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

Jung Hee Cheon[1,2†], Hyeongmin Choe[1], Joongeun Choi[3], Dongyeon Hong[4], Jeongdae Hong[5], Chi-Gon Jung[3], Honggoo Kang[3], Janghyun Lee[3], Seonghyuck Lim[3], Aesun Park[3], Seunghwan Park[3†], Hyoeun Seong[2], and Junbum Shin[2]

[1] Seoul National University {jhcheon, sixtail528}@snu.ac.kr
[2] CryptoLab Inc. {she000, junbum.shin}@cryptolab.co.kr
[3] Defense Counter-intelligence Command
{joongeuntom, wjdclrhs, honggoonin, jhlee, 794613sh, aesunpark18, horriblepaper}@gmail.com
[4] The Affiliated Institute of ETRI jjoker041@gmail.com
[5] Ministry of National Defense ghjd2000@gmail.com

Version 3.0
(February 23, 2024)

Abstract. This paper introduces SMAUG-T, a lattice-based post-quantum key exchange algorithm submitted to Round 2 of the Korean Post-Quantum Cryptography Competition (KpqC). SMAUG-T is designed by merging SMAUG and TiGER according to the KpqC Round 1 recommendation. 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. With a constant-time C reference implementation, SMAUG-T achieves ciphertext sizes up to 12% and 9% smaller than Kyber and Saber, with much faster running time, up to 103% and 58%, respectively. Compared to Sable, SMAUG-T has the same ciphertext sizes but a larger public key, which gives a trade-off between the public key size versus performance; SMAUG-T has 39%-55% faster encapsulation and decapsulation speed in the parameter sets having comparable security

Keywords: Lattice-based Cryptography · Post-Quantum Cryptography · Key Encapsulation Mechanism · Module Learning With Errors · Module Learning With Roundings.

This work is submitted to 'Korean Post-Quantum Cryptography Competition' (www.kpqc.or.kr).

†: Principal submitters.

Changelog

February 23, 2024 (version 3.0) The two schemes SMAUG and TiGER are merged to SMAUG-T, taking the advantageous features from both schemes. Along with the three SMAUG parameter sets (renamed as SMAUG-T128, 192, 256), a new parameter set TiMER is added, which allows a much lower decryption failure probability, thanks to the error reconciliation from D2 encoding.

A countermeasure was included for the side channel analysis as some vulnerabilities were reported in the KpqC round 1. Hamming weight sampling has been changed and applied to the default, and dGaussian sampling with hiding is provided as an additional implementation.

An optimized implementation with AVX vectorization is also provided, which reports 1.7-1.8x speed-ups. For a fair comparison to other KpqC candidates that provide implementations using so-called 90s symmetric primitives, we also provide an optimized implementation using the 90s, which reports 2.5-3.0x speed-ups compared to the reference implemenation.

October 30, 2023 (version 2.0) First, we updated the hamming weight sampler HWT, which was not running at a constant time due to the dependency on the number of hash calls. The new hamming weight sampler is a hybrid of the previous HWT algorithm, which was adopted from SampleInBall algorithm in Dilthium, and the constant weight word sampler [56]. We verified that the new sampler runs at a constant time with a fixed number of hash calls. With the new hamming weight sampler and the partly optimized reference code, SMAUG is now 17% faster than the previous version.

Second, we give an additional security analysis for the choice of the approximate discrete Gaussian sampler. Using the Rényi divergence, it is theoretically guaranteed that the security loss comes from the approximation is minute.

Lastly, we give an additional justification for the decryption failure probability against the state-of-the-art decryption failure attacks, asserting that the current failure probability of SMAUG is already low enough due to the attack scenarios.

May 23, 2023 (version 1.0) First, we updated the Python script for DFP computation as it was computing the decryption failure probability (DFP) wrongly. Note that the script was missing in the submission file, but included in our website. The parameter sets for NIST's security levels 3 and 5 had higher DFPs than they were reported in the KpqC round 1 submission. As a result, the parameter sets are updated.

Second, we additionally compress the ciphertexts. As compression makes the error larger, we exploit the balance between the sizes and DFP.

Third, we put additional cost estimations on some algebraic and topological attacks: Arora-Ge [8], Coded-BKW [39], and Meet-LWE [50] attacks. We note that the previous parameter sets were all in a secure region against these attacks; however, for the new parameter sets, we aim to have more security margins. We

put our code for estimating the cost of the Meet-LWE attack in the Python script.

Based on the above three updates, we changed our recommended parameter sets. As $q = 1024$ is not available anymore for sufficient DFPs in the security levels 3 and 5, we move to $q = 2048$ for those levels, resulting in slightly larger public key and secret key sizes. The ciphertext sizes are decreased by at most 96 bytes.

We also update the reference implementation to have a constant running time with much faster speed. It is uploaded to our website: `kpqc.cryptolab.co.kr/smaug-t`.

