Two-Limb CRT Ring-LWE Encryption with Exact Decryption and Public Re-randomization

Damir Vodenicarevic[1*], Andrei Fleiser[1], Pierre Seznec[1], Karen Mayen Naranjo[1], Lucas Foucher[1], Léo Besançon[1], Thybault Alabarbe[1], Jean-François Morcillo[1], Benjamin Reynes[1], Lilian Urvoy[1]

[1*]Massa Labs, Paris, France.

*Corresponding author(s). E-mail(s): dv@massa.net;
Contributing authors: af2@massa.net; ps@massa.net; km@massa.net; lf2@massa.net; lb@massa.net; ta@massa.net; jfm@massa.net; br@massa.net; lu@massa.net;

Abstract

Anonymity infrastructures such as mix networks, anonymous storage, and privacy-preserving replication rely on *public re-randomization*: any party holding only public information can transform a ciphertext into a fresh-looking encryption of the same plaintext, hiding the linkage between the two. Classical ElGamal-based solutions are broken by quantum adversaries, while existing lattice-based alternatives carry very large ciphertexts with unanalyzed noise growth, rely on heavyweight homomorphic-encryption stacks with approximate (rounded) decryption, or lack a precise analysis of how many re-randomizations are safe. We address this gap with a practical Ring Learning with Errors (Ring-LWE) public-key encryption scheme supporting public re-randomization without ciphertext growth. Our construction is Lyubashevsky–Peikert–Regev / Fan–Vercauteren (LPR/BFV)-style encryption over $\boldsymbol{R = \mathbb{Z}[x]/(x^n + 1)}$ with $\boldsymbol{n = 4096}$, engineered around a *two-limb Chinese Remainder Theorem (CRT) modulus* $\boldsymbol{q = t \cdot q_2}$ with 32-bit primes. Embedding plaintext as $\boldsymbol{\Delta M = q_2 M}$ makes the message vanish modulo $\boldsymbol{q_2}$, so the $\boldsymbol{q_2}$-limb carries *only* the decryption noise, enabling *exact* message recovery without rounding. We prove correctness with explicit decryption-failure bounds that remain valid under repeated re-randomization, via an aggregation lemma showing that arbitrarily many re-randomizations affect decryption only through a single aggregated randomness triple. We also prove that two-limb ciphertexts are pseudorandom (indistinguishable from uniform, IND$) under Decision Ring-LWE over the *combined* modulus $\boldsymbol{q = tq_2}$; security against chosen-plaintext attack (IND-CPA) and re-randomization unlinkability follow. A constant-time Rust implementation encrypts in 0.80 ms, re-randomizes in 0.51 ms, and decrypts in 0.21 ms per 64 KiB ciphertext carrying 15.5 KiB of payload on a fixed-frequency 3.8 GHz CPU—on par with a modulus-matched Microsoft SEAL baseline—and passes timing-leakage tests. Empirical noise simulations validate the analysis.

Keywords: Ring-LWE, Public re-randomization, Post-quantum encryption, CRT modulus splitting, Noise flooding, Unlinkability

1 Introduction

Research area and problem domain. Anonymous communication and privacy-preserving storage systems must hide not only the *content* of

messages but also the *linkage* between observations of the same message at different times or places. Since Chaum's mix networks, the standard cryptographic tool for breaking such linkage has been *public re-randomization*: the ability to transform a ciphertext into another ciphertext of the *same* plaintext, using only public information, in a way that hides the relationship between the two ciphertexts. Applications include:

- **Mix networks and anonymous routing:** each relay re-randomizes forwarded ciphertexts to prevent traffic correlation [1–3].
- **Anonymous storage:** a proxy periodically shuffles stored ciphertexts to defeat access-pattern analysis.
- **Privacy-preserving synchronization:** encrypted blocks are refreshed during replication so that replicas are unlinkable.

A representative deployment scenario, which motivated this work, is a decentralized storage network in which encrypted data blocks are replicated among untrusted nodes: each node periodically re-randomizes the blocks it forwards so that an observer monitoring several nodes cannot tell which blocks are copies of one another, while the data owner retains the ability to decrypt exactly.

Need for this study. Classically, ElGamal-based universal re-encryption [1] admits unlimited re-randomization with statistical unlinkability, but its security collapses against quantum adversaries running Shor's algorithm. With the NIST post-quantum standards now finalized [4] and post-quantum privacy primitives an active research area [3, 5], a quantum-safe replacement is needed. Ring Learning with Errors (Ring-LWE) encryption naturally supports re-randomization by adding an encryption of zero, and this idea underpins recent lattice-based mix networks [2, 3, 6] and proxy re-encryption schemes [7]. However, practical deployments face three concerns: (i) each re-randomization adds noise, so the number of safe re-randomizations is bounded and must be analyzed precisely; (ii) efficient implementations typically rely on multi-prime Residue Number System (RNS) homomorphic-encryption stacks (SEAL [8], OpenFHE [9]) whose generality brings approximate decryption with rounding, large code bases, and parameter sets oversized for the task; and (iii) existing lattice-based universal re-encryption proposals [10] use large matrix ciphertexts whose noise compounds multiplicatively at every hop, and lack concrete parameters or implementations. The problem we address is therefore: *design a minimal, post-quantum, size-preserving publicly re-randomizable encryption scheme with exact decryption, a precise re-randomization budget, and an implementation fast and hardened enough for production use.*

Contributions.

1. **Exact Chinese Remainder Theorem (CRT) decryption from a "noise limb."** With $q = t \cdot q_2$ and $\Delta M = q_2 M$, the message is 0 mod q_2 and the q_2-limb reveals the exact signed noise. In contrast, state-of-the-art RNS–BFV decryption [8, 11] performs approximate scaling, rounding, and base extension; to the best of our knowledge the "message-vanishing limb" formulation and its use for exact noise extraction are new (see Section 10 for a detailed comparison).
2. **A minimal re-randomizable Ring-LWE public-key encryption scheme with a quantified budget.** Re-randomization is $ct' = ct + \mathsf{Enc}(pk, 0)$ and preserves ciphertext size, unlike lattice universal re-encryption [10] (multiplicative noise growth per hop) and unlike FHE-style ciphertext sanitization [12] (bootstrapping cost). An exact noise-aggregation lemma yields a closed-form bound $k_{\max} \approx 1.5 \times 10^{10}$ on the number of safe re-randomizations for our parameters—a quantity left unanalyzed in prior re-randomizable lattice constructions.
3. **Clean security: IND\$.** Ciphertexts are pseudorandom under the Decision Ring-LWE (DRLWE) assumption. IND-CPA and a game-based notion of pairwise unlinkability (Definition 5.5) are corollaries (resolving the common pitfall that IND-CPA does not imply uniformity).
4. **32-bit implementation path, hardened and benchmarked.** Both CRT primes are Number Theoretic Transform (NTT)-friendly and fit in `uint32`. We provide a constant-time Rust implementation (∼1300 lines of code, zero `unsafe`) with cumulative-distribution-table (CDT) and inverse cumulative-distribution-function (inverse-CDF) Gaussian samplers, Test Vector Leakage Assessment (TVLA)-style leakage testing, fixed-frequency benchmarks on

x86-64 (with a same-machine Microsoft SEAL comparison), `aarch64` (mobile) build support, and a Python noise simulator.

Paper organization. Section 2 recalls rings, the DRLWE assumption, and the CRT isomorphism. Section 3 presents the parameters and the four algorithms of the scheme, with a toy example. Section 4 proves exact decryption and derives fixed-key decryption-failure bounds and the re-randomization budget $k_{\max}$. Section 5 formalizes IND\$, IND-CPA, and unlinkability. Section 6 analyzes optional noise flooding. Section 7 discusses composition with Authenticated Encryption with Associated Data (AEAD) and the threat model for malicious re-randomizers. Section 8 assesses concrete security. Section 9 describes the implementation, benchmarks, side-channel evaluation, and noise simulations. Section 10 reviews related work, and Section 11 concludes.

2 Preliminaries

2.1 Rings

Let $n = 4096$ and $R = \mathbb{Z}[x]/(x^n + 1)$. For $m \geq 2$, define $R_m := R/mR \cong \mathbb{Z}_m[x]/(x^n + 1)$. All ring multiplications are negacyclic convolutions, computable in $O(n \log n)$ via the NTT when m is prime with $m \equiv 1 \pmod{2n}$ [13].

2.2 Decision Ring-LWE

Definition 2.1 (DRLWE) Fix q, a secret distribution χ_s over R_q, and an error distribution χ_e over R_q. The DRLWE problem is to distinguish $(a,\ as + e)$ from $(a,\ u)$ where $a \leftarrow U(R_q)$, $s \leftarrow \chi_s$, $e \leftarrow \chi_e$, $u \leftarrow U(R_q)$.

Lemma 2.2 (Discrete Gaussians are sub-Gaussian) *If $X \leftarrow D_{\mathbb{Z},\sigma}$ is a centered discrete Gaussian, then X is σ-sub-Gaussian. Moreover, if $X_1, \ldots, X_m$ are independent σ-sub-Gaussians, then $\sum_{i=1}^{m} X_i$ is $\sigma\sqrt{m}$-sub-Gaussian.*

Proof The centered discrete Gaussian over $\mathbb{Z}$ satisfies the mgf bound $\mathbb{E}[e^{\lambda X}] \leq e^{\lambda^2\sigma^2/2}$ for every $\sigma > 0$ ([14], Lemma 2.8). The sum statement follows by independence and multiplying mgfs. □

Assumption used in this paper. We assume DRLWE hardness in the *combined* ring R_q for $q = tq_2$ with secrets and errors sampled as integer polynomials with coefficients from a discrete Gaussian (parameter σ) and reduced modulo q. (Implementations operate in CRT limbs, but an adversary seeing both limbs is equivalent—via CRT—to seeing a single element of R_q.) This is the DRLWE problem in *normal form*: the secret is drawn from the same distribution as the error. Normal-form (D)RLWE is polynomially equivalent to the variant with a uniform secret [13, 15], and sampling both keygen and encryption randomness from the *same* discrete Gaussian $D_{\mathbb{Z}^n,\sigma}$ is exactly the setting in which we invoke the assumption (Section 5); no auxiliary simulatability condition is needed. Our proofs are otherwise "black-box": any χ_s, χ_e for which DRLWE in R_q is assumed hard suffices.

2.3 CRT Isomorphism

Lemma 2.3 *If $\gcd(t, q_2) = 1$, then coefficientwise CRT gives $R_{tq_2} \cong R_t \times R_{q_2}$. Sampling uniformly in R_{tq_2} is equivalent to sampling independently and uniformly in R_t and R_{q_2}.*

Remark 2.4 (CRT view of keys/ciphertexts) Sampling $a_t \leftarrow U(R_t)$ and $a_{q_2} \leftarrow U(R_{q_2})$ induces a unique $a \leftarrow U(R_q)$ such that $a \bmod t = a_t$ and $a \bmod q_2 = a_{q_2}$ (Lemma 2.3). Likewise, a two-limb ciphertext $(c_{0,t}, c_{0,q_2}, c_{1,t}, c_{1,q_2})$ corresponds bijectively to a single pair $(c_0, c_1) \in R_q^2$.

3 Construction

3.1 Parameters

We conceptually work modulo $q = t \cdot q_2$ but implement in two CRT limbs. Our instantiation is summarized in Table 1, which fixes every constant used in the remainder of the paper. The rationale behind each value is as follows. The ring dimension $n = 4096$ is the smallest power of two for which a 63-bit modulus leaves a comfortable margin below the 128-bit-security ceiling of the Homomorphic Encryption Standard ($\log_2 q \leq 109$ for $n = 4096$ [16]; see Section 8) while still offering a large per-ciphertext payload (15.5 KiB). The plaintext prime $t \approx 2^{31}$ maximizes payload per coefficient (31 bits) subject to fitting comfortably in a `uint32` word, and the noise prime $q_2 \approx 2^{32}$ maximizes the noise

budget $q_2/2$ (hence the re-randomization budget $k_{\max}$, Section 4) subject to the same word-size constraint. Both primes satisfy $p \equiv 1 \pmod{2n}$ so that negacyclic NTTs of length n exist modulo each prime, and $\gcd(t, q_2) = 1$ enables the CRT splitting (Lemma 2.3). Because $q = t \cdot q_2$ exactly, the scaling factor $\Delta = q/t = q_2$ is an *integer*, which is the keystone of exact decryption (Theorem 4.3). The Gaussian width $\sigma = 3.2$ is the standard choice of the Homomorphic Encryption Standard [16] and of mainstream libraries [8, 9], making our security assessment directly comparable to established parameter tables. Δ_t and Δ_t^{-1} are derived constants used by encryption and decryption, respectively.

