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 definition 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., when bootstrapping is available.

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.

Moreover, we compare our PRE method with other lattice-based PRE schemes and approaches targeting 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, i.e., without adding large noise.

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.

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

*** Corresponding author: Yuriy Polyakov (ypolyakov@dualitytech.com)

1 Introduction

Proxy re-encryption (PRE), introduced by Blaze, Bleumer, and Strauss [BBS98], 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 [Coh19], encrypted file storage [AFGH06], secure payment system for credit cards [GLSW21], sharing patient medical records with emergency care providers [PRSV17, BGP+17], and access control for data sharing in IoT [DCN18, ZPW+15]. 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 [ON20], 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 [Coh19] 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 [Reg05,LPR10], (R)LWE, notably [PRSV17], 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 [Coh19], Li and Micciancio [LM21] applied a very similar RLWE attack to an approximate FHE scheme, namely, the Cheon–Kim–Kim–Song (CKKS) scheme [CKKS17], where the adversary gets a somewhat restricted decryption oracle, introduced as part of a new IND-CPA^D security definition [LM21, 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 [LMSS22], but the latter introduced a fine-grained flooding technique for optimal parameters mixing both statistical and computational security, whereas the former relied on loose, theoretical noise flooding bounds [AJL+12]. 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) [Bra12, FV12] and Brakerski-Gentry-Vaikuntanathan (BGV) [BGV14] 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. [LMSS22] 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, provable, 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 [AAB+22]. 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 [Goo] for multiple hops where an AES symmetric key is the data payload. We perform network simulation using the open-source RAVEN framework [Ins]. For the single-hop setting, the re-encryption time in OpenFHE on an Intel® Core™ i7-9700 CPU with 64 GB RAM, a commodity desktop machine, for our HRA-secure PRE scheme is about 2 milliseconds. The timing for a 13-hop

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

parameter set starts with 103 milliseconds for re-encrypting a fresh 6.5MB ciphertext, and ultimately drops down to 32 milliseconds for the last (13^{th}) hop. Our PRE scheme implementation is publicly available as part of the OpenFHE library [AAB+22]. Our networking application system implementation is also publicly available in a separate OpenFHE project repository [Dua].

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. [dCJV21] 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 [DDLM19], which uses simple ciphertext re-randomization (adding small encryption noise) without noise flooding, i.e., without adding large noise, is not HRA-secure despite their claims. (See Appendices B and C for more details.)

Connections to threshold and approximate FHE. Our work is closely related to the state of the art in threshold FHE [AJL+12, KS23] and approximate FHE [LMSS22] 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 [Gen09] 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) [BGP+17, PRSV17] system whose underlying PRE scheme does not provide HRA security. We fix this by applying the fine-grained noise flooding technique of [LMSS22] (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 [dlPVA22] but are not post-quantum. Lattice-based ABE schemes are not efficient [DDP+18, GPR+19, GMP19] but offer richer access policies than PRE. For example, the runtimes presented in Table 2 of [DDP+18] (for a very small number of Boolean attributes and binary messages) are significantly larger (often by orders of magnitude) than the PRE runtimes (for much larger messages) shown in Tables 2 and 4 of our paper.

Fine-grained PRE, first constructed by Zhou et al. [ZLHZ23] in the single-hop CPA-secure setting and later improved to the multi-hop HRA setting [ZLH24], are PRE schemes where the message, m, gets transformed to a known function, $f(\mathsf{m})$. The constructions in [ZLHZ23, ZLH24] are based on lattice trapdoors [GPV08, MP12], 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 [ZLHZ23] nor [ZLH24] 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 [MW17] 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 [DDLM19] presents a PRE scheme as an extension of the scheme in [PRSV17] 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 [PRSV17] 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 [DDLM19] using binary matrix (R)LWE attacks [HM17] together with an averaging attack.

Fuchsbauer et al. [FKKP19] 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-

[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.

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 [DS16], in addition to noise flooding, to achieve this for lattice-based PRE schemes. They did not implement their scheme.

An even more powerful PRE scheme is universal PRE, where re-encryption is done between *any* public key scheme. Döttling and Nishimaki [DN21] achieve this by using either (probabilistic) indistinguishability obfuscation or garbled circuits over the PKE schemes (not practically efficient).

Susilo et al. [SDDR21] 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 the decisional bilinear Diffie–Hellman (DBDH) problem were presented in [AFGH06, CH07, ID03]. The scheme in [AFGH06] is IND-CPA secure and provides low performance run-times for 256 and 512 bits of classical security.

Several papers focused on noise flooding in threshold FHE. For example, Chowdhurry et al. claimed they were able to develop the first practical TFHE scheme with a polynomial modulus-to-noise ratio [CSS+22]. However, their claim was subsequently shown to be incorrect (see the prior works discussion in [PS25] for more details). If more than one round is allowed, flooding noise can be avoided using extra interactions during distributed decryption [PS25]. Further details on the issue of noise flooding in threshold FHE schemes are provided in [BGG+18, MBH23]. An extension of threshold FHE to threshold lattice-based signatures is discussed in [GKS24].

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 logic for setting the parameters is explained in Section 5. The experimental results are presented in Section 6, followed by concluding remarks in Section 7. Appendix A discusses the details of key switching and BGV scheme optimizations. Appendix B shows the necessity of noise flooding in RLWE schemes based on the PRSV scheme. We explore alternatives to noise flooding in Appendix C. The rest of the appendices discusses the details of our implementation for the networking use case.

2 Preliminaries

We use λ to denote the computational security parameter and, if applicable, ν to represent a statistical security parameter. A function, $f : \mathbb{N} \to \mathbb{R}$, 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 Concrete Security

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

Definition 1. *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 [MW18]. First, we define a generic distinguishing game, encompassing CPA and HRA security for PRE.

Definition 2 ([MW18]). *Let $\{\mathcal{D}_\theta^0\}_\theta, \{\mathcal{D}_\theta^1\}_\theta$ 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 $\mathsf{c}_i \leftarrow \mathcal{D}_{\theta_i}^b$. 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 $\mathcal{G}(\{\mathcal{D}_\theta^0\}_\theta, \{\mathcal{D}_\theta^1\}_\theta)$ 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 3 .

Definition 3 ([MW18]). *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. [LMSS22] 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 [DEF+19]). We abbreviate the time of an adversary as $T(\mathcal{A})$.