1 Introduction

SMAUG-T is an efficient post-quantum key encapsulation mechanism whose security is based on the hardness of the lattice problems. The IND-CPA security of SMAUG-T.PKE relies on the hardness of MLWE (Module-Learning with Errors) problem and MLWR (Module-Learning with Rounding) problem, which implies the IND-CCA2 security of SMAUG-T.KEM.

Our SMAUG-T.KEM scheme follows the approaches in recent constructions of post-quantum KEMs such as Lizard [26] and RLizard [48]. SMAUG-T.KEM base their security on the module variant lattice problems: the public key does not leak the secret key information by the hardness of MLWE problem, and the ciphertext protects sharing keys based on the hardness of MLWR problem. SMAUG-T consists of underlying public key encryption (PKE) schemes SMAUG-T.PKE , which turn into SMAUG-T.KEM via Fujisaki-Okamoto 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:

- The shared key should have a large enough entropy, at least ≥ 256 bits, to prevent Grover's search [38].
- 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 [28, 44].
- As KEMs are widely used in various devices and systems, countermeasures against implementation-specific attacks should also be considered. Especially combined with DFP, using error correction code (ECC), which recovers the message with a number of erroneous bits to reduce decryption failures, should be avoided since masking such ECC against side-channel attacks is a challenging problem.

MLWE and MLWR. SMAUG-T is constructed on the hardness of MLWE and MLWR problems and follow the key structure of Lizard [26] and Ring-Lizard (RLizard) [48]. Since LWE problem has been a well-studied problem for the last two decades, there are many LWE-based schemes (e.g., FrodoKEM [17]). Ring and module LWE problems are variants defined over structured lattices and regarded as hard as LWE. Many schemes base their security on RLWE/MLWE (e.g., NewHope [5], Kyber [16] 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 [15], 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 [35, 36]. Line of works on FO transforms in the quantum random oracle model [14, 42, 45, 54] 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 ($\mathsf{FO}^{\not\perp}_m$) following [43].

Sparse secret key and ephemeral key. 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 [25]. We take advantage of the sparsity, e.g., significantly smaller secret keys and faster multiplications. In particular, the small secret makes SMAUG-T more feasible in IoT devices having restricted resources.

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 2 moduli makes it hard to apply Number Theoretic Transform (NTT) on the polynomial multiplications. However, small enough moduli and polynomial degrees enable SMAUG-T to achieve faster speed.

Negligible decapsulation failures. Since we base the security on the lattice problems, noise is inherent. The decryption result of a SMAUG-T.PKE ciphertext could be different from the original message but with negligible probability, say decryption failure probability (DFP). We balance the sizes, DFP, and security of SMAUG-T by fine-tuning the parameters. In particular, additional parameter set TiMER uses the D2 encoding and error reconciliation used in NewHope [5, 53].

Parameters sets	TiMER	SMAUG-T128	SMAUG-T192	SMAUG-T256
Target security	1	1	3	5
(n, k)	$(256, 2)$	$(256, 2)$	$(256, 3)$	$(256, 5)$
(q)	(1024)	(1024)	(2048)	(2048)
(p, p')	$(256, 8)$	$(256, 32)$	$(256, 256)$	$(256, 64)$
Classical core-SVP hardness	120.0	120.0	181.7	264.5
Quantum core-SVP hardness	105.6	105.6	160.9	245.2
Decryption failure probability	-132	-120	-136	-167
Secret key size	136	176	236	218
Public key size	672	672	1088	1792
Ciphertext size	608	672	1024	1472

Table 1: Security and sizes for our parameter sets.

We give estimated security and sizes for our parameter sets in Table 1. The complete parameter sets are given in Section 5. The sizes are given in bytes, and DFP is given logarithm base two. We include the security estimator of SMAUG-T in the reference code package on our website: `kpqc.cryptolab.co.kr/smaug-t`.

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 that avoid using the error correction codes. In terms of DFP, SMAUG-T achieves low enough DFP, which is similar to that of Saber. SMAUG-T parameter sets do not use error correction code (ECC) to avoid possible side-channel attacks, while 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. This makes it much easier to implement and secure against physical attacks. We give the constant-time C reference code and AVX optimized code, which proves the completeness and shows the 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 discrete Gaussian distribution with standard deviation σ as $\mathcal{D}_{\mathbb{Z},\sigma}$. We define Rényi divergence of order α between to 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 $\mathrm{Supp}(D) = \{x \in D : D(x) \neq 0\}$.

2.2 Lattice assumptions

We first 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*

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

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*

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

2.3 Public key encryption and key encapsulation mechanism

We then 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 with respect to the attacks against PKE, namely the INDistinguishability under Chosen Plaintext Attacks ($\mathsf{IND\text{-}CPA}$).

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:*

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

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 with respect to the attacks against KEM, namely the INDistinguishability under Chosen Plaintext Attacks ($\mathsf{IND\text{-}CPA}$) as in PKE and the INDistinguishability under (adaptively) Chosen Ciphertext Attacks ($\mathsf{IND\text{-}CCA}$).

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.

