SMAUG-T: the Key Exchange Algorithm based on Module-LWE and Module-LWR

Authors
Jung Hee Cheon (Seoul National University / CryptoLab, Inc.)
Hyeongmin Choe (University of Luxembourg)
Joongeun Choi (Defense Counter-intelligence Command)
Dongyeon Hong (Samsung Electronics)
Jeongdae Hong (CryptoLab, Inc.)
Chi-Gon Jung (Defense Counter-intelligence Command)
Honggoo Kang (Defense Counter-intelligence Command)
Minhyeok Kang (CryptoLab, Inc.)
Taekyung Kim (CryptoLab, Inc.)
Jeongbeen Ko (CryptoLab, Inc.)
Janghyun Lee (Defense Counter-intelligence Command)
Seonghyuck Lim (Defense Counter-intelligence Command)
Aesun Park (Defense Counter-intelligence Command)
Seunghwan Park (Defense Counter-intelligence Command)
Jungjoo Seo (CryptoLab, Inc.)
Hyoeun Seong
Junbum Shin (CryptoLab, Inc.)
MinJune Yi

Contact `taekyung.kim@cryptolab.co.kr`

Homepage `kpqc.cryptolab.co.kr`

Reference Code `github.com/CryptoLabInc/SMAUG-T`
Integrity Hash (SHA-256):
`d7213f16 18282cbc b4090abe 808a164f`
`9689062f 2b65b34f d86496b9 923200d7`

February 4, 2026

Abstract

This paper introduces SMAUG-T, a lattice-based post-quantum key exchange algorithm. SMAUG-T is designed by merging two prior algorithms SMAUG and TiGER. The algorithm is based on the hardness of the MLWE and MLWR problems defined in the module lattice and using sparse secret chosen by SMAUG.

Along with the original SMAUG parameter sets, we introduce a TiMER (Tiny SMAUG using Error Reconciliation) parameter set suitable for the IoT environment. In terms of size, SMAUG-T achieves ciphertext and public key that is up to 14% and 19% smaller than Kyber, respectively. From a performance perspective, encapsulation demonstrates high efficiency, achieving up to 60% faster than Kyber in the constant-time C implementation and up to 70% in the AVX2 implementation.

1 Introduction

SMAUG-T is an efficient post-quantum Key Encapsulation Mechanism (KEM) whose security is based on the hardness of the lattice problems. SMAUG-T follows the approaches using both Learning-With-Errors (LWE) and Learning-With-Roundings (LWR) variants in recent constructions of post-quantum KEMs such as Lizard [25] and RLizard [51]. Using the two lattice problems, SMAUG-T bases its security on their module variant problems as in Kyber [15] or Saber [32]: the public key does not leak the secret key information by the hardness of Modulue-LWE (MLWE) problem, and the ciphertext protects sharing keys based on the hardness of Modulue-LWR (MLWR) problem.

SMAUG-T consists of the underlying Public Key Encryption (PKE) scheme SMAUG-T.PKE and the KEM scheme SMAUG-T.KEM. SMAUG-T.PKE has INDistinguishability under Chosen Message Attack (IND-CPA), which can be converted to SMAUG-T.KEM scheme with INDistinguishability under adaptive Chosen Ciphertext Attack (IND-CCA2), through the Fujisaki–Okamoto (FO) transform.

1.1 Design rationale

The design rationale of SMAUG-T aims is to achieve small ciphertext and public key with low computational cost while maintaining security against various attacks. In more detail, we target the following practicality and security requirements considering real applications:

Practicality:

- Both the public key and ciphertext, especially the latter, which is transmitted more frequently, need to be short in order to minimize communication costs.
- As the key exchange protocol is frequently required on various personal devices, a KEM algorithm with low computational costs is more feasible than a high-cost one.
- A small secret key is desirable in restricted environments such as embedded or IoT devices since managing the secure zone is crucial to prevent physical attacks on secret key storage.

Security:

- Security should be concretely guaranteed concerning the attacks on the underlying assumptions, say lattice attacks.
- The low enough Decryption Failure Probability (DFP) is essential to avoid the attacks boosting the failure and exploiting the decryption failures [45, 29].
- As KEMs are widely used in various devices and systems, countermeasures against implementation-specific attacks should also be considered.

MLWE and MLWR.

SMAUG-T is constructed on the hardness of MLWE and MLWR problems and follow the key structure of Lizard [25] and Ring-Lizard (RLizard) [51]. Since LWE problem has been a well-studied problem for the last two decades, there are many LWE-based schemes (e.g., FrodoKEM [16]). Ring and module LWE problems (RLWE and MLWE) are variants defined over structured lattices and regarded as hard as LWE. Many schemes base their security on RLWE/MLWE (e.g., NewHope [4], Kyber [15] and Saber [32]) for efficiency reasons. We chose the module structure, which enables us to fine-tune security and efficiency in a much more scalable way, unlike standard and ring versions. Since MLWR problem is regarded as hard as MLWE problem unless we overuse the same secret to generate the samples [14], we chose to use MLWR samples for the encryption. By basing the MLWR, we reduce the ciphertext size by $\log q / \log p$ than MLWE instances so that more efficient encryption and decryption are possible.

Quantum Fujisaki–Okamoto transform.

SMAUG-T consists of key encapsulation mechanisms SMAUG-T.KEM, and public key encryption schemes SMAUG-T.PKE. On top of the PKE schemes, we construct the KEM schemes using the Fujisaki–Okamoto (FO) transform [37, 38]. Line of works on FO transforms in the quantum random oracle model [13, 43, 46, 55] make it possible to analyze the quantum security, i.e., in the Quantum Random Oracle Model (QROM). In particular, we use the FO transform with implicit rejection and no ciphertext contributions ($\mathsf{FO}_m^{\not\perp}$) following [44].

Sparse secret key and ephemeral secret.

We design the key generation algorithm based on MLWE problem using sparse secret. We use sparse ternary polynomials for the secret key and the ephemeral polynomial vectors based on the hardness reduction on the LWE problem using sparse secret [24]. We take advantage of the sparsity, e.g., significantly smaller secret keys. In particular, the small secret makes SMAUG-T more feasible in IoT devices having restricted resources. Specifically, we choose to use a fixed Hamming weight for the secret keys and a non-fixed Hamming weight for the ephemeral secret, a sparse version of the Centered Binomial Distribution (CBD), for secure implementation.

Choice of moduli.

All our parameter sets use powers of two moduli. This choice makes SMAUG-T enjoy faster encapsulation using simple bit shiftings, easy uniform samplings, and scalings. The power-of-two moduli makes it hard to apply the Number Theoretic Transform (NTT) on the polynomial multiplications. However, by embedding the power-of-two arithmetic into a larger NTT prime arithmetic, SMAUG-T achieves fast speeds.

Negligible decapsulation failures.

Since we base the security on the lattice problems, noise is inherent. Thus decryption result of a SMAUG-T.PKE ciphertext could be different from the original message. We balance the sizes, DFP, and security of SMAUG-T by fine-tuning the parameters while maintaining the DFP to be negligible. In addition, additional parameter set TiMER uses the D2 encoding and error reconciliation as in NewHope [4, 54], to further decrease the DFP and the sizes.

SMAUG-T.

We give estimated security and sizes for SMAUG-T parameter sets in Table 1, where the complete version of it can be found in Section 5.2. The sizes are given in bytes, and DFP is given logarithm base two.

Parameters sets	TiMER	SMAUG-T Mode 1	SMAUG-T Mode 3	SMAUG-T Mode 5
Target security	1	1	3	5
(n, k)	$(256, 2)$	$(256, 2)$	$(256, 3)$	$(256, 4)$
q	1024	1024	2048	2048
(p, p')	$(256, 8)$	$(256, 32)$	$(512, 16)$	$(512, 128)$
Classical core-SVP	119.7	119.7	180.2	250.1
Quantum core-SVP	105.4	105.4	158.6	221.0
Classical gates count	152.1	152.1	212.5	282.6
DFP	-161.0	-118.3	-179.2	-194.2
Secret key size	832	832	1312	1728
Public key	672	672	1088	1440
Ciphertext	608	672	992	1376

Table 1: Security and sizes for SMAUG-T parameter sets.

1.2 Advantages and limitations

Advantages.

The security of SMAUG-T relies on the hardness of the lattice problems MLWE and MLWR, which enable balancing between security and efficiency. In terms of sizes, SMAUG-T has smaller ciphertext sizes compared to Kyber or Saber, which is the smallest ciphertext size among the recent practical lattice-based KEMs. In terms of DFP, SMAUG-T achieves low enough DFP, which is less than or similar to that of Saber. SMAUG-T parameter sets do not use Error Correction Code (ECC) to avoid possible side-channel attacks, except for the TiMER parameter set. TiMER benefits from the single-bit error correcting D2 encoding, which is masking-friendly from its constructions. Implementation-wise, encapsulation and decapsulation of SMAUG-T can be done efficiently using NTT. Each sub-procedure are masking friendly, against the physical attacks. We give the constant-time C reference code and AVX optimization, which validates the completeness and efficiency of SMAUG-T.

Limitations.

We use MLWR problem, which has been studied shorter than MLWE or LWE problems; however, it has a security reduction to MLWE. MLWE problem with a sparse secret has a similar issue but has been studied much longer and is used in various applications, e.g., homomorphic encryptions. As we use MLWE problem for the secret key security, larger public key sizes than Saber are inherent. It can be seen as a trade-off between the public key size versus performance with a smaller secret key size.

2 Preliminaries

2.1 Notation

We denote matrices with bold and upper case letters (e.g., $\mathbf{A}$) and vectors with bold type and lower case letters (e.g., $\mathbf{b}$). Unless otherwise stated, the vector is a column vector.

We define a polynomial ring $\mathcal{R} = \mathbb{Z}[x]/(x^n + 1)$ where n is a power of 2 integers and denote a quotient ring by $\mathcal{R}_q = \mathbb{Z}[x]/(q, x^n + 1) = \mathbb{Z}_q[x]/(x^n + 1)$ for a positive integer q.

For an integer η, we denote the set of polynomials of degree less than n with coefficients in $[-\eta, \eta] \cap \mathbb{Z}$ as S_η. Let $\tilde{S}_\eta$ be a set of polynomials of degree less than n with coefficients in $[-\eta, \eta) \cap \mathbb{Z}$. We denote a uniform distribution over a discrete set C as $U(C)$. We denote a zero-centered discrete Gaussian distribution with standard deviation σ as $\mathcal{D}_{\mathbb{Z},\sigma}$. We define Rényi divergence of order α between two probability distributions P and Q such that $\mathrm{Supp}(P) \subseteq \mathrm{Supp}(Q)$ as

$$R_\alpha(P\|Q) = \left(\sum_{x \in \mathrm{Supp}(P)} \frac{P(x)^\alpha}{Q(x)^{\alpha-1}} \right)^{1/(\alpha-1)},$$

where $\mathrm{Supp}(D)$ for a distribution D is defined as $\mathrm{Supp}(D) = \{x \in D : D(x) \neq 0\}$. We denote a binomial distribution with a parameter n and a probability p as $B(n, p)$. We denote the Centered Binomial Distribution (CBD) with a parameter d as CBD_d, where the samples range from $-d$ to d.

2.2 Lattice assumptions

We define some well-known lattice assumptions MLWE and MLWR on the structured Euclidean lattices.

Definition 1 (Decision-MLWE$_{n,q,k,\ell,\eta}$)**.** For positive integers q, k, ℓ, η and the dimension n of $\mathcal{R}$, we say that the advantage of an adversary $\mathcal{A}$ solving the decision-MLWE$_{n,q,k,\ell,\eta}$ problem is

$$\begin{aligned}\mathsf{Adv}^{\mathrm{MLWE}}_{n,q,k,\ell,\eta}(\mathcal{A}) = \Big| &\Pr\left[b = 1 \mid \mathbf{A} \leftarrow \mathcal{R}_q^{k\times\ell}; \mathbf{b} \leftarrow \mathcal{R}_q^k; b \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{b})\right] \\ &- \Pr\left[b = 1 \mid \mathbf{A} \leftarrow \mathcal{R}_q^{k\times\ell}; (\mathbf{s}, \mathbf{e}) \leftarrow S_\eta^\ell \times S_\eta^k; b \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{A}\cdot\mathbf{s} + \mathbf{e})\right] \Big|\end{aligned}$$

Definition 2 (Decision-MLWR$_{n,p,q,k,\ell,\eta}$)**.** For positive integers p, q, k, ℓ, η with $q \geq p \geq 2$ and the dimension n of $\mathcal{R}$, we say that the advantage of an adversary $\mathcal{A}$ solving the decision-MLWR$_{n,p,q,k,\ell,\eta}$ problem is

$$\begin{aligned}\mathsf{Adv}^{\mathrm{MLWR}}_{n,p,q,k,\ell,\eta}(\mathcal{A}) = \Big| &\Pr\left[b = 1 \mid \mathbf{A} \leftarrow \mathcal{R}_p^{k\times\ell}; \mathbf{b} \leftarrow \mathcal{R}_q^k; b \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{b})\right] \\ &- \Pr\left[b = 1 \mid \mathbf{A} \leftarrow \mathcal{R}_q^{k\times\ell}; \mathbf{s} \leftarrow S_\eta^\ell; b \leftarrow \mathcal{A}(\mathbf{A}, \lfloor p/q \cdot \mathbf{A}\cdot\mathbf{s}\rceil)\right] \Big|\end{aligned}$$

2.3 Public key encryption and key encapsulation mechanism

We recap the formalisms of PKE and KEM.

