ is generated by $\mathsf{rk}_i = \mathsf{Enc}(\mathsf{pk}_{s^*}, -s2^i)$. Therefore, the noise in the re-encryption key is simply the fresh noise as an i.i.d. vector.

Similar to ours, the scheme in [16] uses a digit decomposition step in re-encryption (Algorithm 8 below). For simplicity, we give the attack where the digit decomposition is given in binary digits, or $r = 1$, in Figure 8 in [16].

Algorithm 8 Re-Encryption Without Noise Flooding [16].

Input: A ciphertext $c = (c_\alpha, c_\beta) \in R_q^2$ encrypted under s, a re-encryption key $\mathsf{rk}_{s \mapsto s^*} = (\mathbf{rk}_\beta, \mathbf{rk}_\alpha)$ as described in $\mathsf{ReKeyGen}$, and a pk public key to s.
Output: A ciphertext c^* encrypting the same message as c under s^*.
1: Decompose $\tilde{c}_\alpha = \langle \mathbf{c}_\alpha, \mathbf{2} \rangle$.
2: Compute $c^*_\beta = c_\beta + \langle \mathbf{c}_\alpha, \mathsf{rk}_\beta \rangle$, and $c^*_\alpha = \langle \mathbf{c}_\alpha, \mathsf{rk}_\alpha \rangle$.
3: $c^* = (c^*_\beta, c^*_\alpha)$, $c' \leftarrow \mathsf{Enc}(\mathsf{pk}_{s^*}, 0)$.
4: **return** $c^* + c'$

Attack. The main idea of our attack is to simply re-encrypt the many ciphertexts from the same honest party. The error will be structured in a way which enables the receiver to recover the original ciphertext's error. In turn, we can recover the secret key just as Li and Micciancio [39] and Cohen [14] attack approximate FHE and previous RLWE PRE schemes, respectively.

Re-encryption without noise-flooding is given in Algorithm 8. Let the vector $\mathbf{x}$ denote the ciphertext error in the re-encryption key $\mathsf{rk}_{s \mapsto s^*}$, $\mathbf{x} = e_{\mathsf{pk}_{s^*}} \mathbf{v} + \mathbf{e}'' - s^* \mathbf{e}' \in R_q^{\log_2 q}$. If the input ciphertext is $\mathsf{ct}_{\mathsf{in}} = (c_0, c_1) = (as + pe_{\mathsf{ct}} + m, a)$ and the re-encryption key is $\mathsf{rk}_{s \mapsto s^*}$, then the output is a randomized ciphertext with noise $\langle \mathbf{x}, \tilde{\mathbf{c}}_1 \rangle + e_{\mathsf{fresh}, s^*} + e_{\mathsf{ct}}$ where $\tilde{\mathbf{c}}_1$ is a binary vector (vector of binary polynomials) representing the bit-decomposition of the input ciphertext's second ring element c_1. Note, that $\tilde{\mathbf{c}}_1$ is known to adversary since it is the bit decomposition of the input.

Then, we can randomize the ciphertext by calling a new encryption from the same party. Repeating this (many) times gives us the binary RLWE problem: $(\mathbf{C}\mathbf{x}+\mathbf{e}, \mathbf{C})$ where $\mathbf{e}$ is the vector $\mathbf{e} = e_{\mathsf{ct}}\mathbf{1} + \mathbf{e}_{\mathsf{fresh},s^*}$ with e_{ct} as the original, fixed, ciphertext error, and $\mathbf{C}$ is a public binary matrix. Once we get the vector $\mathbf{x}$, we can subtract the inner-product $\langle \mathbf{e}, \mathbf{x} \rangle$. This now reduces to an averaging argument, by re-encrypting the same ciphertext repeatedly, to recover e_{ct} and therefore recover the original secret s. Lastly, we note that the magnitude of e_{ct} and the entries of$\mathbf{e}_{\mathsf{fresh},s^*}$ are all of similar magnitude since e_{ct} is a fixed error resulting from a fresh encryption under s. Generic meaning finding algorithms require a quadratic number of samples in order find a mean[11].