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 $\mathcal{O}$ and (quantum) access to a random oracle H, it holds that $\mathsf{Adv}_{\mathcal{S}}^{\mathsf{ATK}}(\mathcal{A}) < \epsilon$, where*

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

2.4 Fujisaki-Okamoto transform

Fujiskai and Okamoto proposed a novel generic transform [35, 36] 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 [30] and revisited by Hofheinz et al. [42], Bindel et al. [13], and Hövelmanns et al. [43].

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. [13] allowing the KEMs constructed over non-perfect PKEs to have IND-CCA security.

Theorem 1 ([13], Theorem 1 & 2). *Let G and H be quantum-accessible random oracles, and the deterministic PKE is ϵ-injective. Then the advantage of* IND-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 IND-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 hereby 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 [29].

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

Fig. 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.

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.

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 [19,24,40] 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 binary (ternary, resp.), each coefficient of the decryption error is an addition (signed addition, resp.) 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.

Other major advantages of sparse secrets include reducing the secret key size and enabling fast polynomial multiplication. As the coefficients of the secret key are sparse with a fixed hamming weight, we can store only the information of the non-zero coefficients. We can further use this structure for the polynomial multiplications, which we will describe in Section 3.4.

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 [25] for asymptotic security based on the reductions to worst-case lattice problems, and [12, 33, 58] 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 [20]. The MLWR problem also has a simple reduction from MLWE independent of the secret distribution, and its concrete security is heuristically discussed in [29].

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 [2],

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 hybrid of the SampleInBall algorithm in Dilithium [31] and the CWW (constant weight word) sampler in BIKE [56], which have a constant running time. However, when applying the SampleInBall sampling from BIKE directly, there was a need to reduce the sampling error inevitably arising from the Fisher-Yates Shuffle. Therefore, we eliminate the cause of this deviation by using division operations and a rejection technique. A detailed algorithm is given in Figure 2, which samples a ternary polynomial vector having a hamming weight of h.

$\mathsf{HWT}_h(\mathsf{seed})$: ▷ $\mathsf{seed} \in \{0,1\}^{256}$

1: $\mathsf{idx} = 0$
2: $\mathsf{i} = n - h$
3: (buf, rand) ← PRF(seed)
4: **for** idx from 0 to sizeof(buf) **do**
5: div = 0xffffffff / i
6: remain = 0xffffffff - div * i
7: remain = remain + 1 ▷ $\mathsf{buf[idx]} \in \{0,1\}^{32}$
8: **if** 0xffffffff - buf[idx] ≥ remain and n ≥ i **then**
9: degree = buf[idx] / i
10: res[i] = res[degree]
11: res[degree] = $(-1)^{\mathsf{rand[idx]}}$ ▷ $\mathsf{rand[idx]} \in \{0,1\}$
12: i = i+1
13: **return** `convToIdx`(res) ▷ Storing the indexes

Fig. 2: Hamming weight sampler, HWT_h.

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 [17]. 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 4.

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 [4]. 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 (SMAUG-T128, 192, 256, resp.), the advantage increases from $2^{-120.0}$ ($2^{-120.0}$, $2^{-181.7}$ and $2^{-264.5}$, resp.) to $2^{-116.0}$ ($2^{-118.2}$, $2^{-176.9}$ and $2^{-260.2}$, resp.) with $\alpha = 75$ (200, 75 and 200, resp.). 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.

In addition, by using one more bit for dGaussian algorithm, we can decrease the advantage to $2^{-119.7}$ ($2^{-119.6}$, $2^{-181.3}$ and $2^{-263.6}$, resp.) with $\alpha = 500$. This modification will slightly decrease only the speed of key generation by less than 1.1x. We 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 [17, 47] using Rényi divergence. We then deploy the Quine-McCluskey method[6] 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. The algorithms are easily parallelizable and suitable for IoT devices as their memory requirement is low.

We describe dGaussian with $\sigma = 1.0625$ in Figure 3 and $\sigma = 1.453713$ in Figure 4.

3.4 Polynomial multiplication using sparsity

SMAUG-T uses the power-of-two moduli to ease the correct scaling and roundings. However, this makes the polynomial multiplications hard to benefit from

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

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

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

1: $s = s_1s_0 = 00 \in \{0,1\}^2$
2: $s_0 = x_0x_1x_2x_3x_4x_5x_7\overline{x_8}$
3: $s_0 \mathrel{+}= (x_0x_3x_4x_5x_6x_8) + (x_1x_3x_4x_5x_6x_8) + (x_2x_3x_4x_5x_6x_8)$
4: $s_0 \mathrel{+}= (\overline{x_2}x_3\overline{x_6}x_8) + (\overline{x_1}x_3\overline{x_6}x_8)$
5: $s_0 \mathrel{+}= (x_6x_7\overline{x_8}) + (\overline{x_5}x_6x_8) + (\overline{x_4x_6}x_8) + (\overline{x_7}x_8)$
6: $s_1 = (x_1x_2x_4x_5x_7x_8) + (x_3x_4x_5x_7x_8) + (x_6x_7x_8)$
7: $s = (-1)^{x_9} \cdot s$ ▷ $\cdot$ is the arithmetic multiplication
8: **return** s

Fig. 3: Discrete Gaussian sampler with $\sigma = 1.0625$, $\mathsf{dGaussian}_\sigma$.