Definition 3 (PKE)**.** A *public key encryption* scheme is a tuple of PPT algorithms $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with the following specifications:

- KeyGen: a probabilistic algorithm that outputs a public key pk and a secret key sk;
- Enc: a probabilistic algorithm that takes as input a public key pk and a message μ and outputs a ciphertext ct;
- Dec: a deterministic algorithm that takes as input a secret key sk and a ciphertext ct and outputs a message μ.

Let $0 < \delta < 1$. We say that it is $(1-\delta)$*-correct* if for any $(\mathsf{pk}, \mathsf{sk})$ generated from KeyGen and μ,

$$Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, \mu)) \neq \mu] \leq \delta,$$

where the probability is taken over the randomness of the encryption algorithm. We call the above probability *decryption failure probability (DFP)*. In addition, we say that it is *correct in the (Q)ROM* if the probability is taken over the randomness of the (quantum) random oracle, modeling the hash function.

Definition 4 (KEM)**.** A *key encapsulation mechanism* scheme is a tuple of PPT algorithms $\mathsf{KEM} = (\mathsf{KeyGen}, \mathsf{Encap}, \mathsf{Decap})$ with the following specifications:

- KeyGen: a probabilistic algorithm that outputs a public key pk and a secret key sk;
- Encap: a probabilistic algorithm that takes as input a public key pk and outputs a sharing key K and a ciphertext ct;
- Decap: a deterministic algorithm that takes input a secret key sk and a ciphertext ct and outputs a sharing key K.

The *correctness* of KEM is defined similarly to that of PKE.

We give the advantage function for a $\mathsf{IND\text{-}CPA}$ attacker against PKE.

Definition 5 (IND-CPA security of PKE)**.** For a (quantum) adversary $\mathcal{A}$ against a public key encryption scheme $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, we define the $\mathsf{IND\text{-}CPA}$ *advantage of* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ as follows:

$$\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A}) = \left| \Pr_{(\mathsf{pk},\mathsf{sk})}\left[b = b' \;\middle|\; \begin{array}{c} (\mu_0, \mu_1, \mathrm{st}) \leftarrow \mathcal{A}_1(\mathsf{pk});\; b \leftarrow \{0,1\}; \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mu_b);\; b' \leftarrow \mathcal{A}_2(\mathsf{pk}, \mathsf{ct}, \mathrm{st}) \end{array} \right] - \frac{1}{2} \right|.$$

The probability is taken over the randomness of $\mathcal{A}$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.

We then define two advantage functions for $\mathsf{IND\text{-}CPA}$ and $\mathsf{IND\text{-}CCA2}$ attackers.

Definition 6 (IND-CPA and IND-CCA security of KEM)**.** For a (quantum) adversary $\mathcal{A}$ against a key encapsulation mechanism $\mathsf{KEM} = (\mathsf{KeyGen}, \mathsf{Encap}, \mathsf{Decap})$, we define the $\mathsf{IND\text{-}CPA}$ *advantage of* $\mathcal{A}$ as follows:

$$\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{KEM}}(\mathcal{A}) = \left| \Pr_{(\mathsf{pk},\mathsf{sk})}\left[b = b' \;\middle|\; \begin{array}{c} b \leftarrow \{0,1\};\; (K_0, \mathsf{ct}) \leftarrow \mathsf{Encap}(\mathsf{pk}); \\ K_1 \leftarrow \mathcal{K};\; b' \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{ct}, K_b) \end{array} \right] - \frac{1}{2} \right|.$$

The probability is taken over the randomness of $\mathcal{A}$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$. The $\mathsf{IND\text{-}CCA}$ *advantage of* $\mathcal{A}$ is defined similarly except that the adversary can query $\mathsf{Decap}(\mathsf{sk}, \cdot)$ oracle on any ciphertext $\mathsf{ct}'(\neq \mathsf{ct})$.

We can then define the (quantum) security notions of PKE and KEM in the (Q)ROM as follows.

Definition 7 ((Q)ROM security of PKE and KEM)**.** For $T, \epsilon > 0$, we say that a scheme $\mathcal{S} \in \{\mathsf{PKE}, \mathsf{KEM}\}$ is (T, ϵ)-ATK *secure in the (Q)ROM* if for any (quantum) adversary $\mathcal{A}$ with runtime $\leq T$ given classical access to O and (quantum) access to a random oracle H, it holds that $\mathsf{Adv}_{\mathcal{S}}^{\mathsf{ATK}}(\mathcal{A}) < \epsilon$, where

$$O = \begin{cases} \mathsf{Enc} & \text{if } \mathcal{S} = \mathsf{PKE} \text{ and } \mathsf{ATK} \in \{\mathsf{OW\text{-}CPA}, \mathsf{IND\text{-}CPA}\}, \\ \mathsf{Encap} & \text{if } \mathcal{S} = \mathsf{KEM} \text{ and } \mathsf{ATK} = \mathsf{IND\text{-}CPA}, \\ \mathsf{Encap}, \mathsf{Decap}(\mathsf{sk}, \cdot) & \text{if } \mathcal{S} = \mathsf{KEM} \text{ and } \mathsf{ATK} = \mathsf{IND\text{-}CCA}. \end{cases}$$

2.4 Fujisaki–Okamoto transform

Fujiskai and Okamoto proposed a novel generic transform [37, 38] that turns a weakly secure PKE scheme into a strongly secure PKE scheme in the Random Oracle Model (ROM), and various variants have been proposed to deal with tightness, non-correct PKEs, and in the quantum setting, i.e., QROM. Here, we recall the FO transformation for KEM as introduced by Dent [31] and revisited by Hofheinz et al. [43], Bindel et al. [12], and Hövelmanns et al. [44].

The original FO transforms $\mathsf{FO}_m^{\perp}$ constructs a KEM from a deterministic PKE, i.e., a de-randomized version. The encapsulation randomly samples a message m and uses the message's hash value $G(m)$ as randomness for encryption, generating a ciphertext. The sharing key $K = H(m)$ is generated by hashing (with different hash functions) the message. In the decapsulation, it first decrypts the ciphertext and recovers the message, m'. If it fails to decrypt, it outputs $\perp$. If the "re-encryption" of the recovered message is not equal to the received ciphertext, it also outputs $\perp$. The sharing key can be generated by hashing the recovered message.

In the quantum setting, however, the FO transform with "implicit rejection" ($\mathsf{FO}_m^{\not\perp}$) has a tighter security proof than the original version, which implicitly outputs a pseudo-random sharing key if the re-encryption fails. We recap the QROM proof of Bindel et al. [12] allowing the KEMs constructed over non-perfect PKEs to have $\mathsf{IND\text{-}CCA}$ security.

Theorem 8 ([12], Theorem 1 & 2)**.** *Let G and H be quantum-accessible random oracles, and the deterministic PKE is ϵ-injective. Then the advantage of* $\mathsf{IND\text{-}CCA}$ *attacker $\mathcal{A}$ with at most Q_{Dec} decryption queries and Q_G and Q_H hash queries at depth at most d_G and d_H, respectively, is*

$$\mathsf{Adv}_{\mathsf{KEM}}^{\mathsf{IND\text{-}CCA}}(\mathcal{A}) \leq 2\sqrt{(d_G+2)\left(\mathsf{Adv}_{\mathsf{PKE}}^{\mathsf{IND\text{-}CPA}}(\mathcal{B}_1) + 8(Q_G+1)/|\mathcal{M}|\right)} \\ + \mathsf{Adv}_{\mathsf{PKE}}^{DF}(\mathcal{B}_2) + 4\sqrt{d_H Q_H/|\mathcal{M}|} + \epsilon,$$

where $\mathcal{B}_1$ is an $\mathsf{IND\text{-}CPA}$ *adversary on PKE and $\mathcal{B}_2$ is an adversary against finding a decryption failing ciphertext,* returning at most Q_{Dec} ciphertexts.

3 Design choices

In this section, we explain the design choices for SMAUG-T.

3.1 MLWE public key and MLWR ciphertext

One of the core designs of SMAUG-T uses the MLWE hardness for its secret key security and MLWR hardness for its message security. This choice is adapted from Lizard and RLizard, which use LWE/LWR and RLWE/RLWR, respectively. Using both LWE and LWR variant problems makes the conceptual security distinction between the secret key and the ephemeral sharing key: a more conservative secret key with more efficient en/decapsulations. This can be viewed as a trade-off between "conservative" and "efficient" designs. Combined with the sparse secret, bringing the LWE-based key generation to the LWR-based scheme enables balancing the speed and the DFP.

3.1.1 Public key

Public key of SMAUG-T consists of a vector $\mathbf{b}$ over a polynomial ring $\mathcal{R}_q$ and a matrix $\mathbf{A}$, which can be viewed as an MLWE sample,

$$(\mathbf{A}, \mathbf{b} = -\mathbf{A}^\top \mathbf{s} + \mathbf{e}) \in \mathcal{R}_q^{k\times k} \times \mathcal{R}_q^k,$$

where $\mathbf{s}$ is a ternary secret polynomial with hamming weight h_s, and $\mathbf{e}$ is an error sampled from discrete Gaussian distribution with standard deviation σ. We now specify the uniform matrix sampling algorithm for $\mathbf{A} \in \mathcal{R}_q^{k\times k}$ in Figure 1. It is adapted from the pseudorandom generator gen in Saber [30].

```
expandA(seed):                                        ▷ seed ∈ {0, 1}^256
  buf ← XOF(seed)
  for i from 0 to k − 1 do
      A[i] = bytes_to_Rq(buf + polybytes · i)         ▷ Convert to ring elements
  return A
```

Figure 1: Uniform random matrix sampler, expandA.

We note that the public key of SMAUG-T consists of $\mathbf{b}$ and the seed of $\mathbf{A}$.

3.1.2 Ciphertext

The ciphertext of SMAUG-T is a tuple of a vector $\mathbf{c}_1 \in \mathcal{R}_p^k$ and a polynomial $c_2 \in \mathcal{R}_{p'}$. The ciphertext is generated by multiplying a random vector $\mathbf{r}$ to the public key; then it is scaled and rounded as,

$$\mathbf{c} = \begin{bmatrix} \mathbf{c}_1 \\ c_2 \end{bmatrix} = \left\lfloor \frac{p}{q} \cdot \begin{pmatrix} \mathbf{A} \\ \mathbf{b}^\top \end{pmatrix} \cdot \mathbf{r} \right\rceil + \frac{p}{t} \cdot \begin{bmatrix} 0 \\ \mu \end{bmatrix},$$

Along with the public key, it can be treated as an MLWR sample added by a scaled message as $(\mathbf{A}', \lfloor p/q \cdot \mathbf{A}' \cdot \mathbf{r} \rceil) + (0, \mu')$, where $\mathbf{A}'$ is a concatenated matrix of $\mathbf{A}$ and $\mathbf{b}^\top$.

The ciphertext can be further compressed by scaling the second component c_2 by p'/p, resulting in a shorter ciphertext but a larger error. We note that the public key can be compressed with the same technique. However, it introduces a more significant error, so we do not compress the public key in SMAUG-T.

We call the random vector $\mathbf{r}$ the ephemeral secret, which is a sparse ternary vector. Note that the secret and the ephemeral secret are both sparse ternary vectors; however, we sample them from different distributions using different samplers.

3.2 Sparse secret

We use the sparse ternary distribution for the randomnesses, $\mathbf{s}$ and $\mathbf{r}$. In the following, we will discuss the advantages of the sparse secret and give the sampling algorithm. Notably, we use two different sampling algorithms for the sparse secrets: HWT, a fixed Hamming weight sampler for the secret key $\mathbf{s}$, and spCBD, a non-fixed Hamming weight sampler for ephemeral secret $\mathbf{r}$, respectively.