We note that changing Algorithm 8 to rerandomize before digit decomposition is still vulnerable to averaging attacks since the noise there is changed to $\langle \mathbf{e}_{rk}, \mathbf{b} \rangle + e_{s,f} + e_{\mathsf{ct}}$ where is a fixed vector representing the noise in the re-encryption key, $\mathbf{b}$ is a uniformly random binary vector, $e_{s,f}$ is a fresh encryption noise from s's public key, and e_{ct} is the fixed ciphertext noise we are trying to recover.

B Circuit privacy technique of [12]

De Castro et al. [12] describe using a simple modulus reduction technique with the BFV scheme for circuit privacy. Let $\mathsf{round}_{Q\to Q_0}(y) = \lfloor \frac{y}{Q_0^*} \rceil_{Q_0}$ for $q = q_0 q_0^*$. This is the modulus switching operation from Q to Q_0. The high-level idea in [12] is that the error-less portion of a BFV encryption, $as + \Delta m$, is uniformly random over R_Q where $Q = Q_0 Q_0^*$ and $\Delta = \lfloor Q/p \rfloor$. Then, the function round hides the circuit-dependent error, e in $as + \Delta m + e$, as long as $\mathsf{round}(as + \Delta m + e) = \mathsf{round}(as + \Delta m)$. For simplicity, assume $m = 0$. Then, this is the same event that as is not within a distance of $\|e\|_\infty$ of a multiple of q_0^*. The technique hides the circuit dependent error with high probability for the right choice of parameters as in Theorem 3.8 of [12]. We adapt the same technique to re-randomize the ciphertext being re-encrypted. Since there is already a modulus switching operation in the re-encryption algorithm, this allows to achieve HRA security without the additional overhead of noise flooding.

Since we present our instantiation of the PRE scheme in Section 3 with BGV scheme, we first show how this adapts to BGV and that it can be applied for re-randomization in re-encryption. The re-encryption algorithm with this technique is defined in Algorithm 9. Note that the procedure to obtain $ct^{(1)}$ from a ciphertext ct being re-encrypted is exactly the same as Algorithm 1 of [12]. For BGV, we see that multiplying by $p^{-1} \bmod Q$ permutes $\mathbb{Z}_Q$. Then, the probability that $\mathsf{round}(as+e) = \mathsf{round}(as)$ is the same probability as $\mathsf{round}(p[as+e] \bmod Q) = \mathsf{round}(p \cdot as \bmod Q)$ since multiplication by $p \bmod Q$ is invertible when $(p, Q) = 1$. Lastly, we note that as is uniformly random if and only if $p \cdot as$

[11] This can be done with the Central Limit Theorem or concentration bounds, like Bernstein and Hoeffding concentration inequalities [38]. See Lecture 3 of `https://cs.brown.edu/courses/csci1951-w/` for details on generic mean-finding algorithms.

is uniformly random. So, replace a by $a' = p^{-1}a$ and we see

$$\begin{aligned} &Pr\{\mathsf{round}(a's) = \mathsf{round}(a's + e)\} \\ &= Pr\{\mathsf{round}(p \cdot a's) = \mathsf{round}(p \cdot [a's + e])\} \\ &= Pr\{\mathsf{round}(as) = \mathsf{round}(as + pe)\}. \end{aligned}$$

Algorithm 9 HRA-Secure Re-Encryption with divide and round technique from [12]

Input: A ciphertext $\mathsf{ct} \in R_Q^2$ encrypted under s and a re-encryption key where $Q = Q_0 Q_0^*$ $\mathsf{rk}_{s \mapsto s^*}$ as described in $\mathsf{ReKeyGen}$, and a public key for s^*, pk^*.
Output: A ciphertext ct^* encrypting the same message as ct under s^*.
1: Rerandomize: $\mathsf{ct}^{(0)} \leftarrow \mathsf{ct} + \mathsf{Enc}(\mathsf{pk}^*, 0)$.
2: Divide and round: $\mathsf{ct}^{(1)} \leftarrow \mathsf{ModSwitch}_Q^{Q_0}(\mathsf{ct}^{(0)})$.
3: $\mathsf{ct}^{(2)} \leftarrow \mathsf{KeySwitch}(\mathsf{ct}^{(1)}, \mathsf{rk})$.
4: **return** $\mathsf{ct}^* = \mathsf{ct}^{(2)}$.