3.2 Algorithms

Algorithms 1–4 specify the scheme per message polynomial $M \in R_t$. Multi-slot packing (Section 9) is orthogonal.

Algorithm 1 KeyGen(1^λ)

Require: Security parameter 1^λ (instantiated by the fixed parameter set of Table 1)
Ensure: Public key $pk \in (R_t \times R_{q_2})^2$; secret key $sk \in R_t \times R_{q_2}$

Sample $a_t \leftarrow U(R_t)$, $a_{q_2} \leftarrow U(R_{q_2})$
Sample $s, e \leftarrow D_{\mathbb{Z}^n,\sigma}$
$b_t \leftarrow a_t s + e \bmod t$
$b_{q_2} \leftarrow a_{q_2} s + e \bmod q_2$
$pk \leftarrow ((a_t, a_{q_2}), (b_t, b_{q_2}))$
$sk \leftarrow (s_t, s_{q_2}) = (s \bmod t,\ s \bmod q_2)$
return (pk, sk)

Algorithm 2 Enc(pk, M)

Require: Public key $pk = ((a_t, a_{q_2}), (b_t, b_{q_2}))$; message $M \in R_t$ with coefficients in $[0, 2^{31})$
Ensure: Ciphertext $ct \in (R_t \times R_{q_2})^2$

Sample $r, e_1, e_2 \leftarrow D_{\mathbb{Z}^n,\sigma}$
$c_{1,t} \leftarrow a_t r + e_1 \bmod t$
$c_{1,q_2} \leftarrow a_{q_2} r + e_1 \bmod q_2$
$c_{0,t} \leftarrow b_t r + e_2 + \Delta_t M \bmod t$
$c_{0,q_2} \leftarrow b_{q_2} r + e_2 \bmod q_2 \triangleright\ q_2 M \equiv 0 \pmod{q_2}$
return $ct = (c_{0,t}, c_{0,q_2}, c_{1,t}, c_{1,q_2})$

Algorithm 3 ReRand(pk, ct) — public

Require: Public key $pk = ((a_t, a_{q_2}), (b_t, b_{q_2}))$; ciphertext $ct = (c_{0,t}, c_{0,q_2}, c_{1,t}, c_{1,q_2})$
Ensure: Ciphertext ct' of the same plaintext, same size

Sample $r', e_1', e_2' \leftarrow D_{\mathbb{Z}^n,\sigma}$
$c'_{0,t} \leftarrow c_{0,t} + b_t r' + e_2' \bmod t$
$c'_{0,q_2} \leftarrow c_{0,q_2} + b_{q_2} r' + e_2' \bmod q_2$
$c'_{1,t} \leftarrow c_{1,t} + a_t r' + e_1' \bmod t$
$c'_{1,q_2} \leftarrow c_{1,q_2} + a_{q_2} r' + e_1' \bmod q_2$
return $ct' = (c'_{0,t}, c'_{0,q_2}, c'_{1,t}, c'_{1,q_2})$

Algorithm 4 Dec(sk, ct) — noise-limb trick

Require: Secret key $sk = (s_t, s_{q_2})$; ciphertext $ct = (c_{0,t}, c_{0,q_2}, c_{1,t}, c_{1,q_2})$
Ensure: Message $M \in R_t$ (exact, provided $\|\nu\|_\infty < q_2/2$; Theorem 4.3)

$v_t \leftarrow c_{0,t} - c_{1,t} \cdot s_t \bmod t$
$v_{q_2} \leftarrow c_{0,q_2} - c_{1,q_2} \cdot s_{q_2} \bmod q_2$
for each coeff. $v_i \in [0, q_2)$ of v_{q_2} (center-lift) **do**
    $\nu_i \leftarrow v_i - q_2 \cdot [\, v_i > q_2/2 \,] \in (-q_2/2,\ q_2/2)$
end for
return $M \leftarrow (v_t - (\nu \bmod t)) \cdot \Delta_t^{-1} \bmod t$

Remark 3.1 (Center-lift, explicitly) The center-lift loop computes the unique signed representative of v_i modulo q_2 in $(-q_2/2, q_2/2)$: here $[\cdot]$ is 1 if the condition holds and 0 otherwise (equivalently, $\nu_i \leftarrow v_i$ if $v_i < q_2/2$ and $\nu_i \leftarrow v_i - q_2$ otherwise; since q_2 is odd, $v_i = q_2/2$ cannot occur). In the final step, $\nu \bmod t$ denotes reduction of the signed value into $[0, t)$; since $|\nu_i| < q_2/2 < t$ in our parameter set, this is the single conditional addition $\nu_i \bmod t = \nu_i + t \cdot [\nu_i < 0]$. A constant-time implementation derives the selections from comparison masks, e.g. `nu = v - (q2 & -(v > q2/2))` in two's-complement arithmetic, with no data-dependent branch (Section 9).

Remark 3.2 (What is "exact"?) No CRT recombination modulo $q = tq_2$ is performed and no rounding is used. All operations are modulo t or modulo q_2; the 63-bit product modulus never appears on the hot path.

Remark 3.3 (Contrast with BFV rounding) Standard BFV-style decryption recovers $\Delta M + \nu$ modulo a large modulus and then uses rounding (and often CRT recombination) to map back to R_t. Here, correctness is still governed by the classical condition $\|\nu\|_\infty <$

Table 1 Scheme parameters

Parameter	Value	Notes
n	4096	Ring dimension
t	2,147,565,569	Prime $\approx 2^{31}$, $t \equiv 1 \pmod{2n}$
q_2	4,294,828,033	Prime $\approx 2^{32}$, $q_2 \equiv 1 \pmod{2n}$
Δ	$q/t = q_2$	Exact scaling factor
Δ_t	$q_2 \bmod t = 2{,}147{,}262{,}464$	Embedding in R_t
Δ_t^{-1}	1,987,184,532	Inverse of $\Delta_t \bmod t$
σ	3.2	Discrete Gaussian width

$q_2/2$, but ν is *extracted exactly* from the q_2-limb and subtracted, eliminating rounding logic.

3.3 Toy Example

To illustrate concretely (ring dimension $n = 1$, integers only): $t = 7$, $q_2 = 11$, $\Delta_t = 11 \bmod 7 = 4$, $\Delta_t^{-1} = 2$ (since $4 \cdot 2 \equiv 1 \pmod 7$). Secret $s = 2$, keygen error $e = 1$, $b = 3 \cdot 2 + 1 = 7$. Encrypt $M = 5$: sample $r = 1$, $e_1 = e_2 = 0$. $c_{0,t} = 0 + 0 + 4 \cdot 5 = 20 \equiv 6 \pmod 7$; $c_{0,q_2} = 7 + 0 + 0 = 7 \pmod{11}$; $c_{1,t} = 3 \pmod 7$; $c_{1,q_2} = 3 \pmod{11}$. Decrypt: $v_t = 6 - 3 \cdot 2 = 0 \pmod 7$; $v_{q_2} = 7 - 3 \cdot 2 = 1 \pmod{11}$; $\nu = 1$; $M = (0 - 1) \cdot 2 = -2 \equiv 5 \pmod 7$. ✓

4 Correctness

4.1 Noise expression and aggregation (no hidden dependence)

For one limb with $b = as + e$ we have

$$c_0 - c_1 s = er + e_2 - e_1 s + \Delta m, \tag{1}$$

where $\Delta m = 0$ in the q_2-limb. Define the signed noise polynomial for a single encryption as

$$\nu := er + e_2 - e_1 s. \tag{2}$$

Re-randomization adds an independent encryption of zero $(r', e_1', e_2' \leftarrow D_{\mathbb{Z}^n,\sigma})$. Optionally, the encryptor may also add a one-time "flood" encryption of zero with width σ_f (Section 6).

Lemma 4.1 (Exact aggregation of repeated re-randomization) *Fix a keypair (hence fixed s, e). Let $(r_0, e_{1,0}, e_{2,0})$ denote the fresh encryption randomness, let $(r_f, e_{1,f}, e_{2,f})$ denote the optional flooding triple (or 0 if disabled), and let $(r_i, e_{1,i}, e_{2,i})$ for $i = 1, \dots, k$ denote the k independent re-randomization triples. Define the aggregated polynomials*

$$r^{(k)} := r_0 + r_f + \textstyle\sum_{i=1}^{k} r_i,$$
$$e_1^{(k)} := e_{1,0} + e_{1,f} + \textstyle\sum_{i=1}^{k} e_{1,i},$$
$$e_2^{(k)} := e_{2,0} + e_{2,f} + \textstyle\sum_{i=1}^{k} e_{2,i}.$$

Then the q_2-limb decryption value after k re-randomizations satisfies

$$v_{q_2} \equiv \nu^{(k)} := e\, r^{(k)} + e_2^{(k)} - e_1^{(k)} s \pmod{q_2}. \tag{3}$$

In particular, the full sequence of k re-randomizations affects decryption only through the aggregate triple $(r^{(k)}, e_1^{(k)}, e_2^{(k)})$; there is no additional cross-step dependence to account for.

Proof For each limb, encryption and re-randomization update the ciphertext by adding terms linear in (r, e_1, e_2). Plugging into $c_0 - c_1 s$ and using distributivity gives (3) with $(r^{(k)}, e_1^{(k)}, e_2^{(k)})$ as defined above. □

Remark 4.2 (Effective width) Each coefficient of $r^{(k)}$, $e_1^{(k)}$, and $e_2^{(k)}$ is a sum of independent centered discrete Gaussians. Hence each coefficient is sub-Gaussian (Lemma 2.2) with parameter

$$\sigma_k := \sigma \sqrt{k + 1 + \kappa_f}, \quad \kappa_f := \sigma_f^2 / \sigma^2, \tag{4}$$

where $\sigma_f = 0$ when flooding is disabled.

Theorem 4.3 (Exact decryption) *Let $\nu^{(k)}$ be the centered lift of (3) in $(-q_2/2, q_2/2)^n$ (the interval is open since q_2 is odd). If $|\nu_i^{(k)}| < q_2/2$ for every coefficient, Algorithm 4 outputs the exact $M \in R_t$. ("Exact" means no rounding error; correctness remains conditional on the noise bound.)*

Proof Center-lift recovers $\nu^{(k)}$ from v_{q_2} (unique representative in $(-q_2/2, q_2/2)$). Then $v_t - \nu^{(k)} \equiv \Delta_t M \pmod t$, and Δ_t^{-1} recovers M. □

4.2 Good keys and fixed-key failure bound

In practice the key (s, e) is fixed for the lifetime of the system. We condition on a high-probability event over keygen, then bound the failure probability over *only* encryption/rerand randomness.

Lemma 4.4 (Good keys (ℓ_2)) *Say that (s, e) is* good *if $\|s\|_2^2 \le 2n\sigma^2$ and $\|e\|_2^2 \le 2n\sigma^2$. By standard chi-squared concentration (each squared norm is a sum of n independent sub-exponential variables with mean σ^2), $\Pr[(s, e)$ is not good$] \le 2\exp(-n/4)$. For $n = 4096$: $\Pr[$not good$] \le 2\exp(-1024) \approx 0$.*

Lemma 4.5 (Fixed-key tail bound) *Fix a good key (s, e) with $\|e\|_2^2 \le E_0$ and $\|s\|_2^2 \le S_0$. Let $\eta^{(k)}$ be any single coefficient of $\nu^{(k)}$ (Lemma 4.1). Then for all $t > 0$:*

$$\Pr\left[|\eta^{(k)}| \ge t \;\middle|\; s, e\right] \le 2\exp\left(-\frac{t^2}{2(E_0 + S_0 + 1)\,\sigma_k^2}\right),$$

where σ_k is as in (4) and the probability is over encryption/rerand randomness only.

Proof Conditioned on a fixed key, each e_j and s_j is a *deterministic* value. Each coefficient of $r^{(k)}$, $e_1^{(k)}$, $e_2^{(k)}$ is σ_k-sub-Gaussian (Remark 4.2, Lemma 2.2).