3.2.1 Advantage of using sparse secret

The sparse secret is widely used in homomorphic encryption to reduce the noise propagation during the homomorphic operations [40, 23, 18] and to speed up the computations. As the lattice-based KEM schemes have inherent decryption error from LWE or LWR noise, the sparse secret can lower this decryption error and improve the performance of KEMs.

Concretely, the decryption error can be expressed as $\langle \mathbf{e}, \mathbf{r} \rangle + \langle \mathbf{e}_1, \mathbf{s} \rangle + e_2$, where $\mathbf{s}$ is a secret key, $\mathbf{r}$ is a randomness used for encryption, $\mathbf{e} \leftarrow \chi_{pk}^{k}$ is a noise added in public key, and $(\mathbf{e}_1, e_2) \leftarrow \chi_{ct}^{k+1}$ is a noise added in ciphertext. As the vectors $\mathbf{r}$ and $\mathbf{s}$ are ternary, each coefficient of the decryption error is a signed addition of h_r variables from χ_{pk} and $h_s + 1$ variables from χ_{ct}. The magnitude of the decryption error depends greatly on the Hamming weights h_r and h_s; thus, we can take advantage of the sparse secrets.

On the other hand, as the sparse secret reduces the secret key entropy, the hardness of the lattice problem may be decreased. For the security of LWE problem using sparse secret, a series of works have been done, including [24] for asymptotic security based on the reductions to worst-case lattice problems, and [58, 34, 11] for concrete security. Independent of the secret distribution, the module variant (MLWE) is regarded as hard as LWE problem with appropriate parameters, including a smaller modulus. We also exploit the reductions from ordinary MLWE to MLWE using sparse secret or small errors [19]. The MLWR problem also has a simple reduction from MLWE independent of the secret distribution, and its concrete security is heuristically discussed in [30].

Since SMAUG-T uses a sparse secret key $\mathbf{s}$ and a sparse randomness $\mathbf{r}$, the security of SMAUG-T is based on the hardness of MLWE and MLWR problems using sparse secret. For the specific parameters, we exploit the lattice-estimator [1], which covers most of the recent lattice attacks, and also consider some attacks not included in the estimator. Using a smaller modulus, SMAUG-T can maintain high security, as in Kyber or Saber.

3.2.2 Hamming weight sampler

Our Hamming weight sampler, HWT_h, is a shuffling-based algorithm that originated from [35], which has no bias on its output and can be realized in a constant-time implementation. This algorithm outperforms other constant-time samplers, such as the sorting-based one or the bounded-rejection-based one. We first describe a subroutine of the shuffling-based sampler in Figure 2, which generates unbiased random integers. An array of integers **si** from an input seed array where $\mathbf{si}_i$ is uniformly sampled from integers $[0, 1, ..., n - i]$ without bias.

<u>REJ_SAMPLE_MOD(rand):</u> ▷ rand is an array of 16-bit integers

1: $j = n$
2: $\mathsf{t} = t_0, t_1, ..., t_{n-1} = 0, 0, ..., 0$
3: **for** i from 0 to $n - 1$ **do**
4:     $w = 2^L \mod (n - i)$
5:     $m = \mathsf{rand}_i \cdot (n - i)$
6:     $l = m \mod 2^L$
7:     **while** $l < w$ **do**
8:         $m = \mathsf{rand}_j \cdot (n - i), \quad j = j + 1$
9:         $l = m$
10:     $\mathsf{t}_i = m \gg L$
11: **return** t

Figure 2: Algorithm for generating unbiased uniformly random integers

<u>HWT_h(seed):</u>

1: $\mathsf{v} = v_0, v_1, ..., v_{n-1} = 0, 0, ..., 0$
2: buf, sign= PRF(seed)
3: si = REJ_SAMPLE_MOD(buf)
4: $c_0 = n - h$
5: **for** i from 0 to $n - 1$ **do** ▷ Binary fixed-weight sampling
6:     $t = (\mathsf{si}_i - c_0) \gg 15$
7:     $c_0 = c_0 + t_0, \quad \mathsf{v}_i = 1 + t$
8: **for** i from 0 to $n - 1$ **do** ▷ Transform to ternary
9:     $\mathsf{v}_i = (-\mathsf{v}_i) \wedge ((\mathsf{sign}_i \wedge \mathtt{0x02}) - 1)$
10: **return** v

Figure 3: Ternary fixed Hamming weight sampling by shuffling

Then, we introduce the Hamming weight sampler algorithm in Figure 3, which performs a ternary fixed-weight sampling by shuffling, which is slightly modified from [35]. This process generates binary fixed-weight as stated in lines 5 to 11, then transforms to ternary representation using the random bits in sign generated in line 3.

3.2.3 Sparse CBD sampler

Inspired by the efficient CBD sampler from New Hope [4] (and many other KEM schemes) and the approximate discrete Gaussian sampler from SMAUG [22], we introduce a boolean-based efficient sparse ternary sampler, which we call sparse CBD sampler (spCBD).

The spCBD sampler takes input a probability $r < 1/2$ and outputs a signed bit from $\{-1, 0, 1\}$ with a probability mass function $f : \{-1, 0, 1\} \rightarrow [0, 1]$ given as

$$f(-1) = f(1) = r, \; f(0) = 1 - 2r \; .$$

This can be naturally extended to a vector of signed bits, where each coordinate follows the same distribution. The resulting vector is a sparse ternary vector. However, as each coordinate is probabilistically sampled, the vector's Hamming weight is not a fixed value. The Hamming follows the binomial distribution as

$$h \sim B(n, 2r) \; ,$$

where h is the Hamming weight and n is the length of the vector.

We remark that, especially when the denominator of the probability r is a power-of-two integer, say 2^k, the spCBD sampler can be efficiently instantiated by sampling $k + 1$ random bits and applying only the boolean operations. We present two spCBD samplers we will use, which randomly sample from the spCBD distribution with the probability parameter $r = 1/8$ and $r = 3/16$ in Figures 4.

$\mathsf{spCBD}_{1/8}$:

1: $a, b, c \leftarrow \{0, 1\}$
2: $t \leftarrow a \wedge b$
3: $\mathsf{sign} \leftarrow ((c \ll 1) \wedge \mathsf{0x02}) - 1$
4: **return** $t \cdot \mathsf{sign}$ ▹ Distribution: $\{-1\colon 1/8, 0\colon 3/4, 1\colon 1/8\}$

$\mathsf{spCBD}_{3/16}$:

1: $a, b, c, d \leftarrow \{0, 1\}$
2: $t \leftarrow a \wedge b \vee c$
3: $\mathsf{sign} \leftarrow ((d \ll 1) \wedge \mathsf{0x02}) - 1$
4: **return** $t \cdot \mathsf{sign}$ ▹ Distribution: $\{-1\colon 3/16, 0\colon 5/8, 1\colon 3/16\}$

Figure 4: Sparse CBD sampler for $r = 1/8$ and $3/16$.

We further note that the distribution $\mathsf{spCBD}_{1/4}$ is equal to CBD with a parameter $d = 1$, i.e., CBD_1, which outputs a ternary secret.

3.3 Discrete Gaussian noise

3.3.1 Using approximate discrete Gaussian noise

Our design choice for the noise distribution in MLWE follows the conventional discrete Gaussian distribution, but with approximated CDTs following the approaches in

FrodoKEM [16]. As a result, we use a discrete Gaussian noise for the public key generation, which is approximated to a narrow distribution. As this approximated discrete Gaussian noise is used only for the public key, we can efficiently bound the security loss from above. Considering the narrow discrete Gaussian noise, we give a theoretical justification based on Rényi divergence to guarantee the security of SMAUG-T.

In SMAUG-T, the narrow discrete Gaussian noise is used only for the public key generation. So, the difference in the noise distribution only affects the distinguishing advantage between the games G_2 and G_3 in the proof of Theorem 11. Then, the bound for the distinguishing advantage can also be expressed as

$$\left(\mathsf{Adv}^{\mathsf{MLWE}}_{n,q,k,k,\mathcal{D}_{\mathbb{Z},\sigma}}(\mathcal{B}_2) \cdot R_\alpha(\mathsf{dGaussian}_\sigma \| \mathcal{D}_{\mathbb{Z},\sigma})^{nk}\right)^{1-1/\alpha},$$

assuming the pseudorandomness of $\mathsf{dGaussian}_\sigma$. This is due to Lemma 5.5 in [3]. We note that the key generation calls dGaussian only nk times and that the public key is generated only once.

The advantage bound for SMAUG-T parameter set (see Section 5.2) can be computed directly using the given formula; for TiMER parameter set (Resp. SMAUG-T128, 192, and 256), the advantage increases by 1.09 (Resp. 1.09, 1.64, and 2.20) bits with $\alpha = 500$. Opposed to the estimated security based on the bound $\mathsf{Adv}^{\mathsf{MLWE}}_{n,q,k,k,\mathsf{dGaussian}_\sigma}(\mathcal{B}_2)$ given in Section 5.2, this new bound provides a more conservative security preventing some possible future attacks that target the noise distribution.

This modification will slightly decrease only the speed of key generation by less than 1.1x.

We also note that the narrow Gaussian noise is already considered when estimating the concrete security (given in Section 5.2) using the explained estimators. The analysis here provides a more conservative security, preventing possible future attacks that target the noise distribution. We also note that in the core-SVP methodology, we only focus on the estimated attack cost of the underlying MLWE and MLWR problems, not based on the security reductions (as done in most of the NIST-submitted schemes) for a fair comparison to Kyber.

3.3.2 dGaussian sampler

We construct dGaussian, a constant-time approximate discrete Gaussian noise sampler, upon a Cumulative Distribution Table (CDT) but is not used during sampling, as it is expressed with bit operations.

We first scale the discrete Gaussian distribution and make a CDT approximating the discrete Gaussian distribution. We choose an appropriate scaling factor based on the analysis in [49, 16] using Rényi divergence. We then deploy the Quine–McCluskey method[1] and apply logic minimization technique on the CDT. As a result, even though our dGaussian is constructed upon CDT, it is expressed with bit operations and is constant-time.

We describe dGaussian algorithm with $\sigma = 1.0625$ in Figure 5. The algorithm is easily parallelizable and suitable for IoT devices as their memory requirement is low.

[1] We use the python package, from `https://github.com/dreylago/logicmin`.

$\mathsf{dGaussian}_{1.0625}(x)$:

Require: $x = x_0x_1x_2x_3x_4x_5x_6x_7x_8x_9 \in \{0,1\}^{10}$

$s = s_1s_0 = 00 \in \{0,1\}^2$

$s_0 = x_0x_1x_2x_3x_4x_5x_7\overline{x_8}$

$s_0 \mathrel{+}= (x_0x_3x_4x_5x_6x_8) + (x_1x_3x_4x_5x_6x_8) + (x_2x_3x_4x_5x_6x_8)$

$s_0 \mathrel{+}= (\overline{x_2x_3x_6}x_8) + (\overline{x_1x_3x_6}x_8)$

$s_0 \mathrel{+}= (x_6x_7\overline{x_8}) + (\overline{x_5x_6}x_8) + (\overline{x_4x_6}x_8) + (\overline{x_7}x_8)$

$s_1 = (x_1x_2x_4x_5x_7x_8) + (x_3x_4x_5x_7x_8) + (x_6x_7x_8)$

$s = (-1)^{x_9} \cdot s$ ▷ $\cdot$ is the arithmetic multiplication

return s

Figure 5: Discrete Gaussian sampler with $\sigma = 1.0625$, $\mathsf{dGaussian}_\sigma$.

3.4 Polynomial Multiplication

Despite the sparsity of the secret keys in SMAUG-T, a naive approach to take advantage of the sparsity may expose the scheme to a side-channel attack that exploits the time-variant of executions for polynomial multiplication. Well-known multiplication algorithms that can be implemented with constant-time are the Number Theoretic Transform (NTT) and Toom–Cook multiplication.

The moduli of SMAUG-T are all power-of-two integers to efficiently handle rounding by bit shifting and result in non-biased rounding error. To adopt NTT for multiplications in SMAUG-T, the polynomial should be transformed to NTT-friendly ring by switching the modulus. Conversely, the Toom–Cook multiplication is well-suited for handling arbitrary polynomial rings, as its foundation lies in a divide-and-conquer strategy that reduces the problem into smaller sub-problems. This approach ultimately relies on classical polynomial multiplication techniques (i.e., schoolbook multiplication) for base cases of sufficiently small size. These multiplications are commonly used in lattice-based PQC schemes, and the performance of these two algorithms varies depending on the degree of the polynomials, the algorithm's parameters, and the operating hardware architecture.