For decryption correctness and security, appropriate values are chosen for the moduli Q_0 and Q_0^* respectively.

Security The security of the scheme can be shown using similar arguments from Section 3 but with reduced number of queries. This is because Lemma 3.6 of [12] uses statistical distance as a measure to show indistinguishability of the real distribution and the noise free distribution. Using Pinsker inequality to adapt this to KL-Divergence results in a quadratic factor increase of the statistical security parameter s. As a consequence, it results in reduced number of adversarial queries for a given security level compared to the noise flooding approach. In addition to a reduced number of queries, this approach is not favorable for multiple hops. Suppose $Q = Q_0 \dots Q_L$, then for every hop i, we need to choose Q_i such that $\frac{2n}{Q_0}\|e^{(0)}\| < 2^{-s}$ from Theorem 3.8 of [12] where $e^{(0)}$ is the noise in ciphertext $ct^{(0)}$. In the context of our PRE, the noise $e^{(0)}$ is the encryption noise if ct is a fresh encryption or accumulated noise from prior evaluations. For multihop, since we need every RNS moduli satisfy this condition for a given statistical security s and for the larger modulus Q to fit the parameters with respect to RLWE hardness, it either results in very few number of hops or low statistical security s.

Correctness The choice of Q_0 is such that $Q_0 > 2p(\|e_{ms}| + \|e_{ks}\|)$ for decryption correctness.

C Simulated Secure Data Distribution System using the OpenFHE PRE functionality

Using the PRE functionality in OpenFHE, we built a multihop example system that allows multiple trust zones (with a key server for each trust zone) to transfer

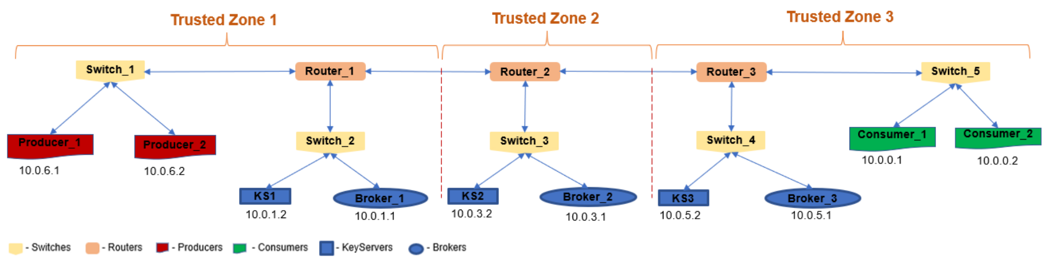

Fig. 2. Network topology used in RAVEN for PRE with three trust zones. Each service - key server, producer, broker and consumer are run on virtual machines. Each trust zone is separated by a router.

256-bit AES keys from multiple producers to multiple authorized consumers using gRPC's [29] authenticated remote procedure calls. We set the number of trust zones to 3. (See Figure 1 for an example with three trust zones, where the arrows represent authenticated gRPC transactions for the messages exchanged.) The example implementation further supports secure communication with TLS (Transport Layer Security) authentication using gRPC's SSL/TLS API with a dummy certificate setup. We used a simple user-name based access control for the producer's content. In general, the information flow is from producers to consumers via (potentially multiple) brokers.

We performed the network simulation using the open-source RAVEN framework [34]. Each service was run in a single thread on a virtual machine created by the RAVEN framework running on a host machine. Our simulation code is flexible: the virtual machines can be configured according to the application and the resources of the host machine. The example RAVEN topology we built is shown in Figure 2. It has 3 trust zones with one key server and one broker for each trust zone. Note that this figure shows our network setup for exactly the information flow of Figure 1, the only difference being that Figure 1 has an additional broker in trust zone 2 to show that consumers can be present in either of the trust zones. The number of trust zones, key servers, brokers, producers and consumers can be adjusted by defining the topology and corresponding network configuration in RAVEN and hence can be adjusted as needed.