Definition 4 ((c, s) security [LMSS22]). *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 $c \leq \log_2 \frac{T(\mathcal{A})}{\mathsf{Adv}^{\mathcal{A}}}$ or $s \leq \log_2 \frac{1}{\mathsf{Adv}^{\mathcal{A}}}$.*

We use (λ, ν)-security to denote the security in Definition 4 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 [LMSS22]). *Let $\mathcal{G}$ be an indistinguishability game (Definition 2) 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 [MW18]). *Let $\{\mathcal{H}_i\}_{i=1}^k$ be k distribution ensembles 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 $\epsilon_{i,j} = \max_{\mathcal{A}} \mathsf{Adv}^{\mathcal{A}}_{\mathcal{G}_{i,j}}$ with $T(\mathcal{A}) \leq C$. Then, $\epsilon_{1,k} \leq 3k \sum_{i=1}^{k-1} \epsilon_{i,i+1}$.*

2.2 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 [Coh19], 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 5 (Proxy Re-Encryption (PRE) Scheme). *A* proxy re-encryption scheme *(PRE) for a message space $\mathcal{M}$ is a tuple of algorithms* ($\mathsf{ParamGen}$, KeyGen, Enc, Dec, $\mathsf{ReKeyGen}$, ReEnc)*:*

- $\mathsf{pp} \leftarrow \mathsf{ParamGen}(1^\lambda, 1^\nu, h)$*: Given a security parameter λ, a statistical security parameter ν, and the maximum number of hops (re-encryptions) h, 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.*

[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.

$\mathsf{ct}_j \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct}_i)$*: 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 6 (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* $\mathsf{rk}_{i\to j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$*:*

$$\mathsf{Dec}(\mathsf{sk}_j, \mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{Enc}(\mathsf{pk}_i, \mathsf{m}))) = \mathsf{m}$$

A PRE scheme is h-hop correct if, in addition, with high probability over $\{(\mathsf{pk}_i, \mathsf{sk}_i), (\mathsf{pk}_j, \mathsf{sk}_j)\}_{i,j\in\mathcal{I}}$ *and* $\{\mathsf{rk}_{i\to j}\}_{i,j\in\mathcal{I}}$ *generated as above, then*

$$\mathsf{Dec}(\mathsf{sk}_{j_h}, \mathsf{ReEnc}^h(\{\mathsf{rk}_{j_i\to j_{i+1}}\}, \mathsf{Enc}(\mathsf{pk}_{j_0}, \mathsf{m}))) = \mathsf{m}$$

with high probability for an index set $\mathcal{I}$ *of size at least* $h+1$*,* $\{j_i\} \subseteq \mathcal{I}$ *where* $\mathsf{ReEnc}^h(\cdot,\cdot)$ *represents re-encryption composed* h *times.*

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 6 for discussion on decryption failure rate of our PRE scheme. Now we define HRA security.

Definition 7 (HRA Security Game, Definition 5 in [Coh19]). *Fix some* λ, ν, h *and let* $\mathcal{A}$ *denote some PPT adversary. The* HRA security game *consists of running* $\mathcal{A}$ *with the keys generated in Phase 1 and the distributions defined in Phase 2 below. Note,* $\mathsf{numKeys}$ *will count the number of public keys, each identified with the counter as they are generated, the same for ciphertexts and* numCts*. The sets* Hon *and* Cor *represent honest and corrupt keys, respectively, and* Deriv *will represent the ciphertexts derived from the challenge ciphertext in Phase 2.*

Phase *1:*

- *Setup: The public parameters* $\mathsf{pp} \leftarrow \mathsf{ParamGen}(1^\lambda, 1^\nu, h)$ *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* $\mathsf{Deriv} \leftarrow \emptyset$ *where* C *denotes all ciphertexts and* Deriv *denotes all ciphertexts re-encrypted from the challenge ciphertext.*

- ◇ *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* numKeys *is added to* Hon *and* 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* numKeys *is added to* Cor *and* numKeys *is incremented.*

Phase 2***:*** *For each pair* $i, j \leq \mathsf{numKeys}$*, compute the re-encryption key* $\mathsf{rk}_{i\to j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$*. Then, sample a bit* $\mathsf{b} \leftarrow \{0,1\}$ *uniformly at random. The distinguishing game is defined by the following oracles. The oracles return* $\perp$ *for any invalid query*

- ◇ *Re-encryption Key Generation* $\mathcal{O}_{\mathsf{ReKeyGen}}(i,j)$*: 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* $\mathsf{rk}_{i\to j}$*. Denote the distribution observed by* $\mathcal{A}$ *as* $\mathsf{RK}^b_\theta = \mathsf{RK}^b_{(i,j)} = \{\mathsf{rk}_{i\to j}\}$*.*
- ◇ *Encryption* $\mathcal{O}_{\mathsf{Enc}}(i,\mathsf{m})$*: 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* $(\mathsf{numCt}, \mathsf{ct})$*. Denote the distribution observed by* $\mathcal{A}$ *as* $\mathsf{E}^b_\theta = \mathsf{E}^b_{(\mathsf{pk}_i,\mathsf{m})} = \{(\mathsf{numCt}, \mathsf{ct})\}$*.*
- ◇ *Challenge Oracle* $\mathcal{O}_{\mathsf{ch}}(i, \mathsf{m}_0, \mathsf{m}_1)$*: On input* $(i, \mathsf{m}_0, \mathsf{m}_1)$ *where* $i \in \mathsf{Hon}$ *and* $\mathsf{m}_0, \mathsf{m}_1 \in \mathcal{M}$*, 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. Denote the distribution observed by* $\mathcal{A}$ *as* $\mathsf{CH}^b_\theta = \mathsf{CH}^b_{(i,\mathsf{m}_0,\mathsf{m}_1)} = \{(\mathsf{numCt}, \mathsf{ct}^*)\}$*.*
- ◇ *Re-encryption* $\mathcal{O}_{\mathsf{ReEnc}}(i,j,k)$*: 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* $(\mathsf{numCt}, \mathsf{ct}_j)$*. Denote the distribution observed by* $\mathcal{A}$ *as* $\mathsf{R}^b_\theta = \mathsf{R}^b_{(i,j,k)} = \{(\mathsf{numCt}, \mathsf{ct}_j)\}$*.*

Phase 3***:*** *The adversary guesses* b*, or outputs* $\perp$*, after observing* $\{(\mathsf{RK}^b_\theta, \mathsf{E}^b_\theta, \mathsf{CH}^b_\theta, \mathsf{R}^b_\theta)\}_\theta$*.*

We say a PRE scheme is (λ,ν)-bit secure if for all adversaries $\mathcal{A}$, $\lambda \leq \log_2(\frac{T(\mathcal{A})}{\mathsf{Adv}^{\mathcal{A}}})$ or $\nu \leq \log_2(\frac{1}{\mathsf{Adv}^{\mathcal{A}}})$. The 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 $j \in \mathsf{Cor}$. We say a PRE-scheme is HRA secure *with* q *queries* if $\mathcal{O}_{\mathsf{ReEnc}}(i,j,k)$ is queried at most q times.