Toom–Cook and Karatsuba

The Toom–Cook [27, 59] and the Karatsuba [47] multiplications are efficient algorithms for large integers that split operands and perform multiplications and additions on smaller parts, resulting in lower time complexity. Both achieve sub-quadratic time complexity $O(n^{1+e})$ in the bit-length n where $0 < e < 1$, and can be utilized for the multiplications of polynomials of large degrees.

The Karatsuba multiplication for computing $c(x) = a(x)b(x)$ divides each of degree n polynomials $a(X)$ and $b(x)$ into two sub-polynomials of degree $\frac{n}{2}$. For instance, $a(x)$ is split into $a(x) = a^0(x) + a^1(x)x^{\frac{n}{2}}$, where $a^0(x)$ and $a^1(x)$ are defined as

$$\begin{aligned}
a^0(x) &= a_0 + a_1x + a_2x^2 + \cdots + a_{\frac{n}{2}-1}x^{\frac{n}{2}-1} \\
a^1(x) &= a_{\frac{n}{2}} + a_{\frac{n}{2}+1}x + a_{\frac{n}{2}+2}x^{\frac{n}{2}+2} + \cdots + a_{n-1}x^{\frac{n}{2}-1}.
\end{aligned}$$

The Karatsuba multiplication computes $c(x)$ with three $\frac{n}{2}$-degree multiplications and some additions rather than 4 polynomial multiplications as follows:

$$\begin{aligned}c(x) &= a^0(x)b^0(x)\\&\quad + \left(\left(a^0(x)+a^1(x)\right)\left(b^0(x)+b^1(x)\right) - \left(a^0(x)+b^0(x)\right)\left(a^1(x)+b^1(x)\right)\right)x^{n/2}\\&\quad + a^1(x)b^1(x)x^n.\end{aligned}$$

The nested polynomial multiplications can be handled by applying Karatsuba multiplication recursively until the degree of input polynomials are sufficiently small to be executed by the naive multiplication method yielding $\Theta(n^{\log_2 3})$ time complexity by the master theorem for divide-and-conquer recurrences.

Toom–Cook multiplication generalizes Karatsuba multiplication in a way that it splits the degree-n polynomials into k sub-polynomials and handles degree-$\frac{n}{k}$ polynomials in an appropriate manner. It is also possible to compute the multiplications of the sub-polynomials by Karatsuba multiplication. The time complexity of k-way Toom–Cook multiplication is $\Theta(n^{\log_k(2k-1)})$ by the master theorem.

Due to its ability to efficiently handle multiplications on polynomial rings that are not well-suited for NTT, several PQC algorithms, such as those in [33], [41], adopt Toom–Cook multiplication. As in [10], a 256-degree polynomial in SMAUG-T is split into $k = 4$ parts by Toom–Cook, requiring 7 multiplications of 64-degree sub-polynomials. The sub-polynomials are further split by Karatsuba with threshold degree 16. With this choice of k and the threshold degree for Karatsuba, 256-degree polynomial multiplication requires 63 polynomial multiplications for degree 16.

Number Theoretic Transform

The NTT is a widely used method for efficient multiplication in lattice-based cryptography that uses polynomial rings since its quasi-linear complexity $O(n \cdot logn)$. For two polynomials $a(x), b(x) \in \mathcal{R}_q$ the product $a(x) \cdot b(x)$ can be computed as follows where NTT^{-1} denotes the inverse of NTT and $\circ$ is an element-wise multiplication in $\mathbb{Z}_q$.

$$\mathsf{NTT}^{-1}(\mathsf{NTT}(a(x)) \circ \mathsf{NTT}(b(x))).$$

However, NTT has a limitation that requires using an NTT-friendly ring. Specifically, the parameter n should be a power-of-two integer, then the modulus q must be a prime that is 1 modulo $2n$ to ensure that $\mathbb{Z}_q$ contains primitive n-th or $2n$-th root of unity. Despite its efficiency advantages, some schemes are unable to leverage the NTT due to their use of NTT-unfriendly rings, such as those employing power-of-two modulus or a prime n. Notable examples of such schemes include Saber, NTRU, and SMAUG-T.

On the other hand, it is still possible to use NTT for the polynomial arithmetic in these schemes by embedding the modulus into the NTT prime ring. [26] shows that this approach is more efficient for Saber and NTRU than Toom–Cook in SIMD environments such as AVX2. An efficient AVX2 implementation using this approach is also feasible for SMAUG-T. For the modulus q such that $n \nmid (q-1)$, we represent coefficients within the range $[-\frac{q}{2}, \frac{q}{2})$. Considering that the maximum value of multiplication result coefficient is $n \cdot q^2/4$, if we find an NTT prime Q satisfying $Q > n \cdot q^2/2$ and $n|(Q-1)$, we can use NTT-based multiplication in $\mathcal{R}_Q$ and recover correct result in $\mathcal{R}_q$. If Q becomes too large, it may exceed the 16-bit data type typically used for coefficient. In such cases, by using the Chinese Remainder Theorem (CRT), the product of multiple NTT primes q_i can be used as sufficiently large Q. For SMAUG-T, $q_0 = 7{,}681$ and $q_1 = 10{,}753$ can be used as NTT primes.

3.5 FO transform, $\mathsf{FO}_m^{\not\perp}$

We construct SMAUG-T upon the FO transform with implicit rejection and without ciphertext contribution to the sharing key generation, say $\mathsf{FO}_m^{\not\perp}$. This choice makes the encapsulation and decapsulation algorithm efficient since the sharing key can be directly generated from a message. The public key is additionally fed into the hash function with the message to avoid multi-target decryption failure attacks. The IND-CCA security of the resulting KEM in the QROM is well-studied in [43, 44, 12].

3.6 D2 encoding

An additional parameter, TiMER, uses D2 encoding. D2 is one of the reconciliation techniques that reduces bandwidth requirements, which was used in NewHope [4]. D2 lowers the decryption failure rate and reduces the ciphertext size by changing the error bound. In Figure 6, we give the description of D2.

$\mathsf{D2Enc}(\mu \in \{0,\ldots,255\}^{16})$:

1: $\mathsf{v} \leftarrow \mathcal{R}_q$
2: **for** i from 0 to 15 **do**
3:     **for** j from 0 to 7 **do**
4:         $\mathsf{mask} \leftarrow ((\mu[i] \gg j)\ \&\ 1$
5:         $v_{8*i+j+0} \leftarrow \mathsf{mask}\ \&\ (q/2)$
6:         $v_{8*i+j+128} \leftarrow \mathsf{mask}\ \&\ (q/2)$
7: **return** $\mathsf{v} \in \mathcal{R}_q$

$\mathsf{D2Dec}(\mathsf{v} \in \mathcal{R}_q)$:

1: $\mu \leftarrow \{0,\ldots,255\}^{16}$
2: **for** i from 0 to 255 **do**
3:     $t \leftarrow |(v_{i+0} \bmod q) - (q-1)/2|$
4:     $t \leftarrow t + |(v_{i+128} \bmod q) - (q-1)/2|$
5:     $t \leftarrow t - q/2$
6:     $t \leftarrow t \gg 15$
7:     $\mu[i \gg 3] \leftarrow \mu[i \gg 3] | (t \ll (i\ \&\ 7))$
8: **return** $\mu \in \{0,\ldots,255\}^{16}$

Figure 6: Description of D2 encoding

To ensure robustness against errors, each bit of the 128-bit message $\mu \in \{0,\ldots,255\}^{16}$ is encoded into 2 coefficients by D2Enc. The decoding function D2Dec maps 2 coefficients back to the original key bit. For example, for $n = 256$, take 2 coefficients (each in the range $\{0,\ldots,q-1\}$), subtract $q/2$ from each of them, accumulate their absolute values, and set the key bit to 0 if the sum is larger than $q/2$ or to 1 otherwise.

4 The SMAUG-T

4.1 Specification of SMAUG-T.PKE

We now describe the public key encryption scheme SMAUG-T.PKE in Figure 7 with the following building blocks:

- Pseudo random function PRF for generating $\mathsf{seed}_{\mathbf{A}}$, $\mathsf{seed}_{\mathsf{sk}}$, and $\mathsf{seed}_{\mathbf{e}}$,
- Uniform random matrix sampler expandA for deriving $\mathbf{A}$ from $\mathsf{seed}_{\mathbf{A}}$,
- Discrete Gaussian sampler $\mathsf{dGaussian}_\sigma$ for deriving a MLWE noise $\mathbf{e}$ with standard deviation σ from $\mathsf{seed}_{\mathbf{e}}$,
- Hamming weight sampler HWT_h for deriving a sparse ternary $\mathbf{s}$ with Hamming weight $h = h_s$ from $\mathsf{seed}_{\mathsf{sk}}$
- Sparse CBD sampler spCBD_r for deriving a sparse ternary $\mathbf{r}$ with a probability parameter r from $\mathsf{seed}_{\mathbf{r}}$

$\underline{\mathsf{KeyGen}(1^\lambda)}$:

1: $\mathbf{A} \leftarrow \mathcal{R}_q^{k\times k}$
2: $\mathbf{s} \leftarrow \mathsf{HWT}_{h_s} \in S_\eta^k$
3: $\mathbf{e} \leftarrow \tilde{D}_\sigma \in \mathcal{R}^k$
4: $\mathbf{b} = -\mathbf{A}^\top \cdot \mathbf{s} + \mathbf{e} \in \mathcal{R}_q^k$
5: **return** $\mathsf{pk} = (\mathbf{A}, \mathbf{b})$, $\mathsf{sk} = \mathbf{s}$

$\underline{\mathsf{Enc}(\mathsf{pk}, \mu)}$: ▷ $\mathsf{pk} = (\mathbf{A}, \mathbf{b})$, $\mu \in \mathcal{R}_t$

1: $\mathbf{r} \leftarrow \mathsf{spCBD}_r \in S_\eta^k$
2: $\mathbf{c}_1 = \lfloor p/q \cdot \mathbf{A} \cdot \mathbf{r} \rceil \in \mathcal{R}_p^k$
3: $c_2 = \lfloor p'/q \cdot \langle \mathbf{b}, \mathbf{r} \rangle + p'/t \cdot \mu \rceil \in \mathcal{R}_{p'}$
4: **return** $\mathsf{ct} = (\mathbf{c}_1, c_2)$

$\underline{\mathsf{Dec}(\mathsf{sk}, \mathbf{c})}$: ▷ $\mathsf{sk} = \mathbf{s}$, $\mathbf{c} = (\mathbf{c}_1, c_2)$

1: $\mu' = \lfloor t/p \cdot \langle \mathbf{c}_1, \mathbf{s} \rangle + t/p' \cdot c_2 \rceil \in \mathcal{R}_t$
2: **return** μ'

Figure 7: Description of SMAUG-T.PKE

Randomness Generation and Seed. Note that, in Figure 7, the key generation uses some randomness for generating $\mathbf{A}$, $\mathbf{s}$, and $\mathbf{e}$, and the encryption uses some randomness for generating $\mathbf{r}$. Thus, for implementation validation purposes, 256-bit seeds can be input to PKE.KeyGenand PKE.Enc. The seed for PKE.KeyGencan then be extended to two 256-bit seeds; one can be input to HWT and $\tilde{D}_\sigma$ and can be used for generating $\mathbf{s}$ and $\mathbf{e}$, and the other can be used for sampling $\mathbf{A}$. The seed for PKE.Enccan be input to spCBD and can be used for generating $\mathbf{r}$.

TiMER Parameter. One of the four parameter sets of SMAUG-T, namely, TiMER, has slightly different features compared to SMAUG-T128 parameter set; the rest are exactly the same as other parameters:

- Reduced message space: $\mu \leftarrow \mathsf{D2Enc}(\mu) \in \{0,1\}^{256}$.
- After decryption, the message needs to be decoded via D2Dec.

In the below, we then prove the completeness of SMAUG-T.PKE.