We configured routers and switches to simulate a real-world network: each trust zone is connected through a router and each router has a switch that multiple services in the same trust zone can connect to, i.e., they are in the same subnet mask. Services make function calls to the OpenFHE PRE functionality to distribute keys.

We used an AMD EPYC 7302 16-Core Processor machine with 500 GB memory as the host machine. Each service is run on virtual machines that are set up to be nodes with 2 cores and 4 GB memory. This is possible because we built the code for the example system and the OpenFHE code with the PRE functionality using a builder node with a 16-core CPU and 64 GB memory.

The docker containers created by the builder node are then used in the actual execution to run the system. We now describe each service in more detail.

Key Server (KS). This service is responsible for generating key pairs for brokers and for generating re-encryption keys for the following flows: from the producer to a downstream broker, from an upstream broker to a downstream broker, and from a downstream broker to a consumer. The key server uses a whitelist of consumers authorized for access control. This is implemented in gRPC as an asynchronous server that handles requests from producers, consumers, and brokers. There is one key server for each trust zone. PRE function calls made by the KS service: KeyGen, ReKeyGen.

Producer. The producer is implemented as a gRPC client that sends its ciphertext to its downstream broker and its key pair to the key server. The downstream broker re-encrypts the ciphertext received to its own key and caches it locally to respond to downstream requests. PRE function calls made by the service: Encrypt.

Broker. This service is responsible for processing the ciphertext (in our example application, an encapsulated AES key) sent from the producer. Each broker acts as a server to its connected downstream brokers by sending them re-encrypted ciphertexts. The broker also acts as a client to its upstream broker by requesting ciphertexts from the upstream broker. This is also implemented as an asynchronous server in gRPC. Note that since each broker can service multiple downstream brokers, we can configure a large cascade tree of brokers to distribute data to a large number of consumers with only a few hops. The PRE function calls made by the service: ReEncrypt.

Consumer. The consumer is implemented as a gRPC client. The first time a consumer requests a given producer's ciphertext from its upstream broker, that broker sends a request for the re-encrypted ciphertext to its upstream broker recursively until it reaches the broker connected to the producer. This is implemented using routing tables to cache the route from a consumer to a producer (known as a channel). Currently only one ciphertext per channel is supported. The brokers cache local re-encrypted copies of the channel's ciphertext, so that if a different consumer requests the same source data, the broker can use its locally cached ciphertext. Note that since a consumer is going to decrypt the ciphertext (rather then re-encrypt), the broker returns a re-encrypted ciphertext specific to the consumer's secret key. The PRE function calls made by the service: Decrypt.

D Setting the parameters

We implemented the scheme presented in Section 3 with three different security modes - Bounded-Query HRA*-secure, HRA-secure (Refer to Section 5), and CPA-Secure. The three modes use the same $\mathsf{ReKeyGen}$ algorithm, Algorithm 5, to generate re-encryption keys and only differ in their ReEnc algorithms. The CPA-secure is the PRE scheme without noise flooding in Step 3 and modulus switching in Step 5 of Algorithm 6. It can be used for applications that do not require HRA security. The scheme is similar to the IND-CPA scheme in [47], but adapted to the public-key setting.