$\mathsf{dGaussian}_{1.453713}(x)$:

Require: $x = x_0x_1x_2x_3x_4x_5x_6x_7x_8x_9x_{10} \in \{0,1\}^{11}$

1: $s = s_2s_1s_0 = 000 \in \{0,1\}^3$
2: $s_0 = (x_0x_1x_2x_3x_5x_7x_8) + (x_1x_2x_3x_5\overline{x_6}x_7x_9) + (\overline{x_1x_2x_3}x_6x_7x_8)$
3: $s_0 \mathrel{+}= (\overline{x_1}x_2x_3x_5x_8x_9) + (\overline{x_0}x_2x_3x_5x_8x_9)$
4: $s_0 \mathrel{+}= (x_4x_5\overline{x_6}x_7x_9) + (x_3x_4x_8\overline{x_9}) + (\overline{x_5}x_6x_7x_8) + (\overline{x_4}x_6x_7x_8) + (\overline{x_4x_5x_8}x_9)$
5: $s_0 \mathrel{+}= (x_5x_8\overline{x_9}) + (x_6x_8\overline{x_9}) + (x_7x_8\overline{x_9}) + (\overline{x_7x_8}x_9) + (\overline{x_6x_8}x_9)$
6: $s_1 = (x_0x_1x_4\overline{x_5}x_6x_7x_9) + (x_2x_4\overline{x_5}x_6x_7x_9) + (x_3x_4\overline{x_5}x_6x_7x_9) + (x_5x_6x_7\overline{x_8}x_9)$
7: $s_1 \mathrel{+}= (\overline{x_1x_2x_3}x_8x_9) + (\overline{x_7}x_8x_9) + (\overline{x_6}x_8x_9) + (\overline{x_5}x_8x_9) + (\overline{x_4}x_8x_9)$
8: $s_2 = (x_1x_4x_5x_6x_7x_8x_9) + (x_2x_4x_5x_6x_7x_8x_9) + (x_3x_4x_5x_6x_7x_8x_9)$
9: $s = (-1)^{x_{10}} \cdot s$ ▷ $\cdot$ is the arithmetic multiplication
10: **return** s

Fig. 4: Discrete Gaussian sampler with $\sigma = 1.453713$, $\mathsf{dGaussian}_\sigma$.

Number Theoretic Transform (NTT). To address this issue, we propose a new polynomial multiplication that takes advantage of sparsity, which we adapt from [1, 48]. Our new multiplication, given in Figure 5, is constant-time and is faster than the previous approach. We also use a secret storing technique like RLizard, where only the degrees of non-zero coefficients are stored in the secret key and directly used in polynomial multiplications.

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 [13, 42, 43].

poly_mult_add$(a, b, \text{neg_start})$: $\triangleright\ a \in \mathcal{R}_q,\ b \in \mathcal{S}_\eta$

1: $c = 0$
2: **for** i from 0 to neg_start $- 1$ **do**
3: degree $= b[i]$
4: **for** j from 0 to $n - 1$ **do**
5: $c[\text{degree} + j] = c[\text{degree} + j] + a[j]$;
6: **for** i from neg_start to $\text{len}(b) - 1$ **do**
7: degree $= b[i]$
8: **for** j from 0 to $n - 1$ **do**
9: $c[\text{degree} + j] = c[\text{degree} + j] - a[j]$;
10: **for** j from 0 to $n - 1$ **do**
11: $c[j] = c[j] - c[n + j]$;
12: **return** c

Fig. 5: Polynomial multiplication using sparsity.

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 [5]. This technique lowers the decryption failure rate and reduces the ciphertext size by changing the error bound. In Figure 6, we give the description of D2.

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: mask $\leftarrow ((\mu[i] \gg j)\ \&\ 1$
5: $\upsilon_{8*i+j+0} \leftarrow$ mask $\&\ (q/2)$
6: $\upsilon_{8*i+j+128} \leftarrow$ mask $\&\ (q/2)$
7: **return** $\mathsf{v} \in \mathcal{R}_q$

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

1: $\mu \leftarrow \{0, \ldots, 255\}^{16}$
2: **for** i from 0 to 255 **do**
3: $t \leftarrow |(\upsilon_{i+0} \bmod q) - (q-1)/2|$
4: $t \leftarrow t + |(\upsilon_{i+128} \bmod q) - (q-1)/2|$
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** $\mu \in \{0, \ldots, 255\}^{16}$

Fig. 6: Description of D2 encoding

To ensure robustness against errors, each bit of the 128-bit message $\mu \in \{0, ..., 255\}^{16}$ is encoded into 2 coefficients by $\mathsf{D2Enc}$. The decoding function $\mathsf{D2Dec}$ maps 2 coefficients back to the original key bit. For example, for $n = 256$, take 2 coefficients (each in the range $\{0, ..., q-1\}$), 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 $\mathbf{s}$ (resp. $\mathbf{r}$) with hamming weight $h = h_s$ (resp. $h = h_r$) from $\mathsf{seed}_{\mathsf{sk}}$ (resp. $\mathsf{seed}_{\mathbf{r}}$).