Definition 8 (Def. 7 in [Coh19]). *A* PRE *scheme is* (λ,ν)-re-encryption simulatable *if there exists a simulator* 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 under KL-divergence 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:* $\mathsf{pp} \leftarrow \mathsf{ParamGen}(1^\lambda, 1^\nu, h)$*,* $(\mathsf{pk}_a, \mathsf{sk}_a) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$*,* $(\mathsf{pk}_b, \mathsf{sk}_b) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$*,* $\mathsf{rk}_{a\to b} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_a, \mathsf{pk}_b)$*,* $\mathsf{ct}_a \leftarrow \mathsf{Enc}(\mathsf{pk}_a, \mathsf{m})$*.*

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 [Coh19]). *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 [Coh19] on PICADOR [BGP+17] is nearly the same attack as Li and Micciancio's attack on the CKKS scheme [LM21]. Breaking this correlation is crucial to HRA security.

2.3 BGV Homomorphic Encryption Scheme

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.

RLWE Background. Here we describe the necessary RLWE-related technical background needed to understand our PRE scheme based on the BGV homomorphic encryption 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. The secret distribution χ_k is assumed to be the distribution of polynomials in $R = \mathbb{Z}[X]/(X^N + 1)$ with coefficients in $\{0, \pm 1\}$ chosen uniformly at random.

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.*

Our noise flooding distribution is the (standard) discrete Gaussian, denoted as $D_{\sigma_{fl}}$. The noise distribution D_{σ_e} is a discrete Gaussian (Definition 9) of width 3.19 [ACC+19]. Discrete Gaussians can be efficiently sampled for relatively small σ's, and for the parameters we need for noise flooding, in constant time [MW17].

Key Generation. A secret key is $\mathsf{sk} = s$ for $s \leftarrow \chi_k$ and 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.

[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.

Encryption. BGV public key encryption for a given message $\mathsf{m} \in R_p$ and a public key pk is done by sampling $v \leftarrow \chi_k, e, e' \leftarrow D_{\sigma_e}$, and returning

$$\mathsf{ct} = (v \cdot \mathsf{pk}_0 + pe', v \cdot \mathsf{pk}_1 + \mathsf{m} + pe) = (c_0, c_1).$$

Decryption. 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 $(-p/2, p/2]$.

BGV works with a chain of distinct NTT-friendly moduli $Q_1, \ldots, Q_l, \ldots, Q_L$, where $Q_l | Q_{l+1}$ for $l = 1, \ldots, L-1$. The index l denotes the ciphertext level.

Modulus switching. The main noise-control method in BGV encryption is modulus switching, defined as follows:

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 evaluates*

$$\mathsf{ct}' \leftarrow (Q'/Q) \cdot (\mathsf{ct} + \delta) \in R^2_{Q'},$$

where $\delta = p \cdot ([-c_0/p]_D, [-c_1/p]_D) \in R^2$.

Brakerski et al. [BGV14], 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 $\|e'\| \leq \|e\|_\infty / D + \frac{1+\delta_R}{2}$, where δ_R is the expansion factor introduced in [FV12]. Note that $\delta_R = N$ corresponds to the worst-case bound. Halevi et al. [HPS19] heuristically showed (using subgaussian analysis) that $\delta_R = 2\sqrt{N}$ can be used in practice instead, while still achieving practically negligible probability of decryption failure. We denote the algorithm in Definition 10 as: $\mathsf{ct}' \leftarrow \mathsf{ModSwitch}_Q^{Q'}(\mathsf{ct})$.

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. The details of key switching, including both Brakerski-Vaikuntanathan (BV) and hybrid methods, and related optimizations are described in Appendix A.

3 Our HRA-Secure PRE Scheme

Our proposed PRE scheme is an HRA-secure extension of the scheme in [PRSV17]. We rely on the tight noise-flooding analysis of [LMSS22] for precise security estimates. This yields an efficient PRE scheme with HRA security. We show that our scheme is HRA-secure for our target application, both with a single hop and multiple hops, and provide its tight security analysis.

Although we describe and implement the scheme based on the BGV homomorphic encryption scheme [BGV14], the same underlying ideas can be used to construct an efficient, HRA-secure PRE scheme with BFV (Brakerski, Fan, Vercauteren [Bra12]) or CKKS (Cheon, Kim, Kim, Song [CKKS17]) encryption. This is possible because both BFV and CKKS use the same key switching mechanism as BGV. Two other common FHE schemes, Ducas–Micciancio (DM/FHEW) [DM15] and Chilotti–Gama–Georgieva–Izabachène (CGGI/TFHE) [CGGI16] schemes, typically use a different method of key switching, which is more challenging for building an HRA-secure PRE on top of them. In the case of DM/CGGI, one can construct PRE via bootstrapping using the blueprint originally proposed by Gentry [Gen09].

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 [Coh19]. Noise flooding [AJL+12] 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 [MW17,MW18,LMSS22]. The conclusion derived by Li et al. [LMSS22] 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 [DEF+19]). 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 [LMSS22].

3.1 Our Construction

Our scheme is presented in Algorithms 1–6. Recall, a PRE scheme consists of the algorithms (ParamGen, KeyGen, Enc, Dec, ReKeyGen, ReEnc) (Definition 5). Our ParamGen, KeyGen, Enc, Dec algorithms are the same as in the IND-CPA secure scheme in [PRSV17], i.e., they correspond to standard BGV public key encryption, but we modify the 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. [LMSS22]. We denote encrypting a vector of messages, $\mathbf{m} \in R^k$, or the k-repeated public-key encryption algorithm, as

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

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 5 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: $\mathsf{rk}_{s \mapsto s^*} = (\boldsymbol{rk}_0, \boldsymbol{rk}_1) \leftarrow \mathsf{Enc}(\mathsf{pk}^*, \mathcal{PW}_l(s))$
2: **return** $\mathsf{rk}_{s \mapsto s^*}$

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

Input: A ciphertext $\mathsf{ct} \in R_{Q_l}^2$ encrypted under s, a re-encryption key $\mathsf{rk}_{s \mapsto s^*}$ 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: 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: $\mathsf{ct}^{(2)} \leftarrow \mathsf{KeySwitch}(\mathsf{ct}^{(1)}, \mathsf{rk}_{s \mapsto s^*})$.
5: Modulus switch: $\mathsf{ct}^{(3)} \leftarrow \mathsf{ModSwitch}_{Q_l}^{Q_{l-1}}(\mathsf{ct}^{(2)})$.
6: **return** $\mathsf{ct}^* = \mathsf{ct}^{(3)}$.

$\mathbf{ct} = (\mathbf{ct}_0, \mathbf{ct}_1) \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{m})$ where each ct_i is a fresh public key encryption of m_i ($\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, m_i)$).