Theorem 9 (Completeness of SMAUG-T.PKE)**.** *Let* $\mathbf{A}$*,* $\mathbf{b}$*,* $\mathbf{s}$*,* $\mathbf{e}$*, and* $\mathbf{r}$ *are defined as in Figure 7. Let the moduli* t*,* p*,* p'*, and* q *satisfy* $t \mid p \mid q$ *and* $t \mid p' \mid q$*. Let* $\mathbf{e}_1 \in \mathcal{R}_{\mathbb{Q}}^k$ *and* $e_2 \in \mathcal{R}_{\mathbb{Q}}$ *be the rounding errors introduced from the scalings and roundings of* $\mathbf{A} \cdot \mathbf{r}$ *and* $\mathbf{b}^T \cdot \mathbf{r}$*. That is,* $\mathbf{e}_1 = \frac{q}{p}(\lfloor \frac{p}{q} \cdot \mathbf{A} \cdot \mathbf{r} \rceil \mod p) - (\mathbf{A} \cdot \mathbf{r} \mod q)$ *and* $e_2 = \frac{q}{p'}(\lfloor \frac{p'}{q} \cdot \langle \mathbf{b}, \mathbf{r} \rangle \rceil \mod p') - (\langle \mathbf{b}, \mathbf{r} \rangle \mod q)$*. Let* $\delta = \Pr\left[\|\langle \mathbf{e}, \mathbf{r} \rangle + \langle \mathbf{e}_1, \mathbf{s} \rangle + e_2 \|_\infty > \frac{q}{2t} \right]$*, where the probability is taken over the randomness of the encryption. Then* SMAUG-T.PKE *in Figure 7 is* $(1-\delta)$*-correct. That is, for every message* μ *and every key-pair* $(\mathsf{pk}, \mathsf{sk})$ *returned by* $\mathsf{KeyGen}(1^\lambda)$*, the decryption fails with a probability less than* δ*.*

Proof. By the definition of $\mathbf{e}_1$ and e_2, it holds that $\mathbf{c}_1 = \frac{p}{q} \cdot (\mathbf{A} \cdot \mathbf{r} + \mathbf{e}_1) \mod p$ and $c_2 = \frac{p'}{q} \cdot (\langle \mathbf{b}, \mathbf{r} \rangle + e_2) + \frac{p'}{t} \cdot \mu \mod p'$, where the coefficients of $\mathbf{e}_1$ and e_2 are in $\mathbb{Z} \cap (-\frac{q}{2p}, \frac{q}{2p}]$ and $\mathbb{Z} \cap (-\frac{q}{2p'}, \frac{q}{2p'}]$, respectively. Thus, the decryption of the ciphertext $(\mathbf{c}_1, c_2)$ can be written as

$$\begin{aligned}
\left\lfloor \frac{t}{p} \cdot \langle \mathbf{c}_1, \mathbf{s} \rangle + \frac{t}{p'} \cdot c_2 \right\rceil \mod t &= \left\lfloor \frac{t}{q} \left(\langle \mathbf{A} \cdot \mathbf{r}, \mathbf{s} \rangle + \langle \mathbf{e}_1, \mathbf{s} \rangle + \langle \mathbf{b}, \mathbf{r} \rangle + e_2 \right) + \mu \right\rceil \mod t \\
&= \left\lfloor \frac{t}{q} \left(\langle \mathbf{A}^\top \cdot \mathbf{s} + \mathbf{b}, \mathbf{r} \rangle + \langle \mathbf{e}_1, \mathbf{s} \rangle + e_2 \right) + \mu \right\rceil \mod t \\
&= \mu + \left\lfloor \frac{t}{q} \left(\langle \mathbf{e}, \mathbf{r} \rangle + \langle \mathbf{e}_1, \mathbf{s} \rangle + e_2 \right) \right\rceil \mod t.
\end{aligned}$$

This is equal to μ if and only if every coefficient of $\langle \mathbf{e}, \mathbf{r} \rangle + \langle \mathbf{e}_1, \mathbf{s} \rangle + e_2$ is in the interval $[-\frac{q}{2t}, \frac{q}{2t})$. It concludes the proof. □

Note, it can be trivially proven that the use of D2 encoding in TiMER parameter set does not change the completeness of SMAUG-T.

4.2 Specification of SMAUG-T.KEM

We introduce the key encapsulation mechanism SMAUG-T.KEM in Figure 8. SMAUG-T.KEM is designed following the Fujisaki–Okamoto transform with implicit rejection using the non-perfectly correct public key encryption SMAUG-T.PKE. The construction of SMAUG-T.KEM involves the use of the following symmetric primitives:

- Hash function H for hashing a public key,
- Hash function G for deriving a sharing key and a seed.

$\underline{\mathsf{KeyGen}(1^\lambda)}$:

1: $(\mathsf{pk}, \mathsf{sk}') \leftarrow \mathsf{SMAUG\text{-}T.PKE.KeyGen}(1^\lambda)$
2: $d \leftarrow \{0,1\}^{256}$
3: **return** $\mathsf{pk}, \mathsf{sk} = (\mathsf{sk}', d, \mathsf{pk})$

$\underline{\mathsf{Encap}(\mathsf{pk})}$: $\triangleright\ \mathsf{pk} = (\mathsf{seed}_{\mathbf{A}}, \mathbf{b})$

1: $\mu \leftarrow \{0,1\}^{256}$
2: $(K, \mathsf{seed}) \leftarrow G(\mu, H(\mathsf{pk}))$
3: $\mathsf{ct} \leftarrow \mathsf{SMAUG\text{-}T.PKE.Enc}(\mathsf{pk}, \mu; \mathsf{seed})$
4: **return** ct, K

$\underline{\mathsf{Decap}(\mathsf{sk}, \mathsf{ct})}$: $\triangleright\ \mathsf{sk} = (\mathsf{sk}', d, \mathsf{pk})$

1: $\mu' = \mathsf{SMAUG\text{-}T.PKE.Dec}(\mathsf{sk}', \mathsf{ct})$
2: $(K', \mathsf{seed}') \leftarrow G(\mu', H(\mathsf{pk}))$
3: $\mathsf{ct}' = \mathsf{SMAUG\text{-}T.PKE.Enc}\,(\mathsf{pk}, \mu'; \mathsf{seed}')$
4: $(\hat{K}, \cdot) \leftarrow G(d, H(\mathsf{ct}))$
5: **if** $\mathsf{ct} \neq \mathsf{ct}'$ **then**
6: $\quad K' \leftarrow \hat{K}$
7: **return** K'

Figure 8: Description of SMAUG-T.KEM

Randomness Generation and Seed. Note that, in Figure 8, the key generation uses some randomness for generating the keys and d, and the encapsulation uses some randomness to generate μ. Thus, for implementation validation purposes, we can separate the algorithms KEM.KeyGen, KEM.Encap, and KEM.Decap into internal and external algorithms, respectively. The 256-bit seeds can be input to the internal algorithms for KEM.KeyGenand KEM.Encap; two 256-bit seeds to KEM.KeyGenfor d and PKE.KeyGen and one 256-bit seed to KEM.Encap for μ. The external algorithms then need to wrap the internal algorithms and should securely sample the seeds.

TiMER Parameter. As in the SMAUG-T.PKE, we can easily construct the TiMER parameter set, which uses the TiMER parameter set of SMAUG-T.PKE in a black-box manner, with the following change:

- Reduced randomness space and entropy for μ, from $\{0,1\}^{256}$ to $\{0,1\}^{128}$

The Fujisaki–Okamoto transform used in Figure 8 defers from the $\mathsf{FO}_m^{\not\perp}$ transform in [44] in encapsulation and decapsulation. When generating the sharing key and randomness, SMAUG-T's Encap utilizes the hashed public key, which prevents certain multi-target attacks. As for Decap, if $\mathsf{ct} \neq \mathsf{ct}'$ holds, an alternative sharing key should be re-generated so as not to leak failure information against Side-Channel Attacks (SCA). However, even when the failure information is leaked, security can still rely on the explicit FO transform $\mathsf{FO}_m^{\perp}$, recently treated in [43] with a competitive bound.

We also remark that the randomly chosen message μ should be hashed in the environments using a non-cryptographic Random Number Generator (RNG) system. A True Random Number Generator (TRNG) is recommended to sample the message μ in such devices.

We now show the completeness of SMAUG-T.KEM based on the completeness of the underlying public key encryption scheme, SMAUG-T.PKE.

Theorem 10 (Completeness of SMAUG-T.KEM)**.** *We borrow the notations and assumptions from Theorem 9 and Figure 8. Then* SMAUG-T.KEM *in Figure 8 is also* $(1-\delta)$*-correct. That is, for every key-pair* $(\mathsf{pk}, \mathsf{sk})$ *generated by* $\mathsf{KeyGen}(1^\lambda)$*, the shared keys* K *and* K' *are identical with probability larger than* $1-\delta$*.*

Proof. The shared keys K and K' are identical if the decryption succeeds. Assuming the pseudorandomness of the hash function G, the probability of being $K \neq K'$ can be bounded by the DFP of SMAUG-T.PKE. The completeness of SMAUG-T.PKE (Theorem 9) concludes the proof. □

4.3 Security proof

When proving the security of the KEMs constructed using FO transform in the (Q)ROM, on typically relies on the generic reductions from one-wayness or IND-CPA security of the underlying PKE. In the ROM, SMAUG-T.KEM has a tight reduction from the IND-CPA security of the underlying PKE, SMAUG-T.PKE. However, like other lattice-based constructions, the underlying PKE has a chance of decryption failures, which makes the generic reduction unapplicable [55] or non-tight [43, 44, 12] in the QROM. Therefore, we prove the IND-CCA security of SMAUG-T.KEM based on the non-tight QROM reduction of [12] as explained in Section 2 by proving the IND-CPA security of SMAUG-T.PKE.

Theorem 11 (IND-CPA security of SMAUG-T.PKE)**.** *Assuming pseudorandomness of the underlying sampling algorithms, the* IND-CPA *security of* SMAUG-T. PKE *can be tightly reduced to the decisional MLWE and MLWR problems. Specifically, for any* IND-CPA*-adversary* $\mathcal{A}$ *of* SMAUG-T.PKE*, there exist adversaries* $\mathcal{B}_0$*,* $\mathcal{B}_1$*,* $\mathcal{B}_2$*, and* $\mathcal{B}_3$ *attacking the pseudorandomness of* XOF*, and the pseudorandomness of sampling algorithms, the hardness of MLWE, and the hardness of MLWR, respectively, such that,*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{SMAUG\text{-}T.PKE}}(\mathcal{A}) \le \mathsf{Adv}^{\mathsf{PR}}_{\mathsf{XOF}}(\mathcal{B}_0) + \mathsf{Adv}^{\mathsf{PR}}_{\mathsf{expandA,HWT,dGaussian}}(\mathcal{B}_1) \\ + \mathsf{Adv}^{\mathsf{MLWE}}_{n,q,k,k}(\mathcal{B}_2) + \mathsf{Adv}^{\mathsf{MLWR}}_{n,p,q,k+1,k}(\mathcal{B}_3).\end{aligned}$$

The secret distribution terms omitted in the last two advantages (of $\mathcal{B}_1$ *and* $\mathcal{B}_2$*) are uniform over ternary polynomials with Hamming weights* h_s *and* h_r*, respectively. The error distribution term omitted in the advantage of* $\mathcal{B}_2$ *is a pseudorandom distribution following the corresponding CDT.*

Proof. The proof proceeds by a sequence of hybrid games from G_0 to G_4 defined as follows:

- G_0: the genuine IND-CPA game,
- G_1: identical to G_0, except that the XOF is truly random,
- G_2: identical to G_1, except that the sampling algorithms are changed into truly random samplings,

- G_3: identical to G_2, except that $\mathbf{b}$ is randomly chosen from $\mathcal{R}_q^k$,
- G_4: identical to G_3, except that the ciphertext is randomly choosen from $\mathcal{R}_p^k \times \mathcal{R}_{p'}$. As a result, the public key and the ciphertexts are truly random.