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

1: $\mathsf{seed} \leftarrow \{0,1\}^{256}$
2: $(\mathsf{seed}_{\mathbf{A}}, \mathsf{seed}_{\mathsf{sk}}, \mathsf{seed}_{\mathbf{e}}) \leftarrow \mathsf{PRF}(\mathsf{seed})$
3: $\mathbf{A} \leftarrow \mathsf{expandA}(\mathsf{seed}_{\mathbf{A}}) \in \mathcal{R}_q^{k \times k}$
4: $\mathbf{s} \leftarrow \mathsf{HWT}_{h_s}(\mathsf{seed}_{\mathsf{sk}}) \in S_\eta^k$
5: $\mathbf{e} \leftarrow \mathsf{dGaussian}_{\sigma}(\mathsf{seed}_{\mathbf{e}}) \in \mathcal{R}^k$
6: $\mathbf{b} = -\mathbf{A}^\top \cdot \mathbf{s} + \mathbf{e} \in \mathcal{R}_q^k$
7: **return** $\mathsf{pk} = (\mathsf{seed}_{\mathbf{A}}, \mathbf{b})$, $\mathsf{sk} = \mathbf{s}$

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

1: $\mathbf{A} = \mathsf{expandA}(\mathsf{seed}_{\mathbf{A}})$
2: **if** $\mathsf{seed}_{\mathbf{r}}$ is not given **then** $\mathsf{seed}_{\mathbf{r}} \leftarrow \{0,1\}^{256}$
3: $\mathbf{r} \leftarrow \mathsf{HWT}_{h_r}(\mathsf{seed}_{\mathbf{r}}) \in S_\eta^k$
4: $\mathbf{c}_1 = \lfloor p/q \cdot \mathbf{A} \cdot \mathbf{r} \rceil \in \mathcal{R}_p^k$
5: $c_2 = \lfloor p'/q \cdot \langle \mathbf{b}, \mathbf{r} \rangle + p'/t \cdot \mu \rceil \in \mathcal{R}_{p'}$
6: **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'

Fig. 7: Description of SMAUG-T.PKE

One of the four parameter sets of SMAUG-T, namely, TiMER, has slightly different features compared to SMAUG-T128 parameter set:

- Reduced message space: from $\{0,1\}^{256}$ to $\{0,1\}^{128}$ for D2 encoding, i.e., $\mu \leftarrow \mathsf{D2Enc}(\mu)$.
- After decryption, the message adjustment process changed from rounding to D2Dec.

The rest of the parts, including the key generations, are done exactly the same as the description in Figure 7.

We then prove the completeness of SMAUG-T.PKE.

Theorem 2 (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* $t \,|\, p \,|\, q$ *and* $t \,|\, p' \,|\, q$. *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

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

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, since the D2 encoding output can be seen as the message μ in the above proof. The only assumption we require is the completeness of D2 encoding.

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.

KeyGen(1^λ):

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

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

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

Decap(sk, ct): ▷ $\mathsf{sk} = (\mathsf{sk}', d)$

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'

Fig. 8: Description of SMAUG-T.KEM

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 [43] 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 [42] 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 3 (Completeness of SMAUG-T.KEM). *We borrow the notations and assumptions from Theorem 2 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 2) 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 [54] or non-tight [13,42,43] in the QROM. Therefore, we prove the IND-CCA security of SMAUG-T.KEM based on the non-tight QROM reduction of [13] as explained in Section 2 by proving the IND-CPA security of SMAUG-T.PKE.

Theorem 4 (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,*

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

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 public key is changed into $(\mathbf{A}, \mathbf{b})$,
- 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

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

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

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

□

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 [42] in the classical random oracle model. Theorem 1 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 recommended parameter (SMAUG-T) and additional parameter (TiMER) 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 [22, 41, 55]. BKZ has been used in various lattice-based schemes [3, 16, 31, 34, 59]. The security of the schemes is generally estimated as the time complexity of BKZ in core-SVP hardness introduced in [5]. It depends on the block size β of BKZ reporting the best performance. According to Becker et al. [10] and Chailloux et al. [21], 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 [2] 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.

We also focus on the attacks not considered in the lattice estimator, specifically those that target sparse secret, such as Meet-LWE [50] 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.

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 [2]. 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[7] based on May's analysis [50], best among Rep-1 and Rep-2.

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

Parameters sets		TiMER	SMAUG-T128	SMAUG-T192	SMAUG-T256
Target security		1	1	3	5
Classical core-SVP		120.0	120.0	181.7	264.5
Algebraic & Combinatorial attacks					
Arora-Ge	time	653.8	741.3	983.4	-
	(mem)	(317.2)	(598.0)	(636.5)	-
BKW	time	135.3	144.7	202.0	274.6
	(mem)	(123.2)	(133.7)	(190.7)	(256.9)
Meet-LWE	time	144.3	164.3	213.8	283.2
	(mem)	(123.7)	(143.7)	(192.4)	(254.7)

Table 2: Attack costs beyond Core-SVP.