Logic for setting the Parameters The scheme allows for multiple hops, but the choice of optimal parameters depends on many factors: security level required and security mode (CPA, bounded HRA*, HRA-secure), encrypted payload size (in bits), number of broker hops required (number of re-encryptions), and other efficiency considerations such as latency, throughput, computation time and ciphertext/key size. We use the homomorphic encryption standard [2] for a given computational security level (128, 192 or 256 bits of security) to select parameters such as the modulus bit-length $\log_2 Q_L$ and the ring dimension N. For a given Q_L and N, we are estimating the number of hops possible based on the decryption correctness condition of the corresponding security mode. We may need to adjust (increase) these parameters to achieve a desired number of hops. The overall efficiency of the protocol also depends on the choice of the plaintext modulus p and the decomposition digit size used in key switching. For a non-RNS modulus Q less than 60 bits, the digit size r in BV switching is such that the digit decomposition is done with base $\omega = 2^r$ (Refer to Section 2.3). The digit size in BV switching in the RNS setting is the size of each RNS moduli Q_i while the digit size in hybrid switching in the RNS setting is $\lceil \log_2 Q \rceil / d_{num}$, where we use $d_{num} = 3$. The best performance (latency) for re-encryption is usually achieved when $p = 2$ and digit size is 3 or 4 (as observed in [47]). So we start with r such that the digit size is 3 while choosing the parameters. These values may be modified if the resulting number of hops is insufficient for our scenario or if it results in a larger ring dimension, as a trade-off.

In addition to re-encryption, homomorphic computation on ciphertexts is possible as well. However, if brokers need to perform multiplication on an encrypted value, then one needs to increase the multiplicative depth by increasing the modulus, Q_L. This will, in turn, increase the resulting ciphertext and public/re-encryption key size. Since our initial scenario is to use PRE for key distribution and secure access control, we have decided to select parameters assuming no computation is performed on the re-encrypted ciphertexts. We wrote a python script for determining multi-hop cryptographic parameters based on these criteria. The pseudocode for the program is given below. For an input computational security parameter λ, a payload bit length, and a minimum number of hops $\mathsf{h} > 0$, we generate a BGV parameter set $(N, Q_L, p, \chi_k, D_{\sigma_e}, D_{\sigma_{fl}})$ as follows:

- Pick a security level from HE standards which is the at least λ $(128, 192,$ or 256 bits).
- Compute a minimum ring size = (payloadbits/$\log_2(p)$). Verify that the minimum ring size is within the allowable range for the standard (i.e., < 32768), note that to allow for multiple hops with noise flooding, the minimum ring size required in 4096. If this is ≥ 32768, increase p if possible. (Otherwise, the application will use multiple ciphertexts per message vector.)
- While ring size <= 32768:
  - Determine the maximum $\log Q_L$ from λ and N using the tables in [2].

Payload (bits)	N	$\log Q_L$	p	Digits	ReEnc ct Size	ct expansion
IND-CPA security						
1024	1024	27	2	1	16.8 KB	134.34
2048			4			67.17
4096			16			33.59
8192			256			16.8
16384	2048	54	65536	18	32.8 KB	16.4
Bounded (Fixed 20 bits noise) HRA* security						
1024	2048	54	2	18	32.8 KB	262.5
2048			4			131.2
4096			16			65.6
8192			256			32.8
16384			65536			16.4
Provably-secure HRA security						
1024	4096	109	2	30	65.0 KB	520.0
2048			4			260.0
4096			16			130.0
8192			256			65.0
16384			65536			32.5

Table 3. Ciphertext expansion (the ratio of plaintext size vs re-encrypted ciphertext size) for the three security options, single-hop mode.

- Verify that $\log Q_L$ satisfies the noise flooding condition for min h hops, ring size, p. Stop if satisfied, otherwise increase ring size by factor of two and try again.

We also measured the ciphertext expansion for different payload bit sizes for a single-hop PRE as shown in Table 3 for the three security options. The parameters are chosen to allow for decryption correctness with single hop for each payload bits size. The digit size does not impact the ciphertext expansion. For IND-CPA, larger plaintext modulus does not allow for a hop when the digit size is larger than 1 for ring dimension $N = 1024$. So we use digit size 1 for comparison with different values of p until the plaintext modulus is large enough to raise the ring dimension to 2048 for a single hop.

Our full performance numbers are given in Section 5. The runtime of re-encryption is small (under 50 ms), which makes this solution practical for secure distribution of AES keys. The end-to-end performance of such an application system has two components - the network overhead that is influenced by the message sizes (Table 1) and network speed, and the performance of the PRE function calls (Table 2).