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 [Coh19]). In short, this re-randomization breaks the output ciphertext's correlation with the input ciphertext's secret key. This correlation is why the scheme from [PRSV17] 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 [LM21]. Analogously, we use the refined flooding technique from Li et al. [LMSS22], 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. If the (re-randomized) input ciphertext's noise is e, then the output of Algorithm 6 ciphertext's noise is at most $\frac{Q_{l-1}}{Q_l}(\|e\|_\infty + \|e_{re}\|_\infty + \|e_{\mathsf{ks}}\|_\infty) + \frac{1+\delta_R}{2}$, where e_{re} is flooding noise in Algorithm 6.

3.2 The Concrete Security of Our HRA-Secure PRE Scheme

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 8) with KL divergence $\leq \rho$, then the same scheme is $(\lambda - \log_2 24, \log_2(1/\rho) - \log_2(\tau) - \log_2 24)$ HRA secure with at most τ queries.*

Proof. 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 [LMSS22], 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 $\mathcal{G}_0$.

Let $\mathcal{G}_j^b$ be the distribution observed by the adversary in game $\mathcal{G}_j$ with secret bit b. We show that $\mathcal{G}_0^0$ is indistinguishable from $\mathcal{G}_0^1$, proving HRA security. 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 $\mathcal{H}_4 = \mathcal{G}_0^1$. 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, here we move between the actual game and the simulated query game by fixing b. Next,

we consider $\epsilon_{2,3} = \max_{\mathcal{B}} \mathsf{adv}^{\mathcal{B}}_{\mathcal{G}_1}$ and solve for the computational and statistical loss in the reduction.

We have $\max_{\mathcal{A}} \mathsf{adv}^{\mathcal{A}}_{\mathcal{G}_0} \leq 12(\max_{\mathcal{B}} \mathsf{adv}^{\mathcal{B}}_{\mathcal{G}_1} + \tau\rho) \leq 24 \max(\max_{\mathcal{B}} \mathsf{adv}^{\mathcal{B}}_{\mathcal{G}_1}, \tau\rho)$. Now we consider both cases.

1. If $\tau\rho \geq \max_{\mathcal{B}} \mathsf{adv}^{\mathcal{B}}_{\mathcal{G}_1}$, then $\max_{\mathcal{A}} \mathsf{adv}^{\mathcal{A}}_{\mathcal{G}_0} \leq 24\tau\rho$. Or, equivalently:

$$\min_{\mathcal{A}} \log_2(1/\mathsf{adv}^{\mathcal{A}}_{\mathcal{G}_0}) \geq \log_2(1/\rho) - \log_2(24\tau).$$

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

$$\max_{\mathcal{A}} \mathsf{adv}^{\mathcal{A}}_{\mathcal{G}_0} \leq 24 \max_{\mathcal{C}} \mathsf{adv}^{\mathcal{C}}_{\mathcal{C}_2}.$$

   Or, we have a computational security loss of $\log_2(24)$ bits.

Noise Flooding According to Corollary 2 of [LMSS22][9], we must add a discrete Gaussian with standard deviation $\sigma = \sqrt{12\tau} t 2^{\nu/2}$ 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 [LMSS22]). *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 $\mathsf{ReEncSim}(\mathsf{ct}_s, \mathsf{pk}^*_s, m)$ 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 message m.

Output: A simulated ciphertext $\mathsf{ct}^* \in R^2_{Q_{l-1}}$ encrypting the same message as ct_s under s^* with a noise distribution close to the output of Algorithm 6.

1: $e \leftarrow D_{R,\sigma}$ for $\sigma = \sqrt{12\tau} 2^{\nu/2} \mathsf{ct}_s.t$ where $\mathsf{ct}_s.t$ 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^*.

[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.

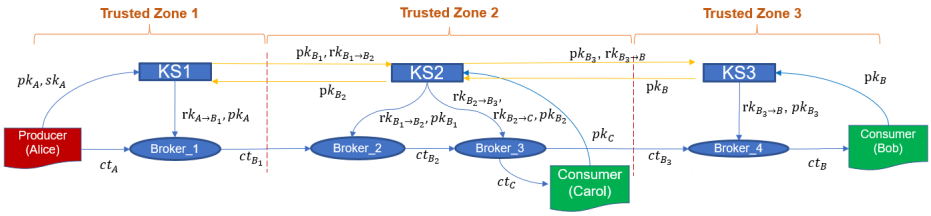

Fig. 1: Example PRE network with three trust zones, one Key Server for each zone, one producer, two consumers, and four brokers. Public, Secret, Re-encryption Keys and ciphertext exchanges are shown as connecting arrows. Secure exchange of keys between trust zones is shown in yellow.

Note that the real noise in the output of Algorithm 6 is $\frac{e_{\text{flood}}+e_{\text{KS}}}{q_{\text{drop}}} + \tau_0' + \tau_1' s^*$ whereas the noise in the output of the simulator, Algorithm 7, is $\frac{e_{\text{flood}}+e_{\text{fresh}}}{q_{\text{drop}}} + \tau_0 + \tau_1 s^*$ 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.

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 Section 6.3. There could also be a setting where the key authorities generate keys for producers and consumers as well, depending on the application.

5 Parameter Selection

We implemented the scheme presented in Section 3 for three different security modes: CPA-Secure, Bounded-Query HRA*-secure, and HRA-secure. The three modes use the same ReKeyGen algorithm, Algorithm 5, to generate re-encryption keys and only differ in their ReEnc algorithms.

IND-CPA-Secure Mode. 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 [PRSV17], but adapted to the public-key setting.

Bounded-Query (Low ν) HRA*-Secure Mode. To achieve a trade-off between performance and security, we implemented the Bounded-Query HRA*-Secure mode[10] that adds a fixed 20-bit noise in 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 3.2 means that the (λ, ν)-HRA-security is about $(128, 20)$ if the adversary gets $2^9 = 512$ re-encryption 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 Mode. This mode is IND-HRA secure and implements Algorithm 6 as described, with noise flooding. It supports both BV and hybrid key switching (see Appendix A for more details on both key switching methods). For the concrete security of the scheme in Section 3.2, the noise flooding parameter needs to factor in the number of re-encryption queries and the desired

[10] HRA-secure mode with a low ν provides better efficiency than the HRA-Secure mode with a higher ν but limited protection against re-encryption attacks.

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.

5.1 Logic for Setting the Parameters

Our PRE scheme supports multiple hops, but the choice of optimal parameters depends on many factors: security level required and security mode (CPA, bounded-query HRA*-secure, 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 [ACC+19] 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. Note that updated security guidelines are now available [BCC+24], which contain slightly revised thresholds for $\log_2 Q_L$ for a given ring dimension N. However, [BCC+24] is new (the updated tables were published in October 2024) and has not been adopted by the FHE community; for example, OpenFHE, the software library we employed for the experiments, currently uses the parameter thresholds of [ACC+19].