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. [9] provided the reduction from the MLWE problem to the MLWR problem, which was subsequently improved in [6,7,15]. 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

Parameters sets	TiMER	SMAUG-T128	SMAUG-T192	SMAUG-T256
Security level	1	1	3	5
n	256	256	256	256
k	2	2	3	5
(q, p)	$(1024, 256)$	$(1024, 256)$	$(2048, 256)$	$(2048, 256)$
(p', t)	$(8, 2)$	$(32, 2)$	$(256, 2)$	$(64, 2)$
(h_s, h_r)	$(100, 132)$	$(140, 132)$	$(198, 151)$	$(176, 160)$
σ	1.453713	1.0625	1.453713	1.0625
Classical core-SVP	120.0	120.0	181.7	264.5
Quantum core-SVP	105.6	105.6	160.9	245.2
Beyond core-SVP	135.3	144.7	202.0	274.6
DFP	-132.9	-119.6	-136.1	-167.2
Secret key	136	176	236	218
Public key	672	672	1088	1792
Ciphertext	608	672	1024	1472

Table 3: The NIST security level, selected parameters, classical and quantum core-SVP hardness and security beyond core-SVP (see Section 5.1.2), DFP (in $\log_2$), and sizes (in bytes) of SMAUG-T.

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, h_r, and σ, and search for the parameters with enough security. Once we have them, we compute the DFP. If it is low enough, we can choose the compression parameter p', but if it is not, we continue searching for appropriate parameters. The compression factor p' can be set to a small integer if the DFP is low enough. Else, we can keep $p' = 256$ as in the level-3 parameter, and not compress.

Table 3 shows the three recommended parameter sets and an additional parameter set of SMAUG-T, corresponding to NIST's security levels 1, 3, and 5. For security levels 3 and 5, we can not find the parameters for $q = 1024$, so we use $q = 2048$. Especially, the standard deviation $\sigma = 1.0625$ is too low for security level 3, so we move to $\sigma = 1.453713$. For the level-5 parameters set, we use $k = 5$ since $k = 4$ is too small for enough security.

TiMER, an additional parameter set, maximizes the efficiency of SMAUG-T128. It has a 64-byte smaller ciphertext size than SMAUG-T128 and 32 bytes smaller than the state-of-art scheme(aspect ciphertext size), lightsable [51], and TiGER [52]. 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. DFP, after reducing p' and still small enough, was used as a trade-off to improve performance. In other words, to improve performance while maintaining security strength, the standard deviation σ was increased from 1.0625 to 1.453713, and the Hamming weight of the secret key h_s was reduced from 140 to 100. Because SMAUG-T uses sparse polynomial multiplication, as shown in Figure 5, it computes n times less than the reduction

of h_s. It speeds up the overall computation, even though it adds a small overhead required for D2 encoding and error reconciliation.

The core-SVP hardness is estimated via the lattice estimator [2] using the cost model "ADPS16" introduced in [5] and "MATZOV" [49]. 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 [2] 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. We note that these attacks require a minimum memory size of 2^{130} to 2^{260}.

5.3 Decryption failure probability

As our primary goal is to push the efficiency of the lattice-based KEMs toward the limit while keeping roughly the same level of security, we follow the frameworks given in the NIST finalist Saber. In particular, we set the DFP to be similar to or lower than that of Saber's, except for TiMER parameter set.

The impact of DFP on the security of KEM is still being investigated. However, we can justify our decision to follow Saber's choice and why it is sufficient for real-world scenarios. To do this, 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 hundreds 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, allowing 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 $2^{64+33+8+8} \cdot 2^{-20} = 2^{93}$. Based on the best-known (multi-target) attacks for Saber [27, Figure 6 (a)], the quantum cost for finding a single failing ciphertext of SMAUG-T192 is much higher than 2^{160}, as desired[8]. For security level 5, we refer to Figure 7(a) in [27], which shows that the quantum cost for finding a single failure is much higher than 2^{245}. 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.

[8] Specifically, the number of observable failures must be larger than $1/\beta$ in [27] to observe at least one failing ciphertext. That is, β should be larger than 2^{93}. The quantum cost is $1/\beta\sqrt{\alpha}$.

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`. Additionally, the TiMER parameter set, designed for lightweight environments, is available in the `additional_implementation`. Our implementations, along with the supporting scripts, are accessible on our website: `www.kpqc.cryptolab.co.kr/smaug-t`.

6.1 Implementation considerations

The most critically time-consuming component in SMAUG-T is the symmetric primitive. We chose SHA3 as the symmetric variant, which occupies about 30% to 40% of the cycles according to the reference implementation. Being based on the Keccak permutation, SHA3 is not the fastest algorithm in software. Thus its usage may impose performance constraints compared to 90s symmetrics (AES, SHA2). However, similar to how ARM provides hardware acceleration for SHA3 on ARMv8 processors, this is not expected to be a problem in the future.