The i-th coefficient of $e\,r^{(k)}$ (negacyclic) is $\sum_{j=0}^{n-1} \pm e_j \cdot r_{i-j}^{(k)}$: a sum of n independent terms, each the product of a *fixed constant* e_j and a σ_k-sub-Gaussian variable. Each term is sub-Gaussian with parameter $|e_j|\sigma_k$. By the sum rule (Lemma 2.2), the sum is sub-Gaussian with parameter $\sigma_k\sqrt{\sum_j e_j^2} = \sigma_k\|e\|_2$.

The same argument gives $\sigma_k\|s\|_2$ for $e_1^{(k)}s$, and $e_2^{(k)}$ contributes σ_k. The full coefficient $\eta^{(k)}$ is sub-Gaussian with parameter $\sigma_k\sqrt{\|e\|_2^2 + \|s\|_2^2 + 1} \le \sigma_k\sqrt{E_0 + S_0 + 1}$. The stated tail follows from $\Pr[|X| \ge t] \le 2e^{-t^2/(2\tau^2)}$. □

Theorem 4.6 (Fixed-key batch failure bound) *Fix a re-randomization count $k \ge 0$. Let B be the number of ciphertexts in a batch and $p_{\text{batch}} = 2^{-106}$. With probability $\ge 1 - 2\exp(-n/4)$ over keygen (Lemma 4.4), the following holds for the* fixed *key:*

$$\Pr\left[\exists\ ct \text{ in batch: } \|\nu^{(k)}\|_\infty \ge q_2/2 \;\middle|\; s, e\right] \le p_{\text{batch}}$$

whenever $B_{\text{coeff}}(k, \delta_{\text{coeff}}) \le q_2/2$ with $\delta_{\text{coeff}} := p_{\text{batch}}/(Bn)$ and

$$B_{\text{coeff}}(k, \delta) := \sigma_k\sqrt{2(4n\sigma^2 + 1)\ln(2/\delta)}. \tag{5}$$

(Here $E_0 = S_0 = 2n\sigma^2$ from Lemma 4.4.)

Maximum safe re-randomizations.

$$k_{\max} := \max\left\{k \ge 0 : B_{\text{coeff}}\left(k, \tfrac{p_{\text{batch}}}{Bn}\right) \le \tfrac{q_2}{2}\right\}. \tag{6}$$

Remark 4.7 (Numerical values) For $n = 4096$, $\sigma = 3.2$, $B = 1024$, $p_{\text{batch}} = 2^{-106}$ (so $\delta_{\text{coeff}} = 2^{-128}$):

- Without flooding ($\kappa_f = 0$): $k_{\max} \approx 1.5 \times 10^{10}$ (∼476 years at 1/s).
- With flooding ($\kappa_f = 858{,}000{,}000$): $k_{\max} \approx 1.4 \times 10^{10}$ additional (∼448 years at 1/s).

The ℓ_2-based good-key conditioning is much tighter than an ℓ_∞ bound and retains essentially the full noise budget.

Remark 4.8 (Tightness: worst-case vs. typical keys) Theorem 4.6 conditions on the good-key event $\|s\|_2^2, \|e\|_2^2 \le 2n\sigma^2$, whereas a typical key has $\|s\|_2^2 \approx \|e\|_2^2 \approx n\sigma^2$ (the mean; our simulations below confirm concentration within a few percent). The bound of Lemma 4.5 is monotone in $E_0 + S_0$, so it can be instantiated with *any* per-key norms: a deployment that measures its actual key (the owner knows s and e) may use $E_0 = \|e\|_2^2$, $S_0 = \|s\|_2^2$ directly. For a typical key ($E_0 = S_0 = n\sigma^2$) the admissible budget doubles, to $k_{\max} \approx 3.0 \times 10^{10}$ without flooding (the headline B_{coeff} shrinks by $1/\sqrt{2}$ and $k_{\max}$ scales as B_{coeff}^{-2}). We keep the conservative $2n\sigma^2$ conditioning in the headline theorem because it holds for *all but a $2e^{-n/4}$ fraction* of keys without any per-key measurement. Appendix A (Lemma A.4) additionally gives a sub-exponential *average-over-keys* bound. We emphasize that this choice has no bearing on security: the correctness bound does not enter the IND$ reduction (Section 5), so tightening it cannot weaken or break the security proof; it only refines the operational budget $k_{\max}$.

Remark 4.9 (On the union bound over Bn coefficients) Theorem 4.6 applies a union bound over the Bn coefficient events. The union bound requires no independence assumption and is therefore valid despite the correlations among coefficients (they share s, e,

and the aggregated randomness). Its cost is moreover modest and quantifiable: moving from the per-batch target $p_{\text{batch}} = 2^{-106}$ to the per-coefficient target $\delta_{\text{coeff}} = p_{\text{batch}}/(Bn) = 2^{-128}$ inflates B_{coeff} by only $\sqrt{\ln(2/\delta_{\text{coeff}})/\ln(2/p_{\text{batch}})} = \sqrt{129/107} \approx 1.10$, i.e. $\leq 10\%$ in the noise bound and $\approx 21\%$ in $k_{\max}$. Dependent-variable concentration tools (e.g., Talagrand-type inequalities) could at best recover part of this $\leq 10\%$, at the price of a substantially more involved analysis, so we retain the union bound. The empirically observed headroom (Figures 1 and 2) is dominated instead by the ℓ_2 conditioning (Remark 4.8) and by sub-Gaussian tail conservatism at moderate deviations, as quantified in Section 9.

Remark 4.10 (Key rotation and cross-key noise) Lemma 4.1 and Theorem 4.6 are stated for a fixed key because the noise polynomial $\nu^{(k)} = e\,r^{(k)} + e_2^{(k)} - e_1^{(k)}s$ is defined relative to one (s, e). This matches how the scheme is used: a ciphertext created under pk can only be re-randomized with the *same* pk and decrypted with the matching sk, so noise never aggregates across keys. When keys are rotated, the owner decrypts each ciphertext under the old key and re-encrypts it under the new key (the scheme deliberately offers no public key-switching; cf. proxy re-encryption [7]). This re-encryption outputs a *fresh* ciphertext: by Theorem 4.3 decryption is exact, so the plaintext—and hence the new ciphertext distribution—is identical to a first-time encryption, the noise budget resets to $k = 0$, and Theorem 4.6 reapplies verbatim under the new key. The aggregation lemma therefore extends across rotations trivially: the per-key budgets are independent, and the only cross-key requirement is that decryption succeed before each rotation, which is exactly the per-key statement of Theorem 4.6. Operational guidance for *when* to rotate or refresh is given in Section 9.6.

5 Security

5.1 Ciphertext Pseudorandomness (IND$)

Definition 5.1 (IND$) For a fixed message M, the IND$ experiment gives $\mathcal{A}$ a public key and either (i) $\mathsf{Enc}(pk, M)$ or (ii) a uniform element of the ciphertext space. The advantage is the distinguishing gap.

Lemma 5.2 (CRT-lifted ciphertext distribution) *Let (a_t, a_{q_2}), (b_t, b_{q_2}) be generated by Algorithm 1, and $(c_{0,t}, c_{0,q_2}, c_{1,t}, c_{1,q_2})$ by Algorithm 2 for message M. Let $a, b, c_0, c_1 \in R_q$ be the unique CRT lifts (Remark 2.4). Then (a, b) is distributed as $(a, as + e)$ in R_q^2, and (c_0, c_1) is distributed as $(br+e_2+\Delta M,\ ar+e_1)$ in R_q^2.*

Theorem 5.3 (IND$ for two-limb ciphertext) *Assume DRLWE is hard in R_q for $q = tq_2$ under the secret/error distributions used by the scheme. Then the two-limb ciphertext output by $\mathsf{Enc}(pk, M)$ is* IND$ *for every fixed M.*

Proof By Lemma 5.2, the CRT-lifted ciphertext $(c_0, c_1) \in R_q^2$ has the standard LPR encryption form. The textbook LPR hybrid (replace $b = as + e$ by uniform, then replace $ar + e_1$ and $br + e_2$ by uniform via DRLWE with fresh secret r) shows (c_0, c_1) is indistinguishable from uniform in R_q^2.

We stress that the error distributions match the assumption *exactly*, so no simulatability condition is needed: the first hybrid consumes a DRLWE sample $(a, as+e)$ with $s, e \leftarrow D_{\mathbb{Z}^n,\sigma}$, which is precisely the distribution of Algorithm 1; the second hybrid consumes two DRLWE samples $(a, ar+e_1)$, $(b, br+e_2)$ with the *same* secret $r \leftarrow D_{\mathbb{Z}^n,\sigma}$ and errors $e_1, e_2 \leftarrow D_{\mathbb{Z}^n,\sigma}$, which is precisely the distribution of Algorithm 2 (a two-sample DRLWE instance with shared secret, as in the standard LPR public-key reduction [13]). Both invocations use DRLWE in normal form (secret drawn from the error distribution), which is polynomially equivalent to the uniform-secret variant [13, 15]. The optional flooding packet does not affect this argument; see Lemma 6.1.

CRT is a bijection between R_q^2 and $R_t^2 \times R_{q_2}^2$, so any distinguisher seeing two limbs induces a distinguisher for (c_0, c_1) by CRT recombination. □

Corollary 5.4 (IND-CPA) *The scheme is IND-CPA: ciphertexts of M_0 and M_1 are both pseudorandom, hence indistinguishable.*

5.2 Pairwise Unlinkability

We now give the formal game-based definition of re-randomization unlinkability whose absence is a common gap in informal treatments. The adversary chooses the plaintext, sees the original ciphertext, and must decide whether a second ciphertext is a re-randomization of the first or an independent fresh encryption of the same plaintext.

Definition 5.5 (Unlinkability experiment $\mathbf{Exp}_b^{\mathsf{link}}(\mathcal{A},\lambda)$) For $b \in \{0,1\}$, the experiment proceeds as follows:

1. The challenger samples $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and gives pk to $\mathcal{A}$.
2. $\mathcal{A}(pk)$ outputs a message $M \in R_t$.
3. The challenger computes $ct \leftarrow \mathsf{Enc}(pk, M)$ and

$$ct' \leftarrow \begin{cases} \mathsf{ReRand}(pk, ct) & \text{if } b = 0, \\ \mathsf{Enc}(pk, M) & \text{if } b = 1 \end{cases}$$

   (fresh independent randomness in both cases), and gives (ct, ct') to $\mathcal{A}$.
4. $\mathcal{A}$ outputs a bit b', the output of the experiment.

The advantage of $\mathcal{A}$ is

$$\mathsf{Adv}^{\mathsf{link}}(\mathcal{A}) := \big|\Pr[\mathbf{Exp}_0^{\mathsf{link}}(\mathcal{A},\lambda) = 1] - \Pr[\mathbf{Exp}_1^{\mathsf{link}}(\mathcal{A},\lambda) = 1]\big|.$$

The scheme is *(pairwise) unlinkable* if $\mathsf{Adv}^{\mathsf{link}}(\mathcal{A})$ is negligible for every probabilistic polynomial-time $\mathcal{A}$.

Remark 5.6 (Relation to IND-CPA and IND$) Unlinkability is *not* implied by IND-CPA: both branches of Definition 5.5 encrypt the same adversarially chosen M, so message hiding is irrelevant; what must be hidden is the *correlation* between ct and ct'. An IND-CPA scheme whose re-randomization left a recognizable trace (e.g., a noise signature) could still be perfectly IND-CPA yet trivially linkable. Conversely, unlinkability alone does not imply message hiding. Both properties follow from the single stronger statement that ciphertexts (and hence re-randomization packets) are pseudorandom: IND$ ⇒ IND-CPA (Corollary 5.4) and IND$ ⇒ unlinkability (Theorem 5.7). This is why we prove IND$ as the central property. The pairwise game extends to chains of re-randomizations by a standard hybrid argument: distinguishing any two ciphertexts within a chain of ℓ re-randomizations from independent encryptions costs at most ℓ times the pairwise advantage.