For a given Q_L and N, we estimate 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 Appendix A for details on key switching and its parameters). 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 [PRSV17]). 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 its 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 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 is 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 [ACC+19].
  - 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.

The bound $B = \alpha\sigma$ for noise from distribution D_σ determines the decryption failure rates. Since $erfc(\alpha) \approx 2^{-55}$ for $\alpha = 6$, where $erfc(z)$ is the complementary error function for z, 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^{-40} while using a union bound with ring dimension up to 2^{15}. Hence, we choose $\alpha = 6$ in our implementation to target a decryption failure rate of at most 2^{-40} [GHS12b]. The quality of the discrete Gaussian samples for noise flooding is verified using the GLITCH framework [HO17].

6 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 for all three PRE modes: IND-CPA-secure, fixed-noise (bounded-query, low ν) HRA*-secure, and provably secure HRA. For all experiments, we used an Intel® Core™ i7-9700 CPU with 64 GB RAM, running Ubuntu 20.04 with g++ v10.5.0. All experiments were run in the single-threaded mode using OpenFHE v1.2.0. We first present the results for the single-hop scenario and then report our results for 13 hops for the use case of secure multi-hop information sharing described in Section 4.

6.1 Single-Hop Setting

The ciphertext expansion at different payload bit sizes for a single-hop PRE is shown in Table 1 for the three security options. To measure the ciphertext size, we use the size of serialized ciphertexts generated using the binary serialization mode of OpenFHE [AAB+22]. 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 moduli do not allow for one hop when the digit size is larger than 1 for ring dimension $N = 1024$.

Payload (bits)	N	$\log Q(P)$	p	Digit size r	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
Low ν (fixed 20-bit 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	56	65.0 KB	520.0
2048			4			260.0
4096			16			130.0
8192			256			65.0
16384			65536			32.5

Table 1: Single-hop ciphertext expansion (the ratio of plaintext size vs re-encrypted ciphertext size). For IND-CPA and bounded query (low ν) HRA-secure modes, BV key switching is used. For the provable HRA-secure mode, the key switching technique is set to hybrid, $\nu = 48$, and $\tau = 2^{18}$, P is auxiliary modulus used in hybrid key switching (see Appendix A for details).

So we use the digit size of 1 for comparison with different values of p until the plaintext modulus is large enough to require raising the ring dimension to 2048 for a single hop. Figure 1 suggests that the smallest ciphertext expansion factor for the IND-CPA-secure and low-ν HRA*-secure modes is about 16, and the corresponding expansion factor for the HRA-secure mode is about 32.

Figure 2 presents the runtimes for all three modes at $p = 2$, which corresponds to the AES secret key sharing use case. The IND-CPA and low-ν HRA*-secure mode have approximately the same runtimes except for the re-encryption, where the low-ν HRA*-secure mode adds a Gaussian with a 20-bit distribution parameter.

6.2 Multi-Hop Setting

For the multi-hop setting, all parameters are chosen to allow a minimum of 13 hops with 128 bits of computational security. We fix the plaintext modulus at $p = 2$ for all our multihop experiments as we focus on the application of key encapsulation that transfers 256-bit AES keys from producers to consumers. The AES key is treated as a vector of bits when encoding the message.

Table 3 presents the parameters for different security modes, along with maximum number of hops supported for each mode, public key size, re-encryption key size and initial re-encrypted ciphertext size. The re-encryption key size is

Security mode	KeyGen (KS)	ReKeyGen (KS)	Enc (Producer)	ReEnc (Broker)	Dec (Consumer)
IND-CPA	0.21	0.50	0.18	0.19	0.032
Bounded HRA*	0.21	0.50	0.18	0.54	0.032
HRA-Secure	1.05	1.00	0.73	2.04	0.142

Table 2: Single-threaded runtime performance (in milliseconds) for different modes of the PRE scheme for the single-hop setting. The plaintext modulus p is set to 2. For IND-CPA and bounded-query HRA*-secure modes, $N = 1024$ and $\log Q = 27$ (both use BV key switching); for the HRA-secure mode, $N = 4096$ and $\log QP = 109$ (other parameters are the same as for Table 1). Each algorithm is labeled with the network node name in parentheses (ReEnc is done by Brokers, etc.) The runtimes are averages over 100 runs, the variability was less than 10% across all runs.

Security mode	N	$\log Q(P)$	Max hops	pk	rk	ReEnc ct	ct reduction
IND-CPA	2048	54	+	32.65 KB	96.93 KB	32.80 KB	-
Bounded HRA*	2048	54	+	32.65 KB	96.93 KB	32.80 KB	-
HRA-Secure	32768	815	13	8.5 MB	25.5 MB	6.5 MB	0.50 MB/hop

Table 3: Parameters and resulting key sizes for a minimum of 13 hops for all security modes, plaintext modulus $p = 2$ and a digit size of 3. We use + to denote a practically unlimited number of hops (over a million). For the provable HRA-secure mode, $\nu = 48$, $\tau = 2^{18}$, the key switching is hybrid, and the public key uses the extended modulus QP to reduce the noise added as part of fresh public key encryption. ReEnc ct stands for the re-encrypted ct largest size.

Security mode	KeyGen (KS)	ReKeyGen (KS)	Enc (Producer)	ReEnc (Broker)	Dec (Consumer)
IND-CPA	0.38	1.02	0.35	0.39	0.086
Bounded HRA*	0.38	1.02	0.35	1.10	0.086
HRA-Secure	51.2	124	38.6	from 103 to 32	from 20.7 to 1.2

Table 4: Single-threaded runtime performance (in milliseconds) for different PRE modes for at least 13 hops (see the caption in Table 3 for other parameter values). Each algorithm is labeled with the network node name in parentheses (ReEnc is done by Brokers, etc.) The runtimes are averages over 100 runs, the variability was less than 10% across all runs.

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 required modulus size and 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 modulus switching 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. The ciphertext size is initially 6.5 MB and then with every hop it reduces by 0.5 MB, resulting in the ciphertext size of 0.5 MB after the last (13^{th}) re-encryption.

Table 4 shows the runtime performance of all PRE scheme operations. Note that the key sizes, ciphertext sizes and runtimes in Tables 3 and 4 are larger for the provably-secure HRA option. This is due to the larger ring dimension and modulus $Q(P)$ 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 4 shows that the re-encryption runtime decreases from 103 milliseconds for the first hop down to 32 milliseconds for the last hop.