To measure the achieved performance of SMAUG-T, we also provide an implementation that uses 90s symmetrics. This implementation serves as benchmark code to demonstrate optimized performance by utilizing AES and SHA2 instead of other symmetric algorithms. Consequently, this leads to differences in the test vector when compared to the reference implementation. The optimized implementation using 90s symmetric can be found at `optimized_implementation/kem-90s`.

6.2 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 and Saber[9]. 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 9.4.0, and the compiler flags as indicated in the CMakeLists included in the submission package.

[9] From `github.com/pq-crystals/kyber` (`518de24`) and `github.com/KULeuven-COSIC/SABER` (`f7f39e4`), respectively.

Schemes	Cycles			Cycles (ratio)		
	KeyGen	Encap	Decap	KeyGen	Encap	Decap
Kyber512	131560	162472	189030	1.87	2.16	1.94
LightSaber	93752	122176	133764	1.33	1.63	1.37
SMAUG-T128	70398	75082	97368	1	1	1
TiMER	70348	71748	90978	1	0.96	0.93
Kyber768	214160	251308	285378	1.57	2	1.78
Saber	18722	224686	239590	1.37	1.78	1.49
SMAUG-T192	136436	126114	160354	1	1	1
Kyber1024	332470	371854	415498	1.43	1.60	1.53
FireSaber	289278	347900	382326	1.25	1.49	1.41
SMAUG-T256	231824	232854	271794	1	1	1

Table 4: Median cycle counts of 1000 executions for Kyber, Saber, and SMAUG-T (and their ratios). The C implementations without AVX optimizations.

In the optimized implementation, we offer two options for the symmetric primitives. The default implementation located in `optimized_implementation/kem` uses all symmetric primitives identically to the reference implementation. In the 90s symmetric implemenation found in `optimized_implementation/kem-90s`, we instantiate the hash functions G, H, the extendable output function XOF, and the pseudo random function PRF with the following symmetric primitives: G is instantiated with SHA2-512, H is instantiated with SHA2-256, and both XOF and PRF are instantiated with AES.

SMAUG-T optimized implementation using AVX2 intrinsics achieved a speed up of about x1.7 - x1.8, while the optimized implementation using 90s symmetric achieved a speed up of about x2.5 - x3. All measurement methods and conditions are identical to those of Table 4.

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 [11, 18]. 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. Specifically, the characteristic designs are dGaussian sampling, Sparse Hamming weight sampling, D2 encoding and error reconciliation, and Sparse polynomial multiplication. Additionally, 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. So, we discuss the security of SMAUG-T against physical attacks based on power/EM.

In this section, Continued research on side-channel analysis is necessary, but in the first round of KpqC, SMAUG and TiGER responded to side-channel analysis for characteristic designs.

7.1 Hamming weight sampler

In the KpqC round 1 SMAUG scheme, rejection sampling generated uniform distribution. This sampling had the vulnerability of constant-time implementation. Heesoek Kim reported that this vulnerability could cause a timing attack. To resolve this problem, SMAUG v2.0 utilized the SampleInBall from BIKE.

However, SampleInBall is based on the Fisher-Yates Shuffling, and it has a non-negligible probability deviation where the probability of selecting a random number changes as the range of random selection changes. While BIKE proved this issue to be negligible in BIKE parameters when applied directly to SMAUG-T, we found there is a significant probability deviation when sampling a uniform distribution [56].

This error is based on complete uniform sampling and cannot be performed without rejection when the power of 2 is not a multiple of $i+1$. SMAUG v2.0 is applied as is, and a probability deviation as described by the equation below occurs.

$$e_t = 2^{32} \mod i$$

$$\prod_{i=380}^{511} 1 - \frac{1}{i+1}(1+e_t) = \prod_{i=380}^{511} (\frac{i}{i+1} - \frac{e_t}{i+1}) \approx \prod_{i=380}^{511} (\frac{i}{i+1})(1 - \sum_{i=380}^{511} \frac{e_t}{i+1})$$

Calculating the above equation using the SMAUG-T parameters, it can be shown that the random number distribution is not completely uniform, with a deviation of about 2^{-28}. In BIKE, it was proven that the deviation caused by the same reason is safe against IND-CCA attacks. However, this is not the case in SMAUG-T.

Therefore, we eliminated the cause of this deviation by using division operations and rejection technique.

7.2 dGaussian$_\sigma$ sampler

Previous PQC algorithms utilizing Gaussian errors have employed various Gaussian samplers. However, designing Gaussian samplers that operate in constant time is challenging, and BLISS has suffered from timing attacks [37]. We adopted dGaussian$_\sigma$, a constant-time implementation well-known for its efficacy, into SMAUG-T to mitigate timing attacks.

However, power/EM-based SCA issues for dGaussian$_\sigma$ arose during KpqC round 1. 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 dGaussian$_\sigma$ to prevent these attacks. In the public key generation process of SMAUG-T, the 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. [46] 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 Fig. 3 and 4 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. Table 5 shows the overhead resulting from the application of countermeasures.