Theorem 5.7 (Unlinkability) *Under the DRLWE assumption of Theorem 5.3,* $\mathsf{Adv}^{\mathsf{link}}(\mathcal{A})$ *is negligible.*

Proof Let $\eta \leftarrow \mathsf{Enc}(pk, 0)$. In the b=0 branch, $ct' = ct + \eta$. By Theorem 5.3 (with M=0), η is pseudorandom, so $(ct, ct+\eta) \approx (ct, ct+U)$ for uniform U. Since $ct + U$ is uniform and independent of ct, this equals (ct, U). Replacing U by $\mathsf{Enc}(pk, M)$ via Theorem 5.3 yields the b=1 distribution. □

6 Optional Noise Flooding

The IND$/unlinkability results of Section 5 do not require flooding. We nevertheless describe flooding as an *optional* defense-in-depth mechanism for deployments that wish to make the *noise magnitude* of a fresh ciphertext resemble that of a ciphertext that has already undergone a large number of re-randomizations. The principal use case is to keep the *number of re-randomizations unknown to the secret-key holder*: without flooding, the decryptor (who recovers the exact noise polynomial $\nu^{(k)}$, by Theorem 4.3) could estimate k from $\|\nu^{(k)}\|$, since the noise width grows as $\sigma\sqrt{k+1}$. In a mix network this would let the decryptor infer how many hops a message traversed; in a replicated-storage setting it would reveal how many times a block was refreshed. With flooding, the fresh-ciphertext width $\sigma\sqrt{1+\kappa_f}$ already dominates the contribution of any realistic number of re-randomizations ($k \ll \kappa_f$), so the observed noise magnitude is statistically uninformative about k. The same masking also reduces side-channel-style "rerand-count" leakage through decryption-failure margins.

Mechanism. At encryption time, sample an additional encryption-of-zero packet $(r_f, e_{1,f}, e_{2,f}) \leftarrow D_{\mathbb{Z}^n,\sigma_f}$ and add it to the ciphertext (i.e., add $\mathsf{Enc}(pk, 0)$ generated with width σ_f).

Effect on correctness bounds. Flooding contributes to the aggregated noise exactly as in Lemma 4.1; its only effect is to increase the effective width from σ to σ_k in (4). It is convenient to parameterize flooding by the *virtual re-randomization count* $\kappa_f := \sigma_f^2/\sigma^2$, so that $\sigma_k = \sigma\sqrt{k+1+\kappa_f}$. Consequently, a deployment that expects at most k public re-randomizations should verify correctness by checking $B_{\mathrm{coeff}}(k, p_{\mathrm{batch}}/(Bn)) \le q_2/2$ in Theorem 4.6 with its chosen κ_f.

Example (as used in our evaluation). For the plots in Section 9 we set $\kappa_f = 858{,}000{,}000$ (so $\sigma_f = \sigma\sqrt{\kappa_f} \approx 93{,}733$) and evaluate additional public re-randomizations on top of this baseline. Flooding increases the effective width from $\sigma\sqrt{k+1}$ to $\sigma\sqrt{k+1+\kappa_f}$. The maximum admissible number of *additional* public re-randomizations is therefore $k_{\max}$ as defined in (6) for the chosen κ_f (equivalently, correctness depends only on the total effective packet count $k+1+\kappa_f$).

Security: flooding does not affect the reduction. After flooding, the overall error of a ciphertext is the sum of a width-σ and a width-σ_f term and is *not* exactly a discrete Gaussian. This does not invalidate the IND$ reduction, because the reduction never consumes the flood randomness as a DRLWE error:

Lemma 6.1 (Flooding preserves IND$) *If the base scheme (Algorithms 1–2) is* IND$*, then so is the variant that adds a flood packet* $fl \leftarrow \mathsf{Enc}_{\sigma_f}(pk, 0)$*, for* any *distribution of the flood triple* $(r_f, e_{1,f}, e_{2,f})$.

Proof The flooded ciphertext is $ct + fl$ where ct and fl are independent given pk. Condition on any fixed value of fl: by IND$ of the base scheme, ct is computationally indistinguishable from uniform, and adding the fixed fl is a bijection of the ciphertext space, so $ct + fl$ is also indistinguishable from uniform. Averaging over fl preserves the bound. □

Quantitative closeness of the implemented flood sampler. Correctness (Theorem 4.6) models the flood triple as a discrete Gaussian $D_{\mathbb{Z},\sigma_f}$, while the constant-time implementation samples a *rounded* continuous Gaussian $\lfloor \mathcal{N}(0, \sigma_f^2) \rceil$ (Section 9.7). For $\sigma_f \gg 1$ the two are close in Rényi divergence, and the gap can be bounded explicitly. Writing $\rho(x) = e^{-x^2/(2\sigma_f^2)}$, the mean-value theorem gives $\Pr[\lfloor \mathcal{N} \rceil = x] \propto \rho(\xi)$ for some $\xi \in (x - \frac{1}{2}, x + \frac{1}{2})$, whence the per-sample probability ratio satisfies

$$\left| \ln \frac{\Pr[\lfloor \mathcal{N} \rceil = x]}{\Pr[D_{\mathbb{Z},\sigma_f} = x]} \right| \le \frac{|x| + 1/4}{2\sigma_f^2} + \theta, \qquad (7)$$

where θ accounts for the normalization constants and is $\le 2^{-100}$ for σ_f at our scale (smoothing-parameter regime [17, 18]). Restricting to $|x| \le 13.4\,\sigma_f$ (the excluded tail has mass $< 2^{-128}$ per sample), the right-hand side of (7) is at most $\delta_s := 13.4/(2\sigma_f) \approx 7.2 \times 10^{-5} = 2^{-13.8}$ for $\sigma_f = 93{,}733$. A flood packet consists of $m = 3n = 12{,}288$ independent samples, so the Rényi divergence of order ∞ between the implemented and ideal packet distributions satisfies $\ln R_\infty \le m\,\delta_s \approx 0.88$, i.e. $R_\infty \le 2.42 \le 2^{1.3}$. By Rényi probability preservation [18], any event—in particular a decryption failure—with probability p under the ideal discrete-Gaussian flood has probability at most $p \cdot 2^{1.3}$ under the implemented sampler: the failure bound degrades from 2^{-106} to at most $2^{-104.7}$, which we absorb by margin. Security is unaffected entirely (Lemma 6.1 holds for any flood distribution). The fixed-point precision of the sampler itself contributes only $\approx 1.4 \times 10^{-7}$ to the per-sample log-ratio, two orders of magnitude below δ_s (see Section 9.7).

7 AEAD Composition

The RLWE layer is CPA-secure and malleable. Authenticity is provided by an external AEAD scheme (e.g., AES-256-GCM) applied *before* RLWE encryption. The AEAD key must be unavailable to re-randomizers. Associated data should include a unique context (file ID, monotonic version counter) to prevent replay, rollback, or ciphertext-swap attacks.

Threat model for re-randomizers. We adopt a Dolev–Yao-style network adversary [19]: the attacker fully controls the network and the re-randomizing relays—it can read, intercept, inject, modify, replay, and reorder any ciphertext, and may corrupt any subset of relays, but cannot break the underlying cryptographic primitives (DRLWE and the AEAD). Within this model, re-randomizers are *malicious*: they may corrupt ciphertexts arbitrarily (e.g., add non-Gaussian noise, replace components, or exceed the noise budget). The AEAD tag check at decryption detects all such tampering, so confidentiality and integrity reduce to the IND$ and AEAD guarantees even against fully malicious relays; the adversary's residual power is denial of service (DoS), discussed below. We note that the Extended Canetti–Krawczyk (eCK) model [20] does not directly apply here: eCK formalizes *authenticated key exchange* (session-key security under ephemeral- and long-term-key reveals), whereas our setting is public-key encryption relayed by untrusted parties with no interactive session or key agreement; the Dolev–Yao adversary is the appropriate abstraction. Key compromise is nonetheless addressed operationally: the AEAD key and RLWE secret key are independent, secret material is zeroized after use in the implementation (Section 9), and key rotation resets the system state (Remark 4.10).

"Noise bomb" DoS by malicious re-randomizers. A particularly stealthy DoS strategy deserves explicit analysis: a malicious relay can add a re-randomization packet whose *format* is valid but whose noise magnitude is enormous (e.g., e'_2 with coefficients near $q_2/2$), exhausting the noise budget in one hop and causing decryption failure. The AEAD detects the resulting corruption at decryption time (the failure cannot be silent), but cannot prevent it, and the attack is not attributable from the ciphertext alone: the noise magnitude of a ciphertext is not publicly observable without the secret key—any public test that distinguished high-noise from low-noise ciphertexts would in particular distinguish honest ciphertexts from uniform, contradicting IND$ (and such a noise meter would itself damage unlinkability, since an observer could correlate ciphertexts across hops by noise level). Mitigations therefore operate at the protocol level:

- *Verifiable re-randomization.* Each relay attaches a zero-knowledge proof that its added packet is a well-formed encryption of zero with bounded noise. Lattice-based proofs of correct re-randomization with exactly this structure are deployed in verifiable mix-nets [2, 3, 6]; they add per-hop proof sizes on the order of 10^2 kilobytes and proving times on the order of 10^2 milliseconds per ciphertext (Section 10), and make noise bombs publicly attributable (the bomber cannot produce a valid proof). We deliberately keep proofs out of the base scheme so that applications that do not need attribution avoid the cost.
- *Redundancy.* The owner stores or routes ρ replicas along disjoint relay paths; decryption succeeds unless all ρ paths contain a bomber. Combined with AEAD-based detection this also identifies which paths are compromised.
- *Accountability.* Relays sign the ciphertexts they output (signatures are stripped before the next hop in mix settings, or retained in storage settings). After an AEAD failure the owner replays the chain to locate the first corrupted hop.
- *Rate-limiting and reputation* at the admission layer bound the blast radius of any single relay.

In summary, a noise bomb can destroy individual ciphertexts but can never compromise confidentiality, integrity, or unlinkability; it is detected at decryption, and deployments requiring prevention or attribution can layer verifiable re-randomization proofs [2, 6] on top of the scheme unchanged.

Decryption-oracle hygiene. Implementations should treat the RLWE layer as a *raw* transport and ensure that any validity signal comes *only* from the AEAD verification. In particular, (i) the AEAD verification must be constant-time with respect to the tag and plaintext, and (ii) callers should receive a single generic failure indication (no distinct "RLWE failure" vs. "AEAD failure" codes). This avoids introducing a decryption-oracle side channel at the system level.

8 Concrete Security

We assess security for the DRLWE problem with $n = 4096$, $q = tq_2$ ($\log_2 q \approx 63$), and discrete-Gaussian secret/error with $\sigma = 3.2$. Since the adversary sees both CRT limbs sharing the same secret s, the effective lattice problem uses the full modulus q (Remark 2.4).

Rationale for the numerical values used in this section. The lattice parameters (n, q, σ) are those of Table 1, whose design rationale is given in Section 3; here they are simply carried into the hardness assessment. The remaining values are calibrated as follows. The security target is the standard 128-bit level, chosen to match the HE Standard tables [16] and NIST category guidance. The per-coefficient decryption-failure target $\delta_{\text{coeff}} = 2^{-128}$ aligns the correctness guarantee with that same 128-bit level (a failure is a correctness event, but keeping it at the security level is conservative and removes any failure-based attack surface). The batch size $B = 1024$ models a realistic storage unit (1024 slots $\times$ 15.5 KiB = 15.5 MiB of payload, 64 MiB of ciphertext) and yields the batch failure target $p_{\text{batch}} = \delta_{\text{coeff}} \cdot Bn = 2^{-128} \cdot 2^{22} = 2^{-106}$ via the union bound (Remark 4.9). The query budget $Q = 2^{40}$ in the multi-query analysis is a deliberately generous upper bound on the number of ciphertexts a single key will ever produce (one encryption per second for ∼35,000 years); the hybrid loss $Q \cdot 2^{-128} = 2^{-88}$ remains negligible. Finally, the estimator is configured

with one ring sample (the public key), the standard regime for RLWE public-key encryption, as discussed below.

HE Security Standard. The HE Standard v1.1 [16] lists, for $n = 4096$ and error width $\sigma \approx 3.2$, maximum moduli of $\log_2 q = 109$ (128-bit classical, cost model BKZ.sieve) and $\log_2 q = 101$ (128-bit quantum, BKZ.qsieve) for its most conservative secret distribution (ternary); for Gaussian error-distribution secrets, as used here, the corresponding entries are 111 and 103. We adopt the stricter ternary values throughout. Our $\log_2 q = 63$ uses only 58% of the classical budget and 62% of the quantum budget.

Lattice-estimator inputs and outputs. Table 2 lists the exact inputs for the lattice estimator [21]. A ready-to-run SageMath script is included in the Online Resource (`tools/lattice_estimate_sage.py`).