6.3 Simulated Secure Data Distribution System

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 256-bit AES keys from multiple producers to multiple authorized consumers using gRPC's [Goo] 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 [Ins]. 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 was flexible: the virtual machines could be configured according to the application and the resources of the host machine.

We configured routers and switches to simulate a real-world network: each trust zone was connected through a router and each router had a switch that multiple services in the same trust zone could connect to, i.e., they were in the same subnet mask. Services made function calls to the OpenFHE PRE functionality to distribute keys. More details on the setup are provided in Appendix D.

We used an AMD EPYC 7302 16-Core Processor machine with 500 GB memory as the host machine. Each service was run on virtual machines that were set up to be nodes with 2 cores and 4 GB memory. This was 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 were then used in the actual execution to run the system.

We validated the functionality and correctness of the multi-hop PRE protocol using the simulated secure distribution system. As the simulation was run on the same host machine (with high-bandwidth communication), the timing

results were dominated by the computational complexity of OpenFHE operations. Instead of discussing the runtime results of this experiment in detail (the runtimes are comparable to those in Table 4), here we examine the tradeoff between the computational complexity and communication costs associated with the PRE protocol based on the results already reported in Tables 3–4.

The main online operation is proxy re-encryption (as the generation of the re-encryption key is done offline). If we use the HRA-secure parameters that support up to 13 hops, then one ciphertext has the size from 0.5 MB (last hop) up to 6.5 MB (first hop), as shown in Table 3. An AES-256 key fully fits in a single ciphertext. If the communication link (to a broker) has a capacity of 1Gbps, then roughly 100 MB can be transferred per second. This implies that one ciphertext has the communication latency between 5 (last hop) and 65 (first hop) milliseconds; for a 100Mbps link, these become 50 and 650 milliseconds, respectively. Table 4 suggests that the single-threaded runtime of re-encryption at the broker is between 32 (last hop) and 103 (first hop) milliseconds. As ciphertexts need to be sent to the broker twice (ingress and egress), we can conclude that for 1 Gbps links, the communication time is roughly the same as the single-threaded computation time for proxy re-encryption (on the order of 100 milliseconds for a 13-hop configuration).

When the number of hops is reduced, both the ciphertext size and communication size scale quasi-linearly with N and $\log_2 Q_L$, which implies that the ratio is expected to be preserved, i.e., at 1Gbps, the communication time is comparable to the proxy re-encryption time. If we look at the extreme case of 1 hop (Tables 1-2), then the proxy re-encryption computation is expected to take 2 milliseconds, while the communication time for 65 KB over a 1Gbps link is expected to be around 0.65 milliseconds, i.e., 1.3 milliseconds for the combined ingress and egress links.

6.4 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.

7 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 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.

A future direction of research is showing a tight connection between HRA security, CPA^{D} security [LMSS22], and multi-party threshold decryption and their relation to FHE schemes featuring circuit privacy and key homomorphism, ideally using a form of (c, s) security (Definition 4). Kluczniak and Santato [KS23] showed some relations for approximate FHE, e.g., CKKS, circuit privacy and threshold/multi-key FHE. However, a complete analysis between all the aforementioned security notions, independent of the underlying FHE scheme, is an open problem. We suspect that any circuit-private key-homomorphic FHE scheme can be shown to imply these security definitions.

References

[AAB+22] Ahmad Al Badawi, Andreea Alexandru, Jack Bates, Flavio Bergamaschi, David Bruce Cousins, Saroja Erabelli, Nicholas Genise, Shai Halevi, Hamish Hunt, Andrey Kim, Yongwoo Lee, Zeyu Liu, Daniele Micciancio, Carlo Pascoe, Yuriy Polyakov, Ian Quah, Saraswathy R.V., Kurt Rohloff, Jonathan Saylor, Dmitriy Suponitsky, Matthew Triplett, Vinod Vaikuntanathan, and Vincent Zucca. Openfhe: Open-source fully homomorphic encryption library. Cryptology ePrint Archive, Paper 2022/915, 2022. `https://eprint.iacr.org/2022/915`.

[ACC+19] Martin R. Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin E. Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. Homomorphic encryption standard. *IACR Cryptol. ePrint Arch.*, page 939, 2019.

[AFGH06] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, 9(1):1–30, 2006.

[AJL+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 483–501. Springer, 2012.

[BBS98] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *EUROCRYPT*, volume 1403 of *Lecture Notes in Computer Science*, pages 127–144. Springer, 1998.

[BCC+24] Jean-Philippe Bossuat, Rosario Cammarota, Ilaria Chillotti, Benjamin R. Curtis, Wei Dai, Huijing Gong, Erin Hales, Duhyeong Kim, Bryan Kumara, Changmin Lee, Xianhui Lu, Carsten Maple, Alberto Pedrouzo-Ulloa, Rachel Player, Yuriy Polyakov, Luis Antonio Ruiz Lopez, Yongsoo Song, and Donggeon Yhee. Security guidelines for implementing homomorphic encryption. Cryptology ePrint Archive, Paper 2024/463, 2024.

[BGG+18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 565–596. Springer, 2018.

[BGP+17] Cristian Borcea, Arnab Deb Gupta, Yuriy Polyakov, Kurt Rohloff, and Gerard W. Ryan. PICADOR: end-to-end encrypted publish-subscribe information distribution with proxy re-encryption. *Future Gener. Comput. Syst.*, 71:177–191, 2017.

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theory*, 6(3):13:1–13:36, 2014.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 868–886. Springer, 2012.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106. IEEE Computer Society, 2011.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In Phillip Rogaway, editor, *CRYPTO 2011*, pages 505–524, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[CGGI16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 3–33, 2016.

[CH07] Ran Canetti and Susan Hohenberger. Chosen-ciphertext secure proxy re-encryption. In *CCS*, pages 185–194. ACM, 2007.

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In *ASIACRYPT (1)*, volume 10624 of *Lecture Notes in Computer Science*, pages 409–437. Springer, 2017.

[Coh19] Aloni Cohen. What about bob? the inadequacy of CPA security for proxy reencryption. In *Public Key Cryptography (2)*, volume 11443 of *Lecture Notes in Computer Science*, pages 287–316. Springer, 2019.

[CSS+22] Siddhartha Chowdhury, Sayani Sinha, Animesh Singh, Shubham Mishra, Chandan Chaudhary, Sikhar Patranabis, Pratyay Mukherjee, Ayantika Chatterjee, and Debdeep Mukhopadhyay. Efficient threshold FHE for privacy-preserving applications. Cryptology ePrint Archive, Paper 2022/1625, 2022.

[dCJV21] Leo de Castro, Chiraag Juvekar, and Vinod Vaikuntanathan. Fast vector oblivious linear evaluation from ring learning with errors. In *WAHC@CCS*, pages 29–41. WAHC@ACM, 2021.

