SMAUG: Pushing Lattice-based Key Encapsulation Mechanisms to the Limits

Jung Hee Cheon[1,2], Hyeongmin Choe[1], Dongyeon Hong[2], and MinJune Yi[1]

[1] Seoul National University
{jhcheon, sixtail528, yiminjune}@snu.ac.kr
[2] CryptoLab Inc.
jjoker041@gmail.com

Abstract. Recently, NIST has announced Kyber, a lattice-based key encapsulation mechanism (KEM), as a post-quantum standard. However, it is not the most efficient scheme among the NIST's KEM finalists. Saber enjoys more compact sizes and faster performance, and Mera et al. (TCHES '21) further pushed its efficiency, proposing a shorter KEM, Sable. As KEM are frequently used on the Internet, such as in TLS protocols, it is essential to achieve high efficiency while maintaining sufficient security.

In this paper, we further push the efficiency limit of lattice-based KEMs by proposing SMAUG, a new post-quantum KEM scheme whose IND-CCA2 security is based on the combination of MLWE and MLWR problems. We adopt several recent developments in lattice-based cryptography, targeting the *smallest* and the *fastest* KEM while maintaining high enough security against various attacks, with a full-fledged use of sparse secrets. Our design choices allow SMAUG to balance the decryption failure probability and ciphertext sizes without utilizing error correction codes, whose side-channel resistance remains open.

With a constant-time C reference implementation, SMAUG 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 has the same ciphertext sizes but a larger public key, which gives a trade-off between the public key size versus performance; SMAUG has 39%-55% faster encapsulation and decapsulation speed in the parameter sets having comparable security.

Keywords: Key Encapsulation Mechanism, Public Key Encryption, Post Quantum Cryptography, Module Learning With Errors, Module Learning With Roundings.

1 Introduction

Recent advances in quantum computers raise the demand for quantum-resistant cryptographic protocols, i.e., the Post-Quantum Cryptographic (PQC) schemes. As a consequence, the American National Institute of Standards and Technology (NIST) has established a standardization process focusing on Public Key Encryption (PKE), digital signature, and Key Encapsulation Mechanism (KEM).

In particular, KEM is one of the most widely used algorithms over the Internet, such as in Transport Layer Security (TLS) protocols; however, the KEM currently used in the protocol is considered vulnerable to quantum attacks.

Various lattice-based KEMs [5,11,16,17,22,26,48,58] have been proposed and submitted to NIST PQC standardization to secure the Internet in the quantum world. During the standardization process, diverse techniques improving efficiency or security were introduced. In 2020, NIST selected Kyber [16], Saber [58], and NTRU [22] as the lattice-based KEM finalists, having enough efficiency and quantum security based on the Module Learning With Errors (MLWE), Module Learning With Roundings (MLWR), and NTRU problems, respectively. Recently Kyber is selected as a future standard among the candidates.

As of independent interest to NIST's standardization, the KEM's efficiency is crucial since it is executed and transmitted frequently on the Internet. In particular, the TLS protocols are also necessary for embedded devices, so the efficiency requirement has become even more pressing with the proliferation of the Internet of Things (IoT). To this end, some variants of Saber focusing on efficiency, Scabbard [52], have been recently proposed by Mera et al. Scabbard consists of three schemes based on the Ring Learning With Roundings (RLWR) or MLWR problems, Floreta, Espada, and Sable, each targeting HW/SW-efficient, parallelizable, and shorter KEM than Saber. In particular, Sable achieves the smallest public key and ciphertext sizes among the KEM schemes targetting the NIST's security level 1 and low enough Decryption Failure Probability (DFP).

1.1 Our results

In this work, we ask:

Can we further push the efficiency of the lattice-based KEMs to the limits?

Specifically, we propose a new lattice-based KEM, SMAUG, constructed based on both MLWE and MLWR problems. By bringing the MLWE-based public key generation and the sparse secret to Sable, SMAUG is able to further enhance efficiency. We aimed to achieve *the shortest* ciphertext among the LWE/LWR-based KEM schemes while maintaining the security level with *even better* performance.

The SMAUG. The design rationale of SMAUG aims 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 need to be short to minimize communication costs. We especially focus on the ciphertext size as it is more frequently transmitted.

- The key exchange protocol is frequently required on various personal devices, so a KEM algorithm with low computational costs is desirable.
- A compact secret key is beneficial in restricted settings such as embedded or IoT devices. Maintaining a secure zone is imperative to thwart physical attacks on the storage storing secret keys.

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.
- The low enough DFP is essential to avoid the attacks boosting the failure and exploiting the decryption failures [29, 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 Codes (ECC) on the message to reduce decryption failures should be avoided since masking ECC against side-channel attacks is yet remaining as a challenging problem.