Schemes	TiMER	SMAUG-T128	SMAUG-T192	SMAUG-T256
Original	70348	70398	136436	231824
Countermeasure	122258	124426	241444	491184
Overhead	76.7%	76.9%	111.8%	73.7%

Table 5: Results of the application of the countermeasure to clustering SCA in the dGaussian$_\sigma$ - Median cycle counts of 1000 executions. The reference code is in `additional_implementation/side_channel_countermeasure`

Overhead occurs only during key generation, with no involvement in the encryption and decryption. 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.

7.3 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.

However, during the KpqC round 1, Hee-Seok Kim 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.

Acknowledgments. Part of this work was done while Dongyeon Hong was in CryptoLab Inc during KpqC round 1. Part of this specification is from its conference version [23].

References

1. Akleylek, S., Alkım, E., Tok, Z.Y.: Sparse polynomial multiplication for lattice-based cryptography with small complexity. The Journal of Supercomputing **72**, 438–450 (2016)
2. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Journal of Mathematical Cryptology **9**(3), 169–203 (2015)
3. 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
4. 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)
5. 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)
6. 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
7. 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)
8. Arora, S., Ge, R.: New algorithms for learning in presence of errors. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) Automata, Languages and Programming. pp. 403–415. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
9. 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)
10. 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
11. 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
12. 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, Heidelberg (Nov 2022). https://doi.org/10.1007/978-3-031-22301-3_9
13. 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, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-36033-7_3
14. 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)
15. 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)

16. 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)
17. 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
18. 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
19. 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 22. LNCS, vol. 13269, pp. 521–541. Springer, Heidelberg (Jun 2022). https://doi.org/10.1007/978-3-031-09234-3_26
20. 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
21. Chailloux, A., Loyer, J.: Lattice sieving via quantum random walks. In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT. pp. 63–91 (2021)
22. 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, Heidelberg (Dec 2011). https://doi.org/10.1007/978-3-642-25385-0_1
23. Cheon, J.H., Choe, H., Hong, D., Yi, M.: Smaug: Pushing lattice-based key encapsulation mechanisms to the limits. Cryptology ePrint Archive, Paper 2023/739 (2023), https://eprint.iacr.org/2023/739
24. 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, Heidelberg (Apr / May 2018). https://doi.org/10.1007/978-3-319-78381-9_14
25. 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, Heidelberg (Nov / Dec 2017). https://doi.org/10.1007/978-3-319-53177-9_3
26. 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, Heidelberg (Sep 2018). https://doi.org/10.1007/978-3-319-98113-0_9
27. 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, Heidelberg (Mar 2022). https://doi.org/10.1007/978-3-030-97121-2_1
28. 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)

29. 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, Heidelberg (May 2018). https://doi.org/10.1007/978-3-319-89339-6_16
30. 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)
31. Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Dilithium: A lattice-based digital signature scheme. IACR TCHES **2018**(1), 238–268 (2018). https://doi.org/10.13154/tches.v2018.i1.238-268, https://tches.iacr.org/index.php/TCHES/article/view/839
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. 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, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-65277-7_20
34. 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)
35. 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, Heidelberg (Aug 1999). https://doi.org/10.1007/3-540-48405-1_34
36. 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
37. 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)
38. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. pp. 212–219 (1996)
39. Guo, Q., Johansson, T., Stankovski, P.: Coded-bkw: Solving lwe using lattice codes. In: Annual Cryptology Conference. pp. 23–42. Springer (2015)
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, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46800-5_25
41. 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)
42. 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, Heidelberg (Nov 2017). https://doi.org/10.1007/978-3-319-70500-2_12

43. 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, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45388-6_14
44. 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)
45. 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)
46. 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)
47. 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, Heidelberg (May 2014). https://doi.org/10.1007/978-3-642-55220-5_14
48. 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)
49. MATZOV: Report on the Security of LWE: Improved Dual Lattice Attack (Apr 2022). https://doi.org/10.5281/zenodo.6493704
50. 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, Heidelberg, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84245-1_24
51. Mera, J.M.B., Karmakar, A., Kundu, S., Verbauwhede, I.: Scabbard: a suite of efficient learning with rounding key-encapsulation mechanisms. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 474–509 (2021)
52. Park, S., Jung, C.G., Park, A., Choi, J., Kang, H.: Tiger: Tiny bandwidth key encapsulation mechanism for easy migration based on rlwe(r). Cryptology ePrint Archive, Paper 2022/1651 (2022), https://eprint.iacr.org/2022/1651
53. 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)
54. 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, Heidelberg (Apr / May 2018). https://doi.org/10.1007/978-3-319-78372-7_17
55. Schnorr, C.P., Euchner, M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Mathematical programming **66**(1), 181–199 (1994)
56. Sendrier, N.: Secure sampling of constant-weight words – application to bike. Cryptology ePrint Archive, Paper 2021/1631 (2021), https://eprint.iacr.org/2021/1631

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