[DCN18] Abebe Abeshu Diro, Naveen K. Chilamkurti, and Yunyoung Nam. Analysis of lightweight encryption scheme for fog-to-things communication. *IEEE Access*, 6:26820–26830, 2018.

[DDLM19] Alex Davidson, Amit Deo, Ela Lee, and Keith Martin. Strong post-compromise secure proxy re-encryption. In *ACISP*, volume 11547 of *Lecture Notes in Computer Science*, pages 58–77. Springer, 2019.

[DDP+18] Wei Dai, Yarkin Doröz, Yuriy Polyakov, Kurt Rohloff, Hadi Sajjadpour, Erkay Savas, and Berk Sunar. Implementation and evaluation of a lattice-based key-policy ABE scheme. *IEEE Trans. Inf. Forensics Secur.*, 13(5):1169–1184, 2018.

[DEF+19] Ivan Damgård, Daniel Escudero, Tore Kasper Frederiksen, Marcel Keller, Peter Scholl, and Nikolaj Volgushev. New primitives for actively-secure MPC over rings with applications to private machine learning. In *IEEE Symposium on Security and Privacy*, pages 1102–1120. IEEE, 2019.

[dlPVA22] Antonio de la Piedra, Marloes Venema, and Greg Alpár. ABE squared: Accurately benchmarking efficiency of attribute-based encryption. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(2):192–239, 2022.

[DM15] Léo Ducas and Daniele Micciancio. FHEW: bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 617–640. Springer, 2015.

[DN21] Nico Döttling and Ryo Nishimaki. Universal proxy re-encryption. In *Public Key Cryptography (1)*, volume 12710 of *Lecture Notes in Computer Science*, pages 512–542. Springer, 2021.

[DS16] Léo Ducas and Damien Stehlé. Sanitization of FHE ciphertexts. In *EUROCRYPT (1)*, volume 9665 of *Lecture Notes in Computer Science*, pages 294–310. Springer, 2016.

[Dua] Duality Technologies. Duality Labs OpenFHE experiments for encrypted network measurement/control and secure data distribution with proxy re-encryption.

[FKKP19] Georg Fuchsbauer, Chethan Kamath, Karen Klein, and Krzysztof Pietrzak. Adaptively secure proxy re-encryption. In *Public Key Cryptography (2)*, volume 11443 of *Lecture Notes in Computer Science*, pages 317–346. Springer, 2019.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, page 144, 2012.

[Gen09] Craig Gentry. *A fully homomorphic encryption scheme.* PhD thesis, Stanford University, USA, 2009.

[GHS12a] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 465–482. Springer, 2012.

[GHS12b] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 850–867. Springer, 2012.

[GKS24] Kamil Doruk Gür, Jonathan Katz, and Tjerand Silde. Two-round threshold lattice-based signatures from threshold homomorphic encryption. In Markku-Juhani O. Saarinen and Daniel Smith-Tone, editors, *Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Oxford, UK, June 12-14, 2024, Proceedings, Part II*, volume 14772 of *Lecture Notes in Computer Science*, pages 266–300. Springer, 2024.

[GLSW21] Sivanarayana Gaddam, Atul Luykx, Rohit Sinha, and Gaven J. Watson. Reducing HSM reliance in payments through proxy re-encryption. In *USENIX Security Symposium*, pages 4061–4078. USENIX Association, 2021.

[GMP19] Nicholas Genise, Daniele Micciancio, and Yuriy Polyakov. Building an efficient lattice gadget toolkit: Subgaussian sampling and more. In *EUROCRYPT (2)*, volume 11477 of *Lecture Notes in Computer Science*, pages 655–684. Springer, 2019.

[Goo] Google.com. grpc: Google remote procedure call framework.

[GPR+19] Kamil Doruk Gür, Yuriy Polyakov, Kurt Rohloff, Gerard W. Ryan, Hadi Sajjadpour, and Erkay Savas. Practical applications of improved gaussian sampling for trapdoor lattices. *IEEE Trans. Computers*, 68(4):570–584, 2019.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206. ACM, 2008.

[HM17] Gottfried Herold and Alexander May. LP solutions of vectorial integer subset sums - cryptanalysis of galbraith's binary matrix LWE. In *Public Key Cryptography (1)*, volume 10174 of *Lecture Notes in Computer Science*, pages 3–15. Springer, 2017.

[HO17] James Howe and Máire O'Neill. GLITCH: A discrete gaussian testing suite for lattice-based cryptography. In Pierangela Samarati, Mohammad S. Obaidat, and Enrique Cabello, editors, *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017) - Volume 4: SECRYPT, Madrid, Spain, July 24-26, 2017*, pages 413–419. SciTePress, 2017.

[HPS19] Shai Halevi, Yuriy Polyakov, and Victor Shoup. An improved rns variant of the bfv homomorphic encryption scheme. In Mitsuru Matsui, editor, *CT-RSA 2019*, pages 83–105, Cham, 2019. Springer International Publishing.

[ID03] Anca-Andreea Ivan and Yevgeniy Dodis. Proxy cryptography revisited. In *NDSS*. The Internet Society, 2003.

[Ins] USC Information Sciences Institute. Raven: Ry's apparatus for virtual encodable networks.

[KPZ21] Andrey Kim, Yuriy Polyakov, and Vincent Zucca. 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*, page 608–639, Berlin, Heidelberg, 2021. Springer-Verlag.

[KS23] Kamil Kluczniak and Giacomo Santato. On circuit private, multikey and threshold approximate homomorphic encryption. *IACR Cryptol. ePrint Arch.*, page 301, 2023.

[Lee20] Jasper Lee. Lecture notes in sublinear algorithms for big data, 2020.

[LM21] Baiyu Li and Daniele Micciancio. On the security of homomorphic encryption on approximate numbers. In *EUROCRYPT (1)*, volume 12696 of *Lecture Notes in Computer Science*, pages 648–677. Springer, 2021.

[LMSS22] Baiyu Li, Daniele Micciancio, Mark Schultz, and Jessica Sorrell. Securing approximate homomorphic encryption using differential privacy. In *CRYPTO (1)*, volume 13507 of *Lecture Notes in Computer Science*, pages 560–589. Springer, 2022.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010.

[MBH23] Christian Mouchet, Elliott Bertrand, and Jean-Pierre Hubaux. An efficient threshold access-structure for rlwe-based multiparty homomorphic encryption. *J. Cryptol.*, 36(2):10, 2023.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer, 2012.

[MW17] Daniele Micciancio and Michael Walter. Gaussian sampling over the integers: Efficient, generic, constant-time. In *CRYPTO (2)*, volume 10402 of *Lecture Notes in Computer Science*, pages 455–485. Springer, 2017.