We denote the advantage of the adversary on each game G_i as Adv_i, where $\mathsf{Adv}_0 = \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{SMAUG\text{-}T.PKE}}(\mathcal{A})$ and $\mathsf{Adv}_4 = 0$. Then, it holds that

$$|\mathsf{Adv}_0 - \mathsf{Adv}_1| \leq \mathsf{Adv}^{\mathsf{PR}}_{\mathsf{XOF}}(\mathcal{B}_0),$$

for some adversary $\mathcal{B}_0$ against the pseudorandomness of the extendable output function. Given that the only difference between the transcripts viewed in hybrid games G_1 and G_2 is the randomness sampling, it can be concluded that

$$|\mathsf{Adv}_1 - \mathsf{Adv}_2| \leq \mathsf{Adv}^{\mathsf{PR}}_{\mathsf{expandA,HWT,dGaussian}}(\mathcal{B}_1),$$

for some adversary, $\mathcal{B}_1$ attacking the pseudorandomness of at least one of the samplers. The difference in the games G_2 and G_3 is in the way the polynomial vector $\mathbf{b}$ is sampled. In G_2, it is sampled as part of an MLWE sample, whereas in G_3, it is randomly selected. Thus, the difference in the advantages Adv_2 and Adv_3 can be bounded by $\mathsf{Adv}^{\mathsf{MLWE}}_{n,q,k,k}(\mathcal{B}_2)$, where $\mathcal{B}_2$ is an adversary distinguishing the MLWE samples from random. In the hybrids G_3 and G_4, the only difference is in the way the ciphertexts are generated; they are either randomly chosen from $\mathcal{R}_p^k \times \mathcal{R}_{p'}$ or generated to be $(\mathbf{c}_1, \lfloor p'/p \cdot c_2 \rceil)$, where

$$\begin{bmatrix} \mathbf{c}_1 \\ c_2 \end{bmatrix} = \left\lfloor \frac{p}{q} \cdot \begin{pmatrix} \mathbf{A} \\ \mathbf{b}^\top \end{pmatrix} \cdot \mathbf{r} \right\rceil + \frac{p}{t} \cdot \begin{bmatrix} 0 \\ \mu \end{bmatrix}.$$

If an adversary $\mathcal{A}$ can distinguish the two ciphertexts, we can construct an adversary $\mathcal{B}_3$ distinguishing the MLWR sample from random: *for given a sample* $(\mathbf{A}, \mathbf{b}) \in \mathcal{R}_q^{(k+1)\times k} \times \mathcal{R}_p^{k+1}$, *$\mathcal{B}_3$ rewrites* $\mathbf{b}$ *as* $(\mathbf{b}_1, b_2) \in \mathcal{R}_p^k \times \mathcal{R}_p$, *computes* $(\mathbf{b}_1, \lfloor p'/p \cdot b_2 \rceil)$, *and use $\mathcal{A}$ to decide the ciphertext type. The output of $\mathcal{A}$ will be the output of $\mathcal{B}_3$.* Therefore, we can conclude the proof by observing that

$$|\mathsf{Adv}_3 - \mathsf{Adv}_4| \leq \mathsf{Adv}^{\mathsf{MLWR}}_{n,p,q,k+1,k}(\mathcal{B}_3). \qquad \square$$

Again, the D2 encoding does not introduce any changes in the above proof, as the encoded messages are added to a full random MLWR instances, assuming the MLWR hardness.

The classical IND-CCA security of SMAUG-T.KEM is then obtained directly from FO transforms [43] in the classical random oracle model. Theorem 8 implies the quantum IND-CCA security of SMAUG-T.KEM in the quantum random oracle model.

The TiMER parameter set is well-suited for lightweight IoT environments thanks to its smaller ciphertext size. However, the use of D2 encoding and the smaller randomness space may affect security in the future. For better-ensuring security when using TiMER parameter set, it is recommended to limit the number of Encap/Decap by considering the operating environment.

5 Parameter selection and concrete security

In this section, we first give a concrete security analysis of SMAUG-T and provide the parameter sets.

5.1 Concrete security estimation

We exploit the best-known lattice attacks to estimate the concrete security of SMAUG-T.

5.1.1 Core-SVP methodology

Most of the known attacks are essentially finding a nonzero short vector in Euclidean lattices, using the Block–Korkine– Zolotarev (BKZ) lattice reduction algorithm [21, 42, 56]. BKZ has been used in various lattice-based schemes [2, 15, 60, 36]. The security of the schemes is generally estimated as the time complexity of BKZ in core-SVP hardness introduced in [4]. It depends on the block size β of BKZ reporting the best performance. According to Becker et al. [8] and Chailloux et al. [20], the β-BKZ algorithm takes approximately $2^{0.292\beta+o(\beta)}$ and $2^{0.257\beta+o(\beta)}$ time in the classical and quantum setting, respectively. The polynomial factors and $o(\beta)$ terms in the exponent are ignored. We use the lattice estimator [1] to estimate the concrete security of SMAUG-T in core-SVP hardness.

5.1.2 Beyond Core-SVP methodology

In addition to lattice reduction attacks, we also take into consideration the cost of other types of attacks, e.g., algebraic attacks like the Arora-Ge attack or Coded-BKW attacks, and their variants. In general, these attacks have considerably higher costs and memory requirements compared to previously introduced attacks.

MLWE with fixed Hamming weight secret. We also focus on the attacks not considered in the lattice estimator, specifically those that target sparse secret, such as Meet-LWE [53] attack. This attack is inspired by Odlyzko's Meet-in-the-Middle approach and involves using representations of ternary secrets in additive shares. The asymptotic attack complexity is claimed as $\mathcal{S}^{0.25}$; however, it is far from the estimated attack costs in SMAUG-T parameter sets. Even the estimated cost has a significant gap with the real attack, due to the hidden costs behind the estimation.

MLWE with spCBD. When using spCBD, the number of non-zero coefficients is not fixed. Attacks like May's Meet-LWE [53] or Lee et al. [50] cannot be directly applied. As the distribution of h follows the binomial distribution centered at $n \cdot 2r$, an attacker can guess $h = n \cdot 2r$ or a value close to it and apply the attack. The probability of a correct guess is

$$\binom{n}{h} \cdot (2r)^h \cdot (1-2r)^{n-h} \ ,$$

which should be considered for the attack cost estimation. We note the value achieves the maximum when $h = \lfloor n \cdot 2r \rfloor$. Therefore, one can estimate the total cost of MLWE with spCBD secret as

$$\min_h \left\{ \frac{\mathsf{ATK}_h}{\binom{n}{h} \cdot (2r)^h \cdot (1-2r)^{n-h}} \right\} , \tag{1}$$

where ATK_h is the attack cost of [50] for MLWE with a secret having a fixed Hamming weight of h. For a rough estimation, we follow [50] and assume the attack cost ATK_h is greater than $\binom{n}{h}^{0.21}$, the secret space size to the power of 0.21, resulting in an asymptotic lower bound of the attack cost of

$$\min_h \left\{ \frac{1}{\binom{n}{h}^{0.79} \cdot (2r)^h \cdot (1-2r)^{n-h}} \right\} . \tag{2}$$

Depending on the parameters, the use of spCBD increases the attack cost compared to a fixed Hamming weight of $h = n \cdot 2r$ but also decreases the DFP in practice.

We summarize the costs of the algebraic and combinatorial attacks in Table 2. Attack costs for Arora-Ge and Coded-BKW are estimated with lattice estimator [1]. The estimated cost of Arora-Ge attack on SMAUG-T256 is not determined by lattice-estimator, outputting ∞, which is at least a thousand bits of security. The costs for the Meet-LWE attack are estimated with a python script[2] based on May's analysis [53], best among Rep-1 and Rep-2. In addition, we also consider the attack of Lee et al. [50] and its variant.[3] The hardness of an MLWE sample with the secret of a fixed Hamming weight is given based on the analysis of [50]. For the hardness of an MLWR sample with the secret of non-fixed weight spCBD sampler, we applied a variant of the attack as described in Section 5.1.2: We first find h that minimizes the Equation 2 ($h = 102, 102, 384, 352$) then we calculate the attack cost based on the Equation 1. However, as we are unaware of Lee et al.'s attack cost estimator exactly for each h, we give a lower bound of our attack based on their analysis. By using the $\mathsf{ATK}_{h'}$ that we know ($h' = 100, 100, 264, 348$), satisfying $h' < h$ so that $\mathsf{ATK}_h > \mathsf{ATK}_{h'}$ holds. This means that we only give a lower bound of the attack cost estimation, which can be improved using the estimator of Lee et al.

Refined estimation in terms of gate count. Beyond the (relatively) older Core-SVP methods, Kyber (ML-KEM), Dilithium (ML-DSA), FrodoKEM, and HAETAE also adopt refined gate-count–based security estimations, enabling direct comparison with NIST's security requirements of 2^{143}, 2^{207}, and 2^{272} for levels 1, 3, and 5, respectively. This approach incorporates recent advances in lattice sieving, progressive BKZ, improved BKZ simulation, and dimension-for-free techniques, yielding more accurate estimates of attack costs. As a result, the evaluated security levels align more closely with NIST's categories and reflect the current state of knowledge regarding the practical complexity of lattice attacks.

The estimated results are summarized in Table 2, estimated using a script[4] adapted from the one[5] used for Kyber, Dilithium, and FrodoKEM. The estimated gate counts exceed the NIST's requirements for each level.

[2]The script can be found on the team SMAUG-T website: `http://kpqc.cryptolab.co.kr/`

[3]We remark that the attack of Lee et al. and its cost estimation is not yet verified; however, it is worth adding such a countermeasure to the scheme against the attacks.

[4]Available at `https://github.com/hmchoe0528/refined-estimate-smaugt`.

[5]Available at `https://github.com/lducas/leaky-LWE-Estimator/tree/NIST-round3/NIST-round3`.

Parameters sets Security level		TiMER 1	SMAUG-T Mode 1 1	SMAUG-T Mode 3 3	SMAUG-T Mode 5 5
Classical core-SVP		119.7	119.7	180.2	250.1
Algebraic & Combinatorial attacks (in $\log_2$)					
Arora-Ge	time (mem)	693.6 (553.0)	693.6 (553.0)	- (908.9)	- -
BKW	time (mem)	144.7 (132.7)	144.7 (132.7)	213.7 (202.1)	269.0 (257.0)
Meet-LWE	time (mem)	177.2 (157.4)	177.2 (157.4)	295.6 (259.1)	401.4 (353.1)
Lee et al. [50]*	time	$(148, 132)$	$(148, 132)$	$(236, 241)$	$(309, 317)$
Refined Est.	gates	$(152, 152)$	$(152, 152)$	$(212, 219)$	$(282, 286)$

Table 2: Attack costs beyond Core-SVP. The estimated cost of the Arora-Ge attack sometimes overflowed, implying that it requires at least 2^{1000} of operations. For the attack of Lee et al., we apply our modifications detailed in Sections 5.1.2, where the estimated costs are given for both keys (MLWE) and ciphertexts (MLWR).

5.1.3 MLWE hardness

We estimated the cost of the best-known attacks for MLWE, including *primal attack*, *dual attack*, and their hybrid variations, in the core-SVP hardness. We remark that any $\mathsf{MLWE}_{n,q,k,\ell,\eta}$ instance can be viewed as an $\mathsf{LWE}_{q,nk,n\ell,\eta}$ instance. Although the MLWE problem has an additional algebraic structure compared to the LWE problem, no attacks currently take advantage of this structure. Therefore, we assess the hardness of the MLWE problem based on the hardness of the corresponding LWE problem. We also consider the distributions of secret and noise when estimating the concrete security of SMAUG-T. We have also analyzed the costs of recent attacks that aim to target the MLWE problem with sparse secrets. Our narrow discrete Gaussian sampler's tail bound is considered in estimating the security using the lattice estimator.

5.1.4 MLWR hardness

To measure the hardness of the MLWR problem, we treat it as an MLWE problem since no known attack utilizes the deterministic error term in the MLWR structure. Banerjee et al. [7] provided the reduction from the MLWE problem to the MLWR problem, which was subsequently improved in [5, 6, 14]. Basically, for given an MLWR sample $(\mathbf{A}, \lfloor p/q \cdot \mathbf{A} \cdot \mathbf{s} \rceil \mod p)$ with uniformly chosen $\mathbf{A} \leftarrow \mathcal{R}_q^k$ and $\mathbf{s} \leftarrow \mathcal{R}_p^\ell$, it can be expressed as $(\mathbf{A}, p/q \cdot (\mathbf{A} \cdot \mathbf{s} \mod q) + \mathbf{e} \mod p)$. The MLWR sample can be converted to an MLWE sample over $\mathcal{R}_q$ by multiplying q/p as $(\mathbf{A}, \mathbf{b} = \mathbf{A} \cdot \mathbf{s} + q/p \cdot \mathbf{e} \mod q)$. Assuming that the error term in the resulting MLWE sample is a random variable, uniformly distributed within the interval $(-q/2p, q/2p]$, we can estimate the hardness of the MLWR problem as the hardness of the corresponding MLWE problem.

5.2 Parameter sets

The SMAUG-T is parameterized by various integers such as n, k, q, p, p', t, h_s and h_r, as well as a standard deviation $\sigma > 0$ for the discrete Gaussian noise. Our main focus when selecting these parameters is to minimize the ciphertext size while maintaining security. We first set our ring dimension to $n = 256$ and plaintext modulus to $t = 2$ to have a 256-bit (for SMAUG-T128, 192, 256) or 128-bit (for TiMER) message space. The sharing-key space is 256-bit for all the parameter sets. Then, we search for parameters with enough security to offer the smallest ciphertext size. Starting from parameters with a tiny ciphertext size, we increase the ciphertext size, h_s, r, and σ, then search for the parameters with enough security. Once we have a candidate, we compute the DFP. If it is low enough, we can choose the compression parameter p', but if not, we continue searching for appropriate parameters. If the DFP is low enough, the compression factor p' can be set to a smaller integer.