Primary yardstick: HE Security Standard. The HE Standard v1.1 tables [16] are the community-accepted conservative benchmark. For each parameter set they tabulate the security of the primal-uSVP, decoding, and dual attacks under the BKZ.sieve (classical) and BKZ.qsieve (quantum) cost models, and the admissible modulus is set by the weakest of the three. For $n = 4096$ and $\sigma \approx 3.2$, the most conservative 128-bit thresholds are $\log_2 q \leq 109$ (classical) and $\log_2 q \leq 101$ (quantum). Our $\log_2 q = 63$ uses only 58% and 62% of these budgets, respectively, confirming security comfortably exceeds 128 bits in both settings. *We recommend the HE Standard comparison as the authoritative security assessment.*

Auxiliary cross-check: simplified estimator. Table 2 also reports cost estimates from a simplified primal-uSVP / dual-distinguishing estimator using the core-SVP sieving model ($T_{\text{cl}} = 2^{0.292\beta}$, $T_{\text{qu}} = 2^{0.265\beta}$), configured with one ring RLWE sample (equivalently $n = 4096$ plain-LWE samples). The resulting costs ($\geq 2^{394}$ classical, $\geq 2^{357}$ quantum) are *lower bounds* under this specific cost model and do not account for hybrid, primal-BDD, or other advanced attack variants. These numbers are large because our $\log_2 q = 63$ is far below the HE Standard boundary for $n = 4096$; the estimator correctly reflects this wide margin but should not be interpreted as a precise "bits of security" claim. The HE Standard comparison above is the more conservative and operationally relevant measure.

Sample count and multi-query security. Two distinct "sample count" questions arise:

(i) Lattice samples of the secret s. The public key $(a, b = as + e)$ constitutes *one* Ring-LWE sample of s; in the ring setting this yields $n = 4096$ effective plain-LWE samples via the coefficient embedding. Each ciphertext $(c_0, c_1) = (br + e_2 + \Delta M,\ ar + e_1)$ uses the *same, already-known* public key with fresh per-encryption randomness (r, e_1, e_2); it does not reveal a new independent sample of s. Consequently, 2^{20}, 2^{30}, or 2^{40} ciphertexts under a single key do not increase the number of lattice samples available for key recovery. The estimator in Table 2 is configured with one ring sample (equivalently n plain-LWE samples), the standard regime for RLWE public-key encryption and the one used by the HE Security Standard [16].

(ii) Encryption queries in the IND$*/IND-CPA game.* An adversary making Q encryption queries sees Q independent ciphertext/uniform pairs. The IND$ proof (Section 5) proceeds via a standard hybrid argument over queries: each query is switched from real to random using one DRLWE invocation, so the advantage degrades by at most a factor of Q. For $Q \leq 2^{40}$ and a single-query DRLWE distinguishing advantage of at most 2^{-128}, the residual multi-query advantage is $\leq 2^{40} \cdot 2^{-128} = 2^{-88}$, which is negligible.

9 Implementation and Evaluation

9.1 Reference Implementation

We provide a Rust crate (`pq-rerand` v0.2.0, ∼1300 LOC, `#![forbid(unsafe_code)]`) implementing the full KeyGen/Enc/ReRand/Dec pipeline with 31-bit plaintext encoding, 44 unit and integration tests (including a 100-rerand stress test), criterion benchmarks, and a TVLA-style timing-leakage harness. The implementation is written to be *constant-time on the hot path*: merged Cooley–Tukey/Gentleman–Sande NTTs for both 32-bit primes with branchless Barrett and Shoup modular arithmetic (fixed instruction sequences, no secret-dependent branches or table indices), a constant-time CDT discrete-Gaussian sampler for σ, a constant-time inverse-CDF

Table 2 Lattice-estimator inputs and security assessment

Parameter	Value
Ring dimension n	4096
Modulus q	9,223,424,808,446,795,777 ($\approx 2^{63}$)
Secret distribution	Discrete Gaussian, $\sigma = 3.2$
Error distribution	Discrete Gaussian, $\sigma = 3.2$
Simplified estimator outputs (core-SVP sieving model):	
Best attack (dual)	BKZ-$\beta \approx 1349$, $d \approx 8406$
Classical cost	$\geq 2^{394}$ operations
Quantum cost	$\geq 2^{357}$ operations
HE Standard v1.1 cross-reference ([16], conservative ternary-secret rows):	
128-bit classical budget	$\log_2 q \leq 109$ (our $\log_2 q = 63$, 58%)
128-bit quantum budget	$\log_2 q \leq 101$ (our $\log_2 q = 63$, 62%)

sampler for σ_f, and a branchless center-lift in decryption (Remark 3.1); engineering details are given in Section 9.7. All randomness is drawn from caller-supplied CSPRNGs (enforced via Rust's `CryptoRng` trait bound) and transient secrets are zeroized after use. The crate is single-threaded at its core with an optional `rayon`-based batch layer for multi-core desktops, and builds unmodified for `aarch64` (mobile/ARM) targets. Source code and noise simulator are available at https://github.com/massalabs/pq-rerand. Figures are generated by `python3 tools/make_figures.py` (deterministic seed `2024`); lattice-estimator inputs are provided in `tools/lattice_estimate_sage.py` (requires SageMath). The full source tree—including Rust crate, simulation scripts, `Cargo.lock`, and exact commands for reproducing all figures and benchmarks—will be archived as Online Resource with pinned dependency versions.

9.2 Memory Layout

Each ciphertext comprises four `uint32` arrays of length n:

Component	Contents	Size
`c0_t[4096]`	$c_{0,t}$ coefficients	16 KiB
`c0_q2[4096]`	c_{0,q_2} coefficients	16 KiB
`c1_t[4096]`	$c_{1,t}$ coefficients	16 KiB
`c1_q2[4096]`	c_{1,q_2} coefficients	16 KiB
Total per ciphertext		64 KiB
Batch of B=1024 ciphertexts		64 MiB

The layout is cache-friendly (sequential access per NTT) and trivially serializable (little-endian `uint32` dump). No bitpacking, no CRT recombination, no modulus switching.

Plaintext encoding. Since $t \approx 2^{31}$, each polynomial coefficient carries 31 bits of payload. With $n = 4096$ coefficients this yields $4096 \times 31 = 126{,}976$ bits $= 15{,}872$ bytes (15.5 KiB) per ciphertext slot. Application data (e.g., an AEAD-encrypted blob) is bitpacked into coefficients: the encoder treats the input as a bitstream and fills each coefficient with 31 bits, producing a bijection between 15,872-byte payloads and elements of R_t with coefficients in $[0, 2^{31})$. Since all such values are $< t$, the encoding is lossless under ring arithmetic modulo t.

9.3 Comparison with Standard BFV Stacks

Table 3 contrasts the design footprint of our two-limb construction with that of a general-purpose RNS–BFV library: restricting to exactly two CRT primes and three operations eliminates relinearization, evaluation keys, mixed word sizes, and rounding logic, shrinking the trusted code base by more than an order of magnitude (quantitative same-machine performance against SEAL is given in Section 9.4).

9.4 Benchmarks

For reproducibility we record the full benchmarking environment in Table 4 and report every timing in *two* CPU configurations: **fixed** (governor `performance`, turbo boost disabled via `cpupower`/`sysfs`, all cores locked at the 3.8 GHz

Table 3 Comparison with standard BFV stacks

Aspect	This work	General BFV (e.g., SEAL [8])
CRT primes	2 (exactly t, q_2)	Typically 2–5 (RNS chain)
Relinearization	None	Required for mul
Evaluation keys	None	Required; $O(n \cdot L)$
Decryption	Exact (noise-limb)	Rounded
NTT word size	32-bit only	Mixed 32/64-bit
Code complexity	∼1300 lines of code	>50K lines of code

base clock—the primary, reproducible configuration) and **turbo** (boost enabled, OS-default scaling, included for reference). All timings are criterion medians from `cargo bench --release`.

Table 5 reports single-ciphertext costs. A batch of $B = 1024$ ciphertexts scales linearly when single-threaded (fixed config: re-randomize 0.53 s, decrypt 0.21 s) and is embarrassingly parallel: with the `rayon` batch layer on 16 hardware threads the fixed-config batch costs drop to 0.098 s (encrypt), 0.072 s (re-randomize), and 0.026 s (decrypt), a 7–8× speedup. Relative to the v0.1.0 research prototype benchmarked in the original submission (Box–Muller sampler, textbook NTT), the constant-time v0.2.0 is 4.1× faster on encryption, 3.1× on re-randomization, and 7.1× on decryption (fixed config, like for like), showing that the constant-time hardening came with a substantial performance gain rather than a penalty.

Same-machine comparison with Microsoft SEAL. To position these numbers against a state-of-the-art BFV stack, we benchmarked Microsoft SEAL 4.3 [8] on the same machine, same compiler flags (`-O3 -march=native`, bundled dependencies, no HEXL), at $n = 4096$ with batch-encoded random data and `plain_modulus` set to our t, in two configurations: SEAL's default 128-bit parameter set ($\log_2 q = 109$, 3 RNS primes) and a modulus-matched set ($\log_2 q = 63$, $\{31, 32\}$-bit primes). Table 6 summarizes the fixed-frequency results. Our encryption is faster than SEAL's default 128-bit configuration and within 25% of the modulus-matched configuration *while additionally folding in the σ_f flood packet* (which by itself accounts for ≈ 0.27 ms of sampling; without flooding the gap closes), and our exact noise-limb decryption is faster than SEAL's rounded decryption in both configurations. (SEAL has no public re-randomization API; its closest primitive, adding a fresh encryption of zero, costs one encryption plus one addition.) The point of this comparison is not that a 1300-line special-purpose crate "beats" a general-purpose homomorphic-encryption library, but that the two-limb design achieves state-of-the-art per-operation latency while eliminating the RNS machinery, rounding logic, and relinearization keys.

9.5 Noise Simulation

Re-randomization budgets. Table 7 summarizes $k_{\max}$ under the fixed-key bound (Theorem 4.6) for two configurations.

9.6 Noise Budget Management and Reset

The budget $k_{\max}$ is so large (∼476 years at one re-randomization per second) that most deployments will never approach it; nevertheless, a complete system needs a defined recovery path. We recommend the following operational protocol.

Tracking. The re-randomization count k is deliberately absent from the ciphertext (its presence would break unlinkability), so it cannot be enforced cryptographically. Instead the *system* tracks a conservative upper bound on k out of band: in storage settings, the owner records creation time and the maximum refresh rate of the network, giving $k \le k_{\text{est}} = \text{age} \times \max \text{rate}$; in mix settings, k is bounded by the (known) number of hops. Budget sizing with the safety margins of Theorem 4.6 means k_{est} may overestimate k by orders of magnitude without practical consequence.

Reset by re-encryption. When k_{est} approaches a chosen fraction of $k_{\max}$ (e.g., 50%), the owner decrypts and re-encrypts the ciphertext. This

Table 4 Benchmark environment

CPU	AMD Ryzen 7 260 (8C/16T, Zen 4)
Clock config (**fixed**)	governor `performance`, turbo off, 3.8 GHz locked
Clock config (**turbo**)	boost to 5.0 GHz, OS-default scaling
RAM	32 GiB DDR5
OS	Ubuntu, Linux 7.0.0-22-generic (x86_64)
Rust toolchain	`rustc 1.94.1`, `--release`, `-C target-cpu=native`, thin LTO
Crate version	`pq-rerand` v0.2.0 (no SIMD intrinsics, no `unsafe`)

Table 5 Operation timings per ciphertext (criterion medians, single-threaded)

Operation	Fixed (3.8 GHz, turbo off)	Turbo
Key generation	0.92 ms	0.73 ms
Encrypt (incl. flood)	0.80 ms	0.63 ms
Re-randomize	0.51 ms	0.41 ms
Decrypt	0.21 ms	0.16 ms

Table 6 Same-machine BFV comparison (fixed 3.8 GHz, turbo off, $n = 4096$, medians of 3000 iterations)

Implementation	Encrypt	Decrypt
SEAL 4.3, default-128 ($\log_2 q$=109)	0.86 ms	0.25 ms
SEAL 4.3, matched ($\log_2 q$=63)	0.65 ms	0.25 ms
This work ($\log_2 q$=63, incl. flood)	**0.80 ms**	**0.21 ms**

requires the secret key—by design, since a public noise reset would equate to unbounded re-randomization, which the noise growth of Ring-LWE forbids without bootstrapping. The operation is cheap (0.21 ms + 0.80 ms per ciphertext, Table 5), restores the full budget ($k = 0$), and produces an output unlinkable to its input (Theorem 5.3). Because decryption is exact (Theorem 4.3), the cycle is lossless and can be repeated indefinitely, making the effective lifetime unlimited for any owner who can periodically touch the data. The same operation doubles as the key-rotation step (Remark 4.10).