[MW18] Daniele Micciancio and Michael Walter. On the bit security of cryptographic primitives. In *EUROCRYPT (1)*, volume 10820 of *Lecture Notes in Computer Science*, pages 3–28. Springer, 2018.

[ON20] Ruxandra F. Olimid and Gianfranco Nencioni. 5g network slicing: A security overview. *IEEE Access*, 8:99999–100009, 2020.

[PRSV17] Yuriy Polyakov, Kurt Rohloff, Gyana Sahu, and Vinod Vaikuntanathan. Fast proxy re-encryption for publish/subscribe systems. *ACM Trans. Priv. Secur.*, 20(4):14:1–14:31, 2017.

[PS25] Alain Passelègue and Damien Stehlé. Low communication threshold fully homomorphic encryption. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology – ASIACRYPT 2024*, pages 297–329, Singapore, 2025. Springer Nature Singapore.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93. ACM, 2005.

[SDDR21] Willy Susilo, Priyanka Dutta, Dung Hoang Duong, and Partha Sarathi Roy. Lattice-based hra-secure attribute-based proxy re-encryption in standard model. In *ESORICS (2)*, volume 12973 of *Lecture Notes in Computer Science*, pages 169–191. Springer, 2021.

[ZLH24] Yunxiao Zhou, Shengli Liu, and Shuai Han. Multi-hop fine-grained proxy re-encryption. In Qiang Tang and Vanessa Teague, editors, *Public-Key Cryptography – PKC 2024*, pages 161–192, Cham, 2024. Springer Nature Switzerland.

[ZLHZ23] Yunxiao Zhou, Shengli Liu, Shuai Han, and Haibin Zhang. Fine-grained proxy re-encryption: Definitions and constructions from LWE. In *ASIACRYPT (6)*, volume 14443 of *Lecture Notes in Computer Science*, pages 199–231. Springer, 2023.

[ZPW+15] Hannes Zach, Philip Peinsold, Johannes Winter, Peter Danner, and Jakob Hatzl. Using proxy re-encryption for secure data management in an ambient assisted living application. In Detlef Hühnlein, Heiko Roßnagel, Raik Kuhlisch, and Jan Ziesing, editors, *Open Identity Summit 2015*. GI, 2015.

Appendices

A Further Details on BGV Scheme Optimizations and Key Switching

When working with the Residue Number System (RNS) (called the "double-CRT" optimization elsewhere [GHS12a]), the largest modulus Q is chosen as a product of NTT-friendly machine-sized primes $Q = Q_L = \prod q_i,\ i = 1, \ldots, L$ where each $Q_i = \prod q_j,\ j = 1, \ldots, i$.

A.1 Key Switching

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 $\mathbf{a}_i \in \{0,1\}^k \subset R_{Q_l}^k$ is the binary decomposition of a_i: $\sum_j a_i[j]2^j = a_i$. Let $\mathbf{2} = (1, 2, \ldots, 2^{k-1})$ denote the power of two vector in $R_{Q_l}^k$, 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 $\boldsymbol{\omega} = (1, \omega, \omega^2, \ldots, \omega^{\mathsf{dnum}-1})$, 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.

BV key switching. The BV key-switching [BV11a] 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 [KPZ21] 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 $\|e\|_\infty + \frac{\mathsf{dnum}'\tilde{Q}B_{err}\delta_R}{2}$ if we use the decomposition in Equations (1)-(2) and $\tilde{Q} = \max_i \tilde{Q}_i$. Further, the noise magnitude is at most $\|e\|_\infty + \frac{\lceil \log_\omega(\tilde{Q})\rceil \mathsf{dnum}'\omega B_{err}\delta_R}{2}$ 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 $\frac{\lceil \log_\omega(Q)\rceil \omega B_{err}\delta_R}{2}$.

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 [GHS12b] technique and the original digit-decomposition-based (BV) [BV11b] 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 $\frac{\zeta_{num}\mathsf{dnum}'\delta_R B_{err}}{2} + \zeta_{num}\frac{1+\delta_R}{2}$, 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 $\frac{\zeta_{num}\mathsf{dnum}'\omega\delta_R B_{err}}{2P} + \zeta_{num}\frac{1+\delta_R}{2}$ if a base-ω is used in addition to the RNS decomposition in Equations (1)-(2). See the appendix of [KPZ21] for a detailed analysis.

B Rerandomization (Public Key Encryption) Adds Too Little Noise

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 since it does not add enough noise during the re-randomization process. This method was used in the work of Davidson et al. [DDLM19] 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 [HM17]. 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 [DDLM19]. 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 [DDLM19] 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 [DDLM19].

Attack. The main idea of our attack is to simply re-encrypt many ciphertexts from the same honest party. The error will be structured in a way which

Algorithm 8 Re-Encryption Without Noise Flooding [DDLM19].

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** $c^* + c'$

enables the receiver to recover the original ciphertext's error. In turn, we can recover the secret key just as Li and Micciancio [LM21] and Cohen [Coh19] 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: $(\mathbf{Cx} + \mathbf{e}, \mathbf{C})$ 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.

[11] This can be done with the Central Limit Theorem or concentration bounds, like Bernstein and Hoeffding concentration inequalities [Lee20]. See Lecture 3 of `https://cs.brown.edu/courses/csci1951-w/` for details on generic mean-finding algorithms.

C Circuit privacy technique of [dCJV21]

De Castro et al. [dCJV21] 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 [dCJV21] 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 [dCJV21]. 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 [dCJV21]. 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 $p \cdot as$ 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 [dCJV21]

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.

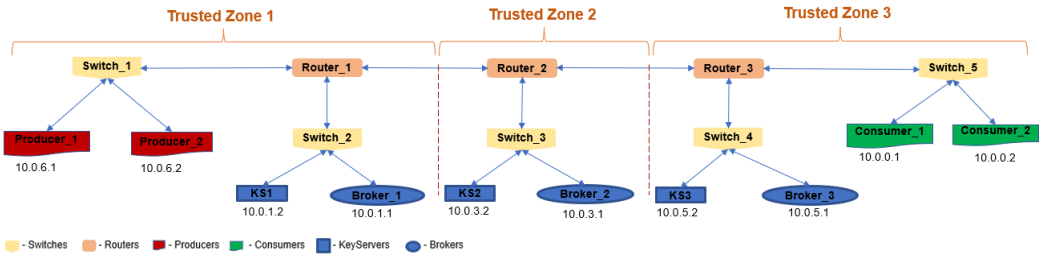

Fig. 2: Network topology used in RAVEN for PRE with three trust zones. Each service - key server, producer, broker and consumer are run on virtual machines. Each trust zone is separated by a router.

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 [dCJV21] 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 [dCJV21] 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.

D More Details on Simulated Secure Data Distribution System

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 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.