Parameters sets Security level	TiMER 1	SMAUG-T128 1	SMAUG-T192 3	SMAUG-T256 5
n	256	256	256	256
k	2	2	3	4
(q, p, t)	$(1024, 256, 2)$	$(1024, 256, 2)$	$(2048, 512, 2)$	$(2048, 512, 2)$
p' (compression)	8	32	16	128
h_s (HWT for $\mathbf{s}$)	140	140	264	348
r (spCBD for $\mathbf{r}$)	$1/8$	$1/8$	$1/4$	$3/16$
σ ($\tilde{D}_\sigma$ for errors)	1.0625	1.0625	1.0625	1.0625
Classical core-SVP	119.7	119.7	180.2	250.1
Quantum core-SVP	105.4	105.4	158.6	221.0
Beyond core-SVP	132	132	214	269
#Gates (Ref. Est.)	152	152	212	282
DFP	-161.0	-118.3	-179.2	-194.2
Public key	672	672	1088	1440
Ciphertext	608	672	992	1376

Table 3: Parameters for SMAUG-T. Classical and quantum security is given in core-SVP hardness. The DFP (in $\log_2$) and sizes (in bytes) are also given in advance.

Table 3 outlines the whole set of recommended parameters corresponding to NIST's security levels 1, 3, and 5. For security levels 3 and 5, we can not find the parameters with $q = 1024$, so we use $q = 2048$. Especially, the standard deviation $\sigma = 1.0625$ is the same across the whole parameter sets.

TiMER, an additional parameter set, further investigates the room for efficiency, introducing the D2 encoding to SMAUG-T Mode 1. It has a 64-byte smaller ciphertext size than SMAUG-T Mode 1. TiMER sufficiently lowers DFP through D2 encoding and error reconciliation techniques. Thanks to this lowered DFP, p' was reduced from 32 to 8, further compressing the ciphertext.

The core-SVP hardness is estimated via the lattice estimator [1] using the cost model "ADPS16" introduced in [4] and "MATZOV" [52]. In the table, the smaller cost is reported. We assumed that the number of 1s is equal to the number of -1s for simplicity, which conservatively underestimates security.

The security beyond core-SVP is estimated via the lattice estimator [1] and the Python script implementing the Meet-LWE attack cost estimation. It shows the lowest attack costs among coded-BKW, Arora-Ge, and Meet-LWE attack and their variants.

5.3 Decryption failure probability

Our primary goal is to push the efficiency of the lattice-based KEMs to the limit while maintaining roughly the same level of security, so we follow the frameworks given in Kyber and Saber. We set the DFPs as small as $\approx 2^{-\lambda}$ for a desired security parameter λ, except for the SMAUG-T Mode 5 parameter set. We set the DFP of SMAUG-T Mode 5 at least much smaller than that of Kyber and Saber.

The impact of DFP on the security of KEM is still being investigated. However, we can justify why our choice is sufficient for real-world scenarios, focusing on SMAUG-T Mode 5. To do so, we make the following assumptions:

1. Each key pair has a limit of $Q_{\text{limit}} = 2^{64}$ decryption queries, as specified in NIST's proposal call.
2. There are approximately 2^{33} people worldwide, each with hundreds of devices. Each device has thousands of *usable* public keys broadcasted for KEM.
3. We introduce an *observable probability* and assume it is far less than 2^{-20}. Even though the decryption failure occurs, it can only be used for an attack when observed. Attackers can observe it through a side-channel attack, which enables the observation of decapsulation failures in the mounted device or through direct communications after key derivation. This allows the detection of decryption failures with a communication per key pair. We assume the two cases can occur much less than 2^{-20}, as they require physically mounted devices or communications with shared keys.

Based on these assumptions, we can deduce that the number of observable decryption failures can be upper bounded by $2^{64+33+10+12} \cdot 2^{-20} = 2^{99}$. Based on the best-known (multi-target) attacks for Saber [28, Figure 7a], the quantum cost for finding a single failing ciphertext that may lead to a successful attack of SMAUG-T Mode 5 is expected to be much higher than 2^{300}, as desired[6]. Regardless of the attack cost estimated above, the scenario of checking the failures in more than 2^{40} different devices is already way too far from the real-world attack scenario.

[6]Specifically, the number of observable failures must be larger than $1/\beta$ in [28] to observe at least one failing ciphertext. That is, β should be smaller than 2^{-93}. When this is assumed, the quantum cost is then $1/\beta\sqrt{\alpha}$, given in the x-axis.

6 Implementation

In this section, we consider the implementation of SMAUG-T and present the performance for each parameter set. We provide a few C implementations: The constant-time reference implementation of SMAUG-T parameter sets can be found in the `reference_implementation`, and an optimized implementation utilizing AVX2 intrinsics on Intel(R) is included in the `optimized_implementation`. Our implementations, along with the supporting scripts, are accessible on our website: `www.kpqc.cryptolab.co.kr/smaug-t`.

6.1 Performance

In the reference implementation and additional implementation, we instantiate the hash functions G, H, the extendable output function XOF, and the pseudo-random function PRF with the following symmetric primitives: G and PRF are instantiated with SHAKE256, H is instantiated with SHA3-256, XOF is instantiated with SHAKE128.

Table 4 presents the performance results of SMAUG-T. For a fair comparison, we also performed measurements on the same system with identical settings of the reference implementation of Kyber [7]. All benchmarks are obtained on one core of an Intel(R) Core(TM) i7-10700K CPU processor with a clock speed of 3.80GHz. The benchmarking machine has 64 GB of RAM and runs Debian GNU/Linux with Linux kernel version 5.4.0. The implementation is compiled with gcc version 11.4.0, and the compiler flags as indicated in the Makefile included in the submission package.

	Cycles (ref)						**Cycles (AVX2)**					
Schemes	KeyGen		Encap		Decap		KeyGen		Encap		Decap	
TiMER	110	1	100	1	135	1	-		-		-	
SMAUG-T Mode 1	110	1	100	1	136	1	38	1	23	1	35	1
Kyber512	128	1.2	158	1.6	187	1.4	27	0.7	39	1.7	29	0.8
SMAUG-T Mode 3	219	1	204	1	253	1	57	1	46	1	61	1
Kyber768	209	1	255	1.3	286	1.1	44	0.8	65	1.4	44	0.7
SMAUG-T Mode 5	357	1	334	1	414	1	77	1	65	1	86	1
Kyber1024	321	0.9	369	1.1	414	1	60	0.8	79	1.2	63	0.7

Table 4: Median kilocycle counts of 1000 executions for SMAUG-T and Kyber (and their ratios). "ref" refers to the reference C implementation, while "AVX2" refers to the implementation with AVX2 intrinsics.

[7]From `github.com/pq-crystals/kyber` (518de24)

7 Side Channel Analysis

SMAUG-T is a scheme based on MLWE and MLWR that has many similarities to Kyber and Saber. As a result of the NIST competition, much research has been conducted on side-channel analysis and countermeasures for Kyber and Saber [17, 9]. These previous findings can also be applied to SMAUG-T. Therefore, we decided to focus our analysis on the characteristic designs in which SMAUG-T differs from Kyber or Saber. While KpqC round 1 focused on timing attacks, power/EM-based attacks are becoming increasingly critical with advanced attack techniques and tools, necessitating proactive countermeasures. In particular, the recently announced clustering attack [57] has become a more lethal threat due to the small number of traces and advances in deep learning technology. Thus, we discuss the security of SMAUG-T against physical attacks based on power/EM.

7.1 Timing analysis

Samplers

At present, SMAUG-T has been carefully implemented to avoid time variations such as branches with respect to secrets. For key generation, a shuffling-based constant-time and unbiased fixed-weight sampler has been used as the fixed-weight sampler. Furthermore, for ephemeral randomness in encapsulation/decapsulation, we propose a new constant-time sparse CBD sampler. This sampler is constructed solely from bit operations, making it highly secure and efficient for implementation. Previous PQC algorithms utilizing Gaussian noises have employed various Gaussian samplers. However, designing Gaussian samplers that operate in constant-time is challenging, and BLISS has suffered from timing attacks [39]. We adopted $\mathsf{dGaussian}_\sigma$, a constant-time implementation well-known for its efficacy, into SMAUG-T to mitigate timing attacks.

D2 encoding and error reconciliation

As mentioned earlier, D2 encoding and error reconciliation were used in NewHope, and due to modulus reduction, the D2 implementation was not constant-time. In NewHope, they solved this problem with constant-time Barrett reduction. On the other hand, in TiMER, since the modulus is all powers of 2, the modulus reduction can be replaced by a shift operation, eliminating the attack surface.

7.2 Differential analysis

$\mathsf{dGaussian}_\sigma$ sampler

Potential vulnerabilitie related to Power/EM-based SCA for $\mathsf{dGaussian}_\sigma$ was reported during KpqC round 1. There were no specific attack scenario and applying this vulnerability in real-world environments may be challenging; however, recent advancements in deep-learning and clustering technologies suggest that this attack could become a practical vulnerability. Therefore, we applied a countermeasure to $\mathsf{dGaussian}_\sigma$ to prevent these attacks. In the public key generation process of SMAUG-T, the $\mathsf{dGaussian}_\sigma$ function produces integer intermediate values within the range of [-3, 3] when generating Gaussian errors. The significant hamming weight difference between positive and negative values distinguishes these values into two sets. (ex, {-3, -2, -1} / {0, 1, 2, 3}) With this distinction and linear algebraic approach, there is the possibility of recovering secret keys or reducing candidates.

Therefore, countermeasures are necessary. First, we consider masking techniques. However, designing a general random masking scheme efficiently in situations with numerous nonlinear bit-operations can be challenging and may incur significant overhead. For example, Krausz et al. [48] have recently proposed masking methods for the fixed hamming weight sampler; their efficiency is lacking, so we see it as future work. Hiding can be considered as another countermeasure. This attack involves logic that categorizes coefficients during the key generation process, making it difficult to distinguish which coefficients belong to which set is sufficient to respond effectively. Therefore, applying hiding would be more effective than masking. The errors are calculated in Figure 5 and then stored sequentially in each index. We applied the Fisher-Yates shuffle algorithm only to the corresponding loops and the loops where those values are utilized. In environments such as TLS, key generation is typically performed on servers with high-performance capabilities and operates less frequently than encryption. Therefore, such countermeasures will have a minimal impact on the cryptographic system.

D2 encoding and error reconciliation

There has reported the vulnerability related to power analysis caused by differences in the Hamming weight of the $mask$ variable in the D2 encoding process. This attack was complemented in TiGER v2.1 by changing the $mask$ variable to 1 and 0 and applying a countermeasure to minimize the Hamming weight difference. TiMER also prevents such vulnerability with the same countermeasure.

References

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

[2] Alkim, E., Barreto, P.S.L.M., Bindel, N., Kramer, J., Longa, P., Ricardini, J.E.: The lattice-based digital signature scheme qtesla. Cryptology ePrint Archive, Paper 2019/085 (2019), `https://eprint.iacr.org/2019/085`

[3] Alkim, E., Bos, J., Ducas, L., Longa, P., Mironov, I., Naehrig, M., Nikolaenko, V., Peikert, C., Raghunathan, A., Stebila, D.: Frodokem: Algorithm specifications and supporting documentation (2021)

[4] Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key exchange - A new hope. In: Holz, T., Savage, S. (eds.) USENIX Security 2016. pp. 327–343. USENIX Association (Aug 2016), `https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/alkim`

[5] Alperin-Sheriff, J., Apon, D.: Dimension-preserving reductions from lwe to lwr. Cryptology ePrint Archive, Paper 2016/589 (2016), `https://eprint.iacr.org/2016/589`

[6] Alwen, J., Krenn, S., Pietrzak, K., Wichs, D.: Learning with rounding, revisited. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology – CRYPTO 2013. pp. 57–74. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)

[7] Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom functions and lattices. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology – EUROCRYPT 2012. pp. 719–737. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)

[8] Becker, A., Ducas, L., Gama, N., Laarhoven, T.: New directions in nearest neighbor searching with applications to lattice sieving, pp. 10–24. Society for Industrial and Applied Mathematics (2016). https://doi.org/10.1137/1.9781611974331.ch2

[9] Beirendonck, M.V., D'anvers, J.P., Karmakar, A., Balasch, J., Verbauwhede, I.: A side-channel-resistant implementation of saber. J. Emerg. Technol. Comput. Syst. **17**(2) (apr 2021). https://doi.org/10.1145/3429983

[10] Bermudo Mera, J.M., Karmakar, A., Verbauwhede, I.: Time-memory trade-off in toom-cook multiplication: an application to module-lattice based cryptography. IACR Transactions on Cryptographic Hardware and Embedded Systems **2020**(2), 222–244 (2020)