If the limit is exceeded. Should a ciphertext exceed $k_{\max}$ before reset (e.g., a runaway relay), decryption may fail, but the failure is *detected*—the AEAD tag cannot verify (Section 7)—and never silent. Redundant replicas stored along independent paths (Section 7) then provide recovery. Public alternatives that avoid the secret key entirely exist but are disproportionate here: FHE ciphertext sanitization/bootstrapping [12] or blind-rotation-based refresh in proxy re-encryption [7] cost orders of magnitude more per operation than the entire remaining budget justifies.

Noise simulation (fixed-key, with flooding). To match the operational setting of Theorem 4.6 (key fixed for the lifetime of the system), we ran a *fixed-key* simulation: 10 independently sampled keys, each with 5 independent trials over encryption/re-randomization randomness only, using the optional flooding baseline ($\kappa_f = 858{,}000{,}000$). Figures 1–4 summarize the results. For each key, all $\|e\|_2^2$ and $\|s\|_2^2$ values were close to the expected $n\sigma^2 \approx 41{,}943$, well within the good-key bound $2n\sigma^2 = 83{,}886$ (Lemma 4.4). The inter-key variability in max noise is small ($\sim$3% relative standard deviation), confirming that the ℓ_2 conditioning is not a practical bottleneck (see Figure 4).

These experiments illustrate the $\sqrt{k}$ scaling behavior and typical noise magnitudes (the per-coefficient noise distribution after $k = 1000$ re-randomizations is shown in Figure 3); they do *not* empirically confirm the stated ultra-low failure probabilities ($\delta_{\text{coeff}} \approx 2^{-128}$), which are inaccessible to simulation and are guaranteed only by the analytical bound (Theorem 4.6). The substantial gap between the empirical noise and the worst-case bound visible in Figure 2 is expected, and serves as the empirical validation of the bound's looseness requested in Remark 4.9: it arises from (i) the union bound over $Bn = 4{,}194{,}304$ coefficients (each individually at $\delta_{\text{coeff}} = 2^{-128}$; quantified at $\leq 10\%$ of B_{coeff} in Remark 4.9), (ii) the ℓ_2 good-key conditioning using $\|e\|_2^2 \leq 2n\sigma^2$ whereas a typical key has $\|e\|_2^2 \approx n\sigma^2$ (a $1/\sqrt{2}$ factor; Remark 4.8), and (iii) sub-Gaussian tail conservatism at moderate noise levels (the Gaussian tail decreases faster than the sub-Gaussian envelope; the dominant contribution at the plotted scales).

Table 7 Fixed-key re-randomization budget ($p_{\text{batch}} = 2^{-106}$, $B = 1024$, ℓ_2 good-key conditioning)

Configuration	$k_{\max}$	Real-time equivalent
No flooding ($\kappa_f = 0$)	$\approx 1.5 \times 10^{10}$	∼476 years at 1/s
With flooding ($\kappa_f = 8.58 \times 10^8$)	$\approx 1.4 \times 10^{10}$ additional	∼448 years at 1/s

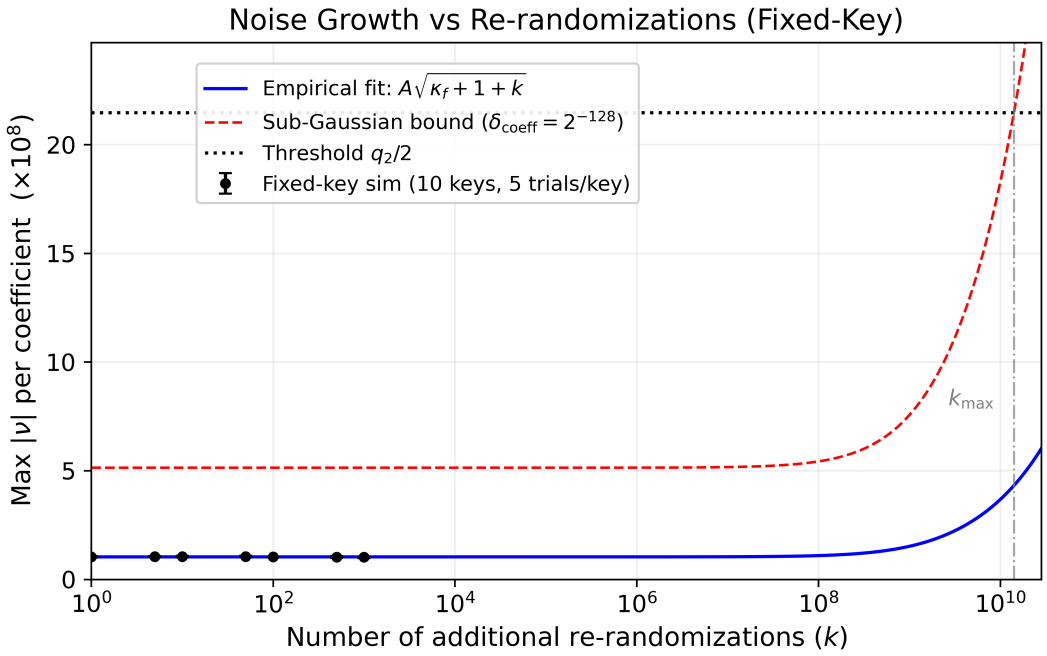

Fig. 1 Max $|\nu^{(k)}|$ vs. additional re-randomizations k (with flooding baseline $\kappa_f = 858{,}000{,}000$; log scale). Dots: fixed-key simulation (10 keys, 5 trials/key; error bars show cross-key variability). Solid curve: empirical fit. Dashed: fixed-key sub-Gaussian bound (Theorem 4.6). Dotted: threshold $q_2/2$. Dash-dotted vertical line: $k_{\max}$ from (6)

9.7 Sampler Engineering and Side Channels

Base-width sampler ($\sigma = 3.2$): constant-time CDT. The implementation samples $D_{\mathbb{Z},\sigma}$ with a cumulative distribution table over folded magnitudes: $\text{cdt}[m] = \lfloor 2^{63} \cdot \Pr[|X| \le m] \rfloor$ for $m = 0, \ldots, 39$ (40 entries × 64 bits = 320 bytes, resident in a fraction of one cache line pair). The tail cut at $12.5\sigma = 40$ discards mass $< 2^{-63}$, below the resolution of the comparison. Per coefficient, the sampler draws one 63-bit uniform u, computes the magnitude as the number of table thresholds less-than-or-equal to u by scanning *all 40 entries unconditionally* with a branchless comparison (sign-bit extraction, accumulated arithmetically), and applies an independently drawn uniform sign unconditionally. Control flow, instruction count, and the memory access pattern are independent of the sampled value. Measured cost at the fixed 3.8 GHz configuration: 15.8 ns per coefficient (64.8 µs per $n = 4096$ polynomial).

Flood-width sampler ($\sigma_f \approx 93{,}733$): constant-time inverse-CDF. A CDT at this width would need $> 10^5$ entries; the implementation instead uses inverse-transform sampling with *no table at all*: draw a 53-bit uniform mantissa u, set $p = (u + 1/2) \cdot 2^{-53} \in (0, 1)$, evaluate the probit $z = \Phi^{-1}(p)$ via Acklam's rational approximation (relative error $< 1.15 \times 10^{-9}$) with all three approximation regions computed unconditionally and combined by arithmetic masks, using a branchless bit-manipulation ln (truncated atanh series, absolute error $< 10^{-7}$) and a hardware $\sqrt{\cdot}$, then output $\lfloor z\,\sigma_f \rceil$. There is no rejection loop,

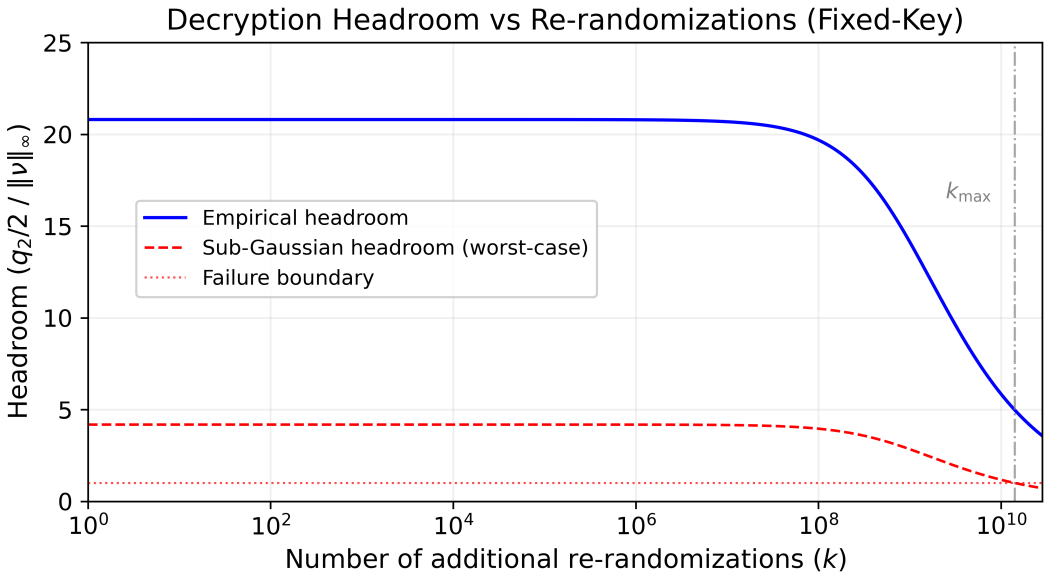

Fig. 2 Decryption headroom $(q_2/2)/\|\nu^{(k)}\|_\infty$ vs. additional re-randomizations k (with flooding baseline $\kappa_f = 858{,}000{,}000$). Vertical line: $k_{\max}$ from (6)

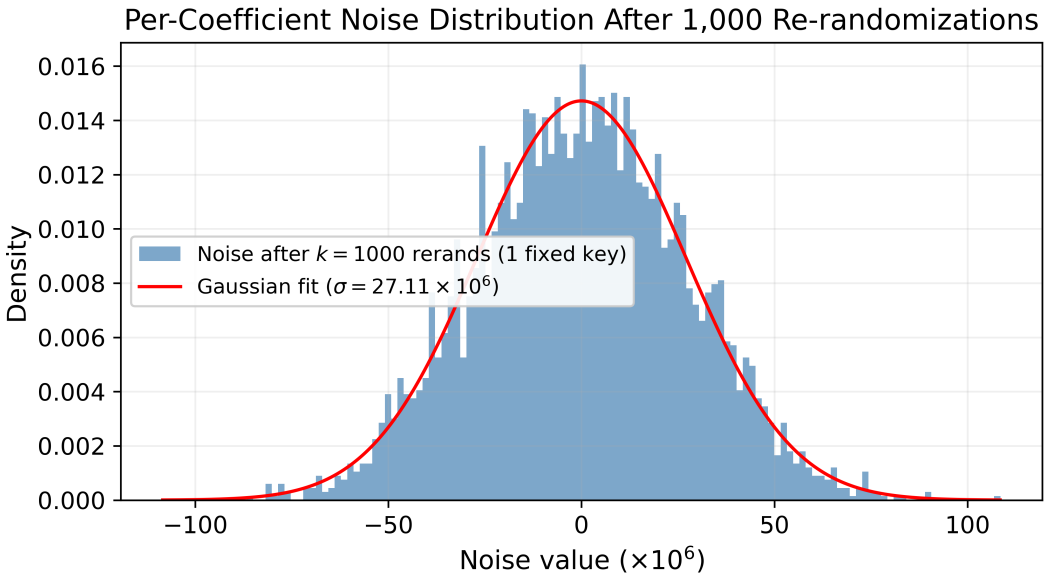

Fig. 3 Per-coefficient noise distribution after $k = 1{,}000$ additional rerands on top of flooding baseline $\kappa_f = 858{,}000{,}000$ (single fixed key)

no data-dependent branch, and no table indexing. *Precision:* the probit error ($\leq 1.15 \times 10^{-9}$ relative over $|z| \leq 8.7$, the range reachable from a 53-bit mantissa) and the ln error (perturbing z by $\leq 1.2 \times 10^{-8}$) shift each output by at most $\approx 10^{-3}$ in absolute value after scaling by σ_f, contributing $\leq 1.4 \times 10^{-7}$ to the per-sample log-probability ratio—two orders of magnitude below the rounded-vs-discrete Gaussian gap $\delta_s = 2^{-13.8}$

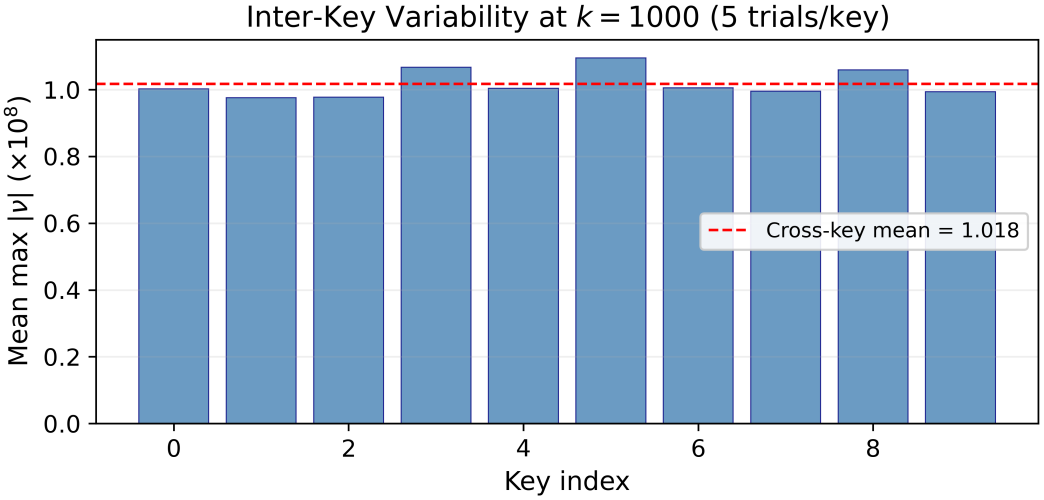

Fig. 4 Inter-key variability of max $|\nu^{(k)}|$ at $k = 1{,}000$ additional rerands (with flooding baseline $\kappa_f = 858{,}000{,}000$) across 10 independently sampled keys (5 trials per key). The dashed horizontal line shows the cross-key mean. Per-key variation is ∼3%, confirming that fixed-key behavior is well-concentrated

of (7), so 53 fixed-point mantissa bits and double-precision arithmetic suffice for the Rényi bound of Section 6 ($R_\infty \leq 2^{1.3}$ per packet) to hold. Measured cost: 22.1 ns per coefficient (90.7 µs per polynomial; 0.27 ms for the three flood polynomials of one packet), included in the encryption timings of Table 5.

Distributional gap. The correctness analysis models flood coefficients as discrete Gaussians; the implemented sampler produces rounded continuous Gaussians. Section 6 bounds the resulting Rényi divergence explicitly and shows the decryption-failure bound degrades by at most a factor $2^{1.3}$, while security is unaffected (Lemma 6.1). For the correctness tail bounds, a rounded continuous Gaussian coefficient is $(\sigma_f + 1)$-sub-Gaussian (the deterministic rounding offset is bounded by $1/2$), a negligible relative widening at $\sigma_f \approx 9.4 \times 10^4$ that is absorbed by the margins of Theorem 4.6. Any sampler that is $O(\sigma_f)$-sub-Gaussian and Rényi-close to $D_{\mathbb{Z},\sigma_f}$ is a valid drop-in replacement.

Constant-time arithmetic. Beyond the samplers, the hot path is engineered for data-independent timing: NTT butterflies use Barrett reduction and Shoup multiplication with branchless conditional subtraction (mask arithmetic instead of `if`); the decryption center-lift follows Remark 3.1; and there are no secret-indexed memory accesses anywhere. The uniform sampler for the *public* polynomial a uses rejection sampling, whose timing depends only on public data.

Empirical leakage assessment (TVLA). We validate the constant-time claims with a dudect-style [22] fixed-vs-random Welch t-test harness, run at the fixed-frequency configuration with cycle-accurate timestamps (30,000–400,000 measurements per class, pass threshold $|t| < 4.5$, worst of raw and tail-cropped statistics). Results: encrypt (fixed-random vs. random plaintext) $|t| = 1.09$; re-randomize (fixed vs. random ciphertext) $|t| = 1.33$; decrypt (secret-dependent path) $|t| = 2.28$—all passing—while a deliberately leaky modular-reduction control registers $|t| > 3000$, confirming the harness's sensitivity. One physical-layer caveat is worth reporting for auditors: on the test CPU (Zen 4), a diagnostic comparing *all-zero* versus random plaintexts intermittently shows $|t|$ up to ∼10, and a no-crypto control (a pure load-XOR loop over all-zero vs. random buffers) shows $|t| > 60$: operand *data content* (bit-toggle density) measurably modulates timing at the silicon level, independent of program control flow. This hardware effect is outside the constant-time software contract and is immaterial in our deployment model, where RLWE plaintexts are AEAD outputs and hence uniformly distributed (Section 7); degenerate all-zero plaintexts do not occur. The leakage harness ships with the crate

(`examples/leakage.rs`) for independent reproduction on other microarchitectures, including `aarch64` (it reads `cntvct_el0` there in place of `rdtsc`).

10 Related Work

Ring-LWE originates with Lyubashevsky, Peikert, and Regev [13]. The BFV family appears in Fan and Vercauteren [23]; RNS variants in Halevi, Polyakov, and Shoup [11]. Noise flooding and ciphertext sanitization are studied by Ducas and Stehlé [12]; Rényi divergence proof techniques by Bai et al. [18]. ML-KEM (FIPS 203) [4] standardizes Module-LWE key encapsulation. Guo et al. [24] showed key-recovery risks from non-worst-case noise flooding in approximate HE, motivating our conservative calibration.

Standard BFV implementations (SEAL [8], OpenFHE [9], lattigo) use multi-prime RNS and support full homomorphic evaluation. Our construction restricts to two primes and three operations (Enc, ReRand, Dec), trading generality for simplicity.

Re-randomizable and re-encryptable schemes. Universal re-encryption for mixnets was introduced by Golle et al. [1] over ElGamal: unlimited re-randomizations with statistical unlinkability, but no post-quantum security. Its lattice-based adaptation by Singh et al. [10] is, to our knowledge, the closest prior primitive to ours; however, it builds on a GHV-style matrix cryptosystem whose $m \times m$ matrix ciphertexts are orders of magnitude larger than ours, its re-encryption multiplies the ciphertext by fresh low-norm matrices so the noise compounds multiplicatively at every mixing step, and it provides neither concrete parameters, noise budget analysis, nor an implementation. Re-randomizable RCCA-secure (Rand-RCCA) encryption [25, 26] strengthens the security target to replayable CCA and even yields tightly-secure mixnets, but existing efficient Rand-RCCA constructions rely on Diffie–Hellman-type assumptions in cyclic groups [25] or pairing groups [26] and are therefore not post-quantum; no lattice-based Rand-RCCA scheme with a practical implementation is known. (We note that Kyber/ML-KEM [4] itself is a KEM and does not support ciphertext re-randomization.) In the lattice setting, verifiable re-encryption mixnets are an active line of work: Aranha et al. [2] build a verifiable shuffle with an amortized proof of correct re-randomization for BGV ciphertexts, Hough et al. [3] improve costs using NTRU, and Bootle et al. [6] revisit soundness of lattice mixnets. These systems target verifiability (zero-knowledge proofs per shuffle) and pay correspondingly: in [2], re-randomization-and-shuffle proofs amount to hundreds of kilobytes (370 KB) and hundreds of milliseconds ($\approx$ 260 ms proving) *per ciphertext per mix*, on top of the BGV arithmetic; [3] reduces the proof material to roughly 125 KB per ciphertext per mix. Our scheme is complementary: it provides the bare re-randomizable encryption layer (proofs can be added on top exactly as in these works; cf. Section 7). Lattice proxy re-encryption [7] re-encrypts ciphertexts *between different keys* via re-encryption keys and blind rotation, a stronger functionality with correspondingly heavier machinery (FHEW-style bootstrapping per hop); public re-randomization keeps the key fixed and needs only one encryption of zero. Finally, FHE ciphertext sanitization [12] achieves statistical re-randomization with an *unbounded* budget, since each bootstrapping resets the noise to a canonical level. It is, however, qualified in three ways: the public sanitizer needs the published FHE bootstrapping key and hence a circular-security assumption; the statistical guarantee applies to honestly generated ciphertexts (a definitional correction recorded in a 2025 erratum to [12], following Smart and Walter [27]); and each sanitization costs a bootstrapping operation—orders of magnitude above our 0.51 ms re-randomization. Table 8 summarizes the comparison.

Classical ElGamal-based universal re-encryption thus remains the only approach with an unbounded budget and cheap operations, but it is not post-quantum; among post-quantum options, ours is the only one that simultaneously preserves ciphertext size, decrypts exactly, proves a concrete (and astronomically large) re-randomization budget, and ships a hardened implementation.

Closest prior technique. Public re-randomization via "add an encryption of zero" is a standard technique in the LPR/BFV literature [13, 23]. The genuine novelty of the present work is the *two-limb CRT message embedding* enabling exact noise extraction. In

Table 8 Publicly re-randomizable encryption approaches. "Size preserved" = ciphertext does not grow with re-randomizations; "ReRand budget" = admissible re-randomization count. Costs are as reported in the cited works (different platforms; order-of-magnitude comparison only). Proxy re-encryption [7] is omitted as it provides key-switching re-encryption, not public re-randomization (see text)

Scheme	PQ	Size pres.	Exact Dec	ReRand budget	ReRand cost	Impl.
ElGamal UR [1]	no	yes	yes	∞	$\sim$0.2 ms (4 exp.)	yes
Lattice UR [10]	yes	yes	yes	unanalyzed[a]	matrix ops, no impl.	no
Rand-RCCA [25, 26]	no	yes	yes	∞	group/pairing ops	partial
Verif. mixnet [2, 3]	yes	yes	yes	per-mix design	10^2 ms + 10^2 KB proof	yes
Sanitization [12]	yes	yes	no	∞[b]	bootstrap ($\gg$100 ms)	no
This work	yes	yes	yes	1.5×10^{10} (proven)	0.51 ms	yes

[a]Noise compounds multiplicatively per hop (exponential in the hop count), so only a few re-randomizations can decrypt for any fixed modulus. [b]Requires the published FHE bootstrapping key (circular-security assumption); statistical guarantee for honestly generated ciphertexts (2025 erratum to [12]).

standard BFV, the scaling factor $\Delta = \lfloor q/t \rfloor$ is approximate, and decryption requires rounding: $M = \lceil (t/q)(c_0 - c_1 s \bmod q) \rfloor$. The RNS–BFV framework of Halevi, Polyakov, and Shoup [11] decomposes q into CRT primes for computational efficiency but still performs approximate scaling and base extension during decryption. Our observation is that choosing $q = t \cdot q_2$ *exactly* (so $\Delta = q_2$ is an integer) makes the plaintext contribution vanish entirely in the q_2-limb: $\Delta M \equiv 0 \pmod{q_2}$. This turns decryption into exact arithmetic in each CRT limb separately—no CRT recombination modulo q, no rounding, no base extension. To the best of our knowledge, this specific "message-vanishing limb" formulation and its exploitation for exact noise extraction have not appeared in prior RLWE/RNS engineering work.

11 Conclusion

We introduced a two-limb CRT Ring-LWE encryption scheme with exact decryption and public re-randomization. The "noise limb" enables exact noise extraction without CRT recombination or rounding. On the theory side, we give a clean DRLWE-based IND$ proof; IND-CPA and game-based pairwise unlinkability follow as corollaries. The exact aggregation lemma (Lemma 4.1) shows that repeated re-randomizations introduce no hidden cross-step dependence, keeping the correctness analysis clean. On the engineering side, the scheme admits a simple 32-bit NTT implementation with a flat ciphertext layout, realized here as a constant-time Rust crate with TVLA-validated timing behavior.