[11] Bi, L., Lu, X., Luo, J., Wang, K.: Hybrid dual and meet-LWE attack. In: Nguyen, K., Yang, G., Guo, F., Susilo, W. (eds.) ACISP 22. LNCS, vol. 13494, pp. 168–188. Springer, Cham (Nov 2022). https://doi.org/10.1007/978-3-031-22301-3_9

[12] Bindel, N., Hamburg, M., Hövelmanns, K., Hülsing, A., Persichetti, E.: Tighter proofs of CCA security in the quantum random oracle model. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019, Part II. LNCS, vol. 11892, pp. 61–90. Springer, Cham (Dec 2019). https://doi.org/10.1007/978-3-030-36033-7_3

[13] Birkett, J., Dent, A.W.: Relations among notions of plaintext awareness. In: Cramer, R. (ed.) Public Key Cryptography – PKC 2008. pp. 47–64. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)

[14] Bogdanov, A., Guo, S., Masny, D., Richelson, S., Rosen, A.: On the hardness of learning with rounding over small modulus. In: Kushilevitz, E., Malkin, T. (eds.) Theory of Cryptography. pp. 209–224. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)

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

[16] Bos, J.W., Costello, C., Ducas, L., Mironov, I., Naehrig, M., Nikolaenko, V., Raghunathan, A., Stebila, D.: Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016. pp. 1006–1018. ACM Press (Oct 2016). https://doi.org/10.1145/2976749.2978425

[17] Bos, J.W., Gourjon, M., Renes, J., Schneider, T., van Vredendaal, C.: Masking Kyber: First- and higher-order implementations. IACR TCHES **2021**(4), 173–214 (2021). https://doi.org/10.46586/tches.v2021.i4.173-214, https://tches.iacr.org/index.php/TCHES/article/view/9064

[18] Bossuat, J.P., Troncoso-Pastoriza, J.R., Hubaux, J.P.: Bootstrapping for approximate homomorphic encryption with negligible failure-probability by using sparse-secret encapsulation. In: Ateniese, G., Venturi, D. (eds.) ACNS 2022. LNCS, vol. 13269, pp. 521–541. Springer, Cham (Jun 2022). https://doi.org/10.1007/978-3-031-09234-3_26

[19] Boudgoust, K., Jeudy, C., Roux-Langlois, A., Wen, W.: On the hardness of module learning with errors with short distributions. Journal of Cryptology **36**(1), 1 (Jan 2023). https://doi.org/10.1007/s00145-022-09441-3

[20] Chailloux, A., Loyer, J.: Lattice sieving via quantum random walks. In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT. pp. 63–91 (2021)

[21] Chen, Y., Nguyen, P.Q.: BKZ 2.0: Better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 1–20. Springer, Berlin, Heidelberg (Dec 2011). https://doi.org/10.1007/978-3-642-25385-0_1

[22] Cheon, J.H., Choe, H., Hong, D., Yi, M.: SMAUG: Pushing lattice-based key encapsulation mechanisms to the limits. In: Carlet, C., Mandal, K., Rijmen, V. (eds.) SAC 2023. LNCS, vol. 14201, pp. 127–146. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-53368-6_7

[23] Cheon, J.H., Han, K., Kim, A., Kim, M., Song, Y.: Bootstrapping for approximate homomorphic encryption. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 360–384. Springer, Cham (Apr / May 2018). https://doi.org/10.1007/978-3-319-78381-9_14

[24] Cheon, J.H., Han, K., Kim, J., Lee, C., Son, Y.: A practical post-quantum public-key cryptosystem based on spLWE. In: Hong, S., Park, J.H. (eds.) ICISC 16. LNCS, vol. 10157, pp. 51–74. Springer, Cham (Nov / Dec 2017). https://doi.org/10.1007/978-3-319-53177-9_3

[25] Cheon, J.H., Kim, D., Lee, J., Song, Y.: Lizard: Cut off the tail! A practical post-quantum public-key encryption from LWE and LWR. In: Catalano, D., De Prisco, R. (eds.) SCN 18. LNCS, vol. 11035, pp. 160–177. Springer, Cham (Sep 2018). https://doi.org/10.1007/978-3-319-98113-0_9

[26] Chung, C.M.M., Hwang, V., Kannwischer, M.J., Seiler, G., Shih, C.J., Yang, B.Y.: NTT multiplication for NTT-unfriendly rings: New speed records for Saber and NTRU on Cortex-M4 and AVX2. IACR Transactions on Cryptographic Hardware and Embedded Systems **2021**(2), 159–188 (Feb 2021). https://doi.org/10.46586/tches.v2021.i2.159-188, artifact available at `https://artifacts.iacr.org/tches/2021/a7`

[27] Cook, S.A., Aanderaa, S.O.: On the minimum computation time of functions. Transactions of the American Mathematical Society **142**, 291–314 (1969)

[28] D'Anvers, J.P., Batsleer, S.: Multitarget decryption failure attacks and their application to Saber and Kyber. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) PKC 2022, Part I. LNCS, vol. 13177, pp. 3–33. Springer, Cham (Mar 2022). https://doi.org/10.1007/978-3-030-97121-2_1

[29] D'Anvers, J.P., Guo, Q., Johansson, T., Nilsson, A., Vercauteren, F., Verbauwhede, I.: Decryption failure attacks on ind-cca secure lattice-based schemes. In: Lin, D., Sako, K. (eds.) Public-Key Cryptography – PKC 2019. pp. 565–598. Springer International Publishing, Cham (2019)

[30] D'Anvers, J.P., Karmakar, A., Roy, S.S., Vercauteren, F.: Saber: Module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM. In: Joux, A., Nitaj, A., Rachidi, T. (eds.) AFRICACRYPT 18. LNCS, vol. 10831, pp. 282–305. Springer, Cham (May 2018). https://doi.org/10.1007/978-3-319-89339-6_16

[31] Dent, A.W.: A designer's guide to kems. In: Cryptography and Coding: 9th IMA International Conference, Cirencester, UK, December 16-18, 2003. Proceedings 9. pp. 133–151. Springer (2003)

[32] D'Anvers, J.P., Karmakar, A., Sinha Roy, S., Vercauteren, F.: Saber: Module-lwr based key exchange, cpa-secure encryption and cca-secure kem. In: International Conference on Cryptology in Africa. pp. 282–305. Springer (2018)

[33] D'Anvers, J.P., Karmakar, A., Sinha Roy, S., Vercauteren, F.: Saber: Module-lwr based key exchange, cpa-secure encryption and cca-secure kem. In: Progress in Cryptology–AFRICACRYPT 2018: 10th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 7–9, 2018, Proceedings 10. pp. 282–305. Springer (2018)

[34] Espitau, T., Joux, A., Kharchenko, N.: On a dual/hybrid approach to small secret LWE - A dual/enumeration technique for learning with errors and application to security estimates of FHE schemes. In: Bhargavan, K., Oswald, E., Prabhakaran, M. (eds.) INDOCRYPT 2020. LNCS, vol. 12578, pp. 440–462. Springer, Cham (Dec 2020). https://doi.org/10.1007/978-3-030-65277-7_20

[35] Filho, D.L.G., Silva, T.S.R., López, J.: Efficient isochronous fixed-weight sampling with applications to NTRU. Cryptology ePrint Archive, Paper 2024/548 (2024), https://eprint.iacr.org/2024/548, https://eprint.iacr.org/2024/548

[36] Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: Falcon: Fast-fourier lattice-based compact signatures over ntru. Submission to the NIST's post-quantum cryptography standardization process **36**(5) (2018)

[37] Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 537–554. Springer, Berlin, Heidelberg (Aug 1999). https://doi.org/10.1007/3-540-48405-1_34

[38] Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. Journal of Cryptology **26**(1), 80–101 (Jan 2013). https://doi.org/10.1007/s00145-011-9114-1

[39] Groot Bruinderink, L., Hülsing, A., Lange, T., Yarom, Y.: Flush, gauss, and reload–a cache attack on the bliss lattice-based signature scheme. In: International Conference on Cryptographic Hardware and Embedded Systems. pp. 323–345. Springer (2016)

[40] Halevi, S., Shoup, V.: Bootstrapping for HElib. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 641–670. Springer, Berlin, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46800-5_25

[41] Hamburg, M.: Threebears. In: Second PQC Standardization Conference. University of California, Santa Barbara, USA (2019)

[42] Hanrot, G., Pujol, X., Stehlé, D.: Algorithms for the shortest and closest lattice vector problems. In: Chee, Y.M., Guo, Z., Ling, S., Shao, F., Tang, Y., Wang, H., Xing, C. (eds.) Coding and Cryptology. pp. 159–190. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)

[43] Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 341–371. Springer, Cham (Nov 2017). https://doi.org/10.1007/978-3-319-70500-2_12

[44] Hövelmanns, K., Kiltz, E., Schäge, S., Unruh, D.: Generic authenticated key exchange in the quantum random oracle model. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part II. LNCS, vol. 12111, pp. 389–422. Springer, Cham (May 2020). https://doi.org/10.1007/978-3-030-45388-6_14

[45] Howgrave-Graham, N., Nguyen, P.Q., Pointcheval, D., Proos, J., Silverman, J.H., Singer, A., Whyte, W.: The impact of decryption failures on the security of ntru encryption. In: Boneh, D. (ed.) Advances in Cryptology - CRYPTO 2003. pp. 226–246. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)

[46] Jiang, H., Zhang, Z., Chen, L., Wang, H., Ma, Z.: Ind-cca-secure key encapsulation mechanism in the quantum random oracle model, revisited. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018. pp. 96–125. Springer International Publishing, Cham (2018)

[47] Karatsuba, A.A., Ofman, Y.P.: Multiplication of many-digital numbers by automatic computers. Doklady Akademii Nauk **145**(2), 293–294 (1962)

[48] Krausz, M., Land, G., Richter-Brockmann, J., Güneysu, T.: A holistic approach towards side-channel secure fixed-weight polynomial sampling. In: Public-Key Cryptography–PKC 2023: 26th IACR International Conference on Practice and Theory of Public-Key Cryptography, Atlanta, GA, USA, May 7–10, 2023, Proceedings, Part II. pp. 94–124. Springer (2023)

[49] Langlois, A., Stehlé, D., Steinfeld, R.: GGHLite: More efficient multilinear maps from ideal lattices. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 239–256. Springer, Berlin, Heidelberg (May 2014). https://doi.org/10.1007/978-3-642-55220-5_14

[50] Lee, E., Lee, J., Wang, Y.: Improved Meet-LWE Attack via Ternary Trees. Cryptology ePrint Archive, Paper 2024/824 (2024), `https://eprint.iacr.org/2024/824`

[51] Lee, J., Kim, D., Lee, H., Lee, Y., Cheon, J.H.: Rlizard: Post-quantum key encapsulation mechanism for iot devices. IEEE Access **7**, 2080–2091 (2018)

[52] MATZOV: Report on the Security of LWE: Improved Dual Lattice Attack (Apr 2022). https://doi.org/10.5281/zenodo.6493704

[53] May, A.: How to meet ternary LWE keys. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part II. LNCS, vol. 12826, pp. 701–731. Springer, Cham, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84245-1_24

[54] Pöppelmann, T., Güneysu, T.: Towards practical lattice-based public-key encryption on reconfigurable hardware. In: Selected Areas in Cryptography–SAC 2013: 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers 20. pp. 68–85. Springer (2014)

[55] Saito, T., Xagawa, K., Yamakawa, T.: Tightly-secure key-encapsulation mechanism in the quantum random oracle model. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part III. LNCS, vol. 10822, pp. 520–551. Springer, Cham (Apr / May 2018). https://doi.org/10.1007/978-3-319-78372-7_17

[56] Schnorr, C.P., Euchner, M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Mathematical programming **66**(1), 181–199 (1994)

[57] Sim, B.Y., Park, A., Han, D.G.: Chosen-ciphertext clustering attack on crystals-kyber using the side-channel leakage of barrett reduction. IEEE Internet of Things Journal **9**(21), 21382–21397 (2022). https://doi.org/10.1109/JIOT.2022.3179683

[58] Son, Y., Cheon, J.H.: Revisiting the hybrid attack on sparse secret lwe and application to he parameters. In: Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography. p. 11–20. WAHC'19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3338469.3358941

[59] Toom, A.L.: The complexity of a scheme of functional elements realizing the multiplication of integers, published in soviet math (translations of dokl. adad. nauk. sssr), 4 (1963)

[60] Vercauteren, I.F., Sinha Roy, S., D'Anvers, J.P., Karmakar, A.: Saber: Mod-lwr based kem, nIST PQC Round 3 Submision