Limitations. The RLWE layer is CPA-only (authenticity requires the AEAD composition of Section 7). Re-randomization is bounded by the noise budget; resetting the budget requires the secret key (Section 9.6). Malicious re-randomizers can mount detectable-but-not-preventable DoS ("noise bombs"), whose mitigation requires protocol-level measures or verifiable re-randomization proofs (Section 7). Our constant-time guarantees cover control flow and memory addressing; physical-layer data-dependent effects observed on recent CPUs are documented in Section 9.7 and excluded by the AEAD deployment model. The implementation, while hardened, has not yet undergone an independent third-party audit.

Open problems. Can the noise-limb trick support limited homomorphic evaluation while preserving re-randomizability? Can the noise budget be refreshed without bootstrapping?

A Concentration toolkit for correctness bounds

This appendix records the elementary concentration facts used in Section 4. We use two standard notions.

Definition A.1 (Sub-Gaussian / sub-exponential) A centered random variable X is *σ-sub-Gaussian* if $\mathbb{E}[e^{\lambda X}] \le e^{\lambda^2\sigma^2/2}$ for all $\lambda \in \mathbb{R}$. A centered random variable Y is *K-sub-exponential* if $\mathbb{E}[e^{|Y|/K}] \le 2$.

Lemma A.2 (Products of sub-Gaussians are sub-exponential) *If X is σ_X-sub-Gaussian and Z is σ_Z-sub-Gaussian, and X and Z are independent, then XZ is K-sub-exponential with $K \le 4\sigma_X\sigma_Z$.*

Proof This is a standard consequence of Orlicz-norm inequalities [28] ($\|XZ\|_{\psi_1} \le C\|X\|_{\psi_2}\|Z\|_{\psi_2}$) specialized to the mgf-based definitions above. We take a conservative constant factor 4. □

Lemma A.3 (Bernstein for sub-exponential sums) *Let $Y_1, \dots, Y_m$ be independent centered K-sub-exponential random variables. Then for all $t > 0$,*

$$\Pr\Big[\Big|\sum_{i=1}^{m} Y_i\Big| \ge t\Big] \le 2\exp\Big(-\min\Big(\frac{t^2}{16mK^2}, \frac{t}{4K}\Big)\Big).$$

Proof This is Bernstein's inequality for sub-exponential sums ([28], Theorem 2.8.1) with the absolute constant made explicit for our definition of K-sub-exponential. From $\mathbb{E}[e^{|Y_i|/K}] \le 2$ and $|y|^p \le p!\, e^{|y|}$ one gets $\mathbb{E}|Y_i|^p \le 2\,p!\,K^p$, hence for $|\lambda| \le 1/(2K)$, $\mathbb{E}[e^{\lambda Y_i}] \le 1 + 2\sum_{p\ge 2}(|\lambda|K)^p \le e^{4\lambda^2K^2}$ (using $\mathbb{E}[Y_i] = 0$). Applying Markov's inequality to $\exp(\lambda\sum_i Y_i)$, using independence, and optimizing $\lambda \in (0, 1/(2K)]$ yields the bound; repeat for $-\sum_i Y_i$. □

Lemma A.4 (Average-over-keys alternative (sub–exponential)) *If both the key (s, e) and the encryption/rerand randomness are drawn fresh, then each coefficient $\eta^{(k)}$ of $\nu^{(k)}$ satisfies*

$$\Pr\left[|\eta^{(k)}| \ge t\right] \le 2\exp\Big(-\min\Big(\frac{t^2}{(512n+256)\,\sigma^2\sigma_k^2}, \frac{t}{16\,\sigma\,\sigma_k}\Big)\Big).$$

This Bernstein-type (sub-exponential) bound requires no good-key conditioning (hence no $2\exp(-n/4)$ caveat) and exhibits the sub-exponential tail shape characteristic of products of Gaussians, at the price of conservative explicit constants.

Proof Fix a coefficient index. By negacyclic convolution, that coefficient of $e\,r^{(k)}$ is a signed sum of n terms $e_j\, r^{(k)}_{i-j}$, where the coefficients $\{e_j\}_j$ and $\{r^{(k)}_j\}_j$ are independent across j and independent of each other. By Lemma 2.2 and Remark 4.2, each e_j is σ-sub-Gaussian and each $r^{(k)}_j$ is σ_k-sub-Gaussian. Lemma A.2 implies each product is $(4\sigma\sigma_k)$-sub-exponential. The same holds for the n terms of $e_1^{(k)}s$. The remaining coefficient of $e_2^{(k)}$ is σ_k-sub-Gaussian, hence also $(4\sigma\sigma_k)$-sub-exponential for $\sigma \ge 1$. Thus the full coefficient is a sum of $m \le 2n+1$ independent K-sub-exponentials with $K = 4\sigma\sigma_k$. Applying Lemma A.3 with $m \le 2n+1$ and $K = 4\sigma\sigma_k$, and upper bounding $16mK^2 \le 16(2n+1)(4\sigma\sigma_k)^2 = (512n+256)\,\sigma^2\sigma_k^2$ and $4K = 16\sigma\sigma_k$, yields the stated bound. □

Remark A.5 The main correctness theorem (Theorem 4.6) uses the *fixed-key* sub-Gaussian bound (Lemma 4.5), which is operationally relevant for long-lived keys. Lemma A.4 provides an alternative for analyses where key randomness is available (e.g., bounding average failure over key generation).

Supplementary information. The Rust reference implementation (`pq-rerand` crate v0.2.0, ∼1300 LOC, including the TVLA leakage harness and batch layer), Python noise simulation and figure-generation scripts, and a SageMath lattice-estimator script are provided as Online Resource. All files will be archived with pinned dependency versions. Source code is also available at https://github.com/massalabs/pq-rerand.

Acknowledgments. The authors thank the anonymous reviewers for their constructive feedback.

Declarations

- **Funding.** No external funding was received for this research.
- **Competing interests.** The authors are employees of Massa Labs. The authors declare no other competing financial or non-financial interests.
- **Data availability.** All data generated or analysed during this study are included in this article and its Online Resource files. The source code and simulation scripts are available at https://github.com/massalabs/pq-rerand.
- **Author contributions.** D. Vodenicarevic conceived the scheme and wrote the initial draft. All authors contributed to the analysis, implementation, and manuscript preparation. All authors reviewed and approved the final manuscript.

- **Use of AI tools.** Claude Opus 4.6 (Anthropic) and ChatGPT 5.2 (OpenAI) were used for LaTeX formatting assistance and code completion during manuscript preparation and implementation.

References

[1] Golle, P., Jakobsson, M., Juels, A., Syverson, P.: Universal re-encryption for mixnets. In: Topics in Cryptology – CT-RSA 2004. LNCS, vol. 2964, pp. 163–178. Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24660-2_14

[2] Aranha, D.F., Baum, C., Gjøsteen, K., Silde, T.: Verifiable mix-nets and distributed decryption for voting from lattice-based assumptions. In: Proc. 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), pp. 1467–1481. ACM, New York, NY (2023). https://doi.org/10.1145/3576915.3616683

[3] Hough, P., Sandsbråten, C., Silde, T.: More efficient lattice-based electronic voting from NTRU. IACR Communications in Cryptology **1**(4) (2025) https://doi.org/10.62056/a69qudhdj

[4] NIST: Module-Lattice-Based Key-Encapsulation Mechanism Standard (FIPS 203). https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.203.pdf (2024)

[5] Chathurangi, M., Li, Q., Foo, E., Zhang, L.Y.: Post-quantum traceable anonymous credentials from lattices. IACR Communications in Cryptology **2**(4) (2026) https://doi.org/10.62056/ak5wl8n4e

[6] Bootle, J., Lyubashevsky, V., Merino-Gallardo, A.: Efficient Verifiable Mixnets from Lattices, Revisited. IACR ePrint Archive, Report 2025/658. https://eprint.iacr.org/2025/658 (2025)

[7] Wan, X., Wang, Y., Xue, H., Wang, M.: Unbounded Multi-Hop Proxy Re-Encryption with HRA Security: An LWE-Based Optimization. IACR ePrint Archive, Report 2025/656. https://eprint.iacr.org/2025/656 (2025)

[8] Microsoft Research: Microsoft SEAL (release 4.3). https://github.com/microsoft/SEAL (2026)

[9] Badawi, A.A., Bates, J., Bergamaschi, F., Cousins, D.B., Erabelli, S., Genise, N., Halevi, S., Hunt, H., Kim, A., Lee, Y., Liu, Z., Micciancio, D., Quah, I., Polyakov, Y., Saraswathy, R.V., Rohloff, K., Saylor, J., Suponitsky, D., Triplett, M., Vaikuntanathan, V., Zucca, V.: OpenFHE: Open-source fully homomorphic encryption library. In: Proc. 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography (WAHC '22), pp. 53–63. ACM, New York, NY (2022). https://doi.org/10.1145/3560827.3563379

[10] Singh, K., Rangan, C.P., Banerjee, A.K.: Lattice based universal re-encryption for mixnet. Journal of Internet Services and Information Security **4**(1), 1–11 (2014) https://doi.org/10.22667/JISIS.2014.02.31.001

[11] Halevi, S., Polyakov, Y., Shoup, V.: An improved RNS variant of the BFV homomorphic encryption scheme. In: Topics in Cryptology – CT-RSA 2019. LNCS, vol. 11405, pp. 83–105. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-12612-4_5

[12] Ducas, L., Stehlé, D.: Sanitization of FHE ciphertexts. In: Advances in Cryptology – EUROCRYPT 2016. LNCS, vol. 9665, pp. 294–310. Springer, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_12

[13] Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. J. ACM **60**(6), 1–35 (2013) https://doi.org/10.1145/2535925 . Article No. 43

[14] Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Advances in Cryptology – EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_41

[15] Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Advances in Cryptology – CRYPTO 2009. LNCS, vol. 5677, pp. 595–618. Springer, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_35

[16] HomomorphicEncryption.org: Homomorphic Encryption Security Standard. https://homomorphicencryption.org/standard/. Version 1.1, November 21, 2018 (2018)

[17] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proc. 40th Annual ACM Symposium on Theory of Computing (STOC '08), pp. 197–206. ACM, New York, NY (2008). https://doi.org/10.1145/1374376.1374407

[18] Bai, S., Lepoint, T., Roux-Langlois, A., Sakzad, A., Stehlé, D., Steinfeld, R.: Improved security proofs in lattice-based cryptography: Using the Rényi divergence rather than the statistical distance. J. Cryptology **31**(3), 610–640 (2018) https://doi.org/10.1007/s00145-017-9265-9

[19] Dolev, D., Yao, A.C.: On the security of public key protocols. IEEE Trans. Inf. Theory **29**(2), 198–208 (1983) https://doi.org/10.1109/TIT.1983.1056650

[20] LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Provable Security (ProvSec 2007). LNCS, vol. 4784, pp. 1–16. Springer, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75670-5_1

[21] Albrecht, M.R., et al.: Lattice Estimator. https://github.com/malb/lattice-estimator (2023)

[22] Reparaz, O., Balasch, J., Verbauwhede, I.: Dude, is my code constant time? In: Design, Automation & Test in Europe Conference (DATE 2017), pp. 1697–1702. IEEE, Lausanne, Switzerland (2017). https://doi.org/10.23919/DATE.2017.7927267

[23] Fan, J., Vercauteren, F.: Somewhat Practical Fully Homomorphic Encryption. IACR ePrint Archive, Report 2012/144. https://eprint.iacr.org/2012/144 (2012)

[24] Guo, Q., Nabokov, D., Suvanto, E., Johansson, T.: Key recovery attacks on approximate homomorphic encryption with non-worst-case noise flooding countermeasures. In: 33rd USENIX Security Symposium (USENIX Security 24), pp. 7447–7461. USENIX Association, Philadelphia, PA (2024)

[25] Prabhakaran, M., Rosulek, M.: Rerandomizable RCCA encryption. In: Advances in Cryptology – CRYPTO 2007. LNCS, vol. 4622, pp. 517–534. Springer, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_29

[26] Faonio, A., Hofheinz, D., Russo, L.: Almost tightly-secure re-randomizable and replayable CCA-secure public key encryption. In: Public-Key Cryptography – PKC 2023. LNCS, vol. 13941, pp. 275–305. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-31371-4_10

[27] Smart, N.P., Walter, M.: Error-simulatable sanitization for TFHE and applications. IACR Communications in Cryptology **2**(3) (2025) https://doi.org/10.62056/a0lmpgxq

[28] Vershynin, R.: High-Dimensional Probability: An Introduction with Applications in Data Science. Cambridge University Press, Cambridge (2018). https://doi.org/10.1017/9781108231596