	Sizes (bytes)			**Security**			**Cycle**		
Scheme	**sk**	**pk**	**ct**	**Lvl.**	**Sec.**	**DFP**	KeyGen	Encap	Decap
SMAUG-128	176	672	672	1	120	2^{-120}	77k	77k	92k
SMAUG-192	236	1088	1024	3	181	2^{-136}	153k	136k	163k
SMAUG-256	218	1792	1472	5	264	2^{-167}	266k	270k	305k

Table 1: Parameter sets of SMAUG for NIST's security levels 1, 3, and 5. Security (Sec.) is given in classical core-SVP hardness. One core of an Intel Core i7-10700k is used for cycle counts.

To achieve this goal, we exploit the possible combination of the known techniques in lattice-based cryptography, such as underlying lattice assumptions, ciphertext compression, Fujisaki-Okamoto (FO) transforms, and the secret and error distribution. We refer to Appendix A for the design choices of the recent LWE/LWR-based PKEs.

Among the possibilities on the choice of lattice assumptions, we conclude to use LWE-based key generation and LWR-based encapsulation with sparse secrets in module lattices. This choice allows SMAUG to enjoy the conservative secret key security from the hardness of the un-rounded MLWE problem and explore efficiency on encapsulation and decapsulation from the MLWR-based approach.

Sparse secret allows SMAUG to enjoy fast polynomial multiplications and small secret keys. The sparse secret is widely used in homomorphic encryption (HE) schemes to speed up the expensive homomorphic operations and to reduce the noise [24, 39], whose ability is attractive for efficient KEMs. By using the

SampleInBall algorithm of Dilithium [32], we can efficiently sample the sparse ternary secrets. Regarding security, the hardness reductions for sparse LWE and LWR problems [25,26] from LWE problem exists; however, the concrete security should be treated carefully[3].

We take the recent approaches in FO transform for key exchange in the Quantum Random Oracle Model (QROM) [43] and apply it to our IND-CPA PKE, SMAUG.PKE. We use the FO transform with implicit rejection and no ciphertext contributions ($\mathsf{FO}_m^{\not\perp}$).

We delicately choose three parameter sets for SMAUG regarding NIST's security levels 1, 3, and 5 (classical core-SVP hardness of 120, 180, and 260, respectively) and having DFP at least smaller than Saber.

Comparison to other KEMs. We compare SMAUG with the NIST-selected Kyber, one of the round 3 finalists Saber, and a variant Sable in Table 2.

	Sizes (bytes)			**Security**		**Cycle** (ratio)		
Schemes	**sk**	**pk**	**ct**	**Classic.**	**DFP**	KeyGen	Encap	Decap
NIST's security level 1 (120)								
Kyber512 [16]	1632	800	768	118	2^{-139}	1.70	2.10	2.03
LightSaber [58]	832	672	736	118	2^{-120}	1.21	1.58	1.44
LightSable [52]	800	608	672	114	2^{-139}	1.10	1.48	1.39
SMAUG-128	176	672	672	120	2^{-120}	1	1	1
NIST's security level 3 (180)								
Kyber768 [16]	2400	1184	1088	183	2^{-164}	1.38	1.84	1.75
Saber [58]	1248	992	1088	189	2^{-136}	1.21	1.64	1.47
Sable [52]	1152	896	1024	185	2^{-143}	1.10	1.48	1.39
SMAUG-192	236	1088	1024	181	2^{-136}	1	1	1
NIST's security level 5 (260)								
Kyber1024 [16]	3168	1568	1568	256	2^{-174}	1.25	1.38	1.36
FireSaber [58]	1664	1312	1472	260	2^{-165}	1.21	1.58	1.44
FireSable [52]	1632	1312	1376	223	2^{-208}	1.03	1.25	1.22
SMAUG-256	218	1792	1472	264	2^{-167}	1	1	1

Table 2: Comparison of KEM schemes with comparable efficiency and security. Security is given in the classical core-SVP hardness with DFP. The cycle counts are given relative to that of SMAUG's, reported in the same machine. The C implementations without AVX optimizations are used.

Compared to Kyber-512 [16], the NIST-selected standard targeting security level 1, SMAUG-128, has 16% and 12% smaller public key and ciphertext, respectively. The secret key size of SMAUG is tiny and ready to use, which enable efficient management of secure zone in restricted IoT devices. With high enough

[3] We used the lattice-estimator [2], from `github.com/malb/lattice-estimator` (commit `9687562`), while also considering other attacks targeting the sparsity.

security and low enough DFP, SMAUG further achieves 110% and 103% speed up in encapsulation and decapsulation.

Compared to LightSaber [58], one of the round 3 finalists with the security level 1, SMAUG-128, has 9% smaller ciphertext and the same public key size. Again, the secret key is significantly smaller than LightSaber, with a 58% and 44% speed up in encapsulation and decapsulation, respectively.

When compared to Sable [52], SMAUG-128 has the same ciphertext size but a larger public key size. It can be seen as a trade-off as SMAUG achieves 48% and 39% faster encapsulation and decapsulation speed with a significantly smaller secret key and 6 bits higher security.

In NIST's security levels 3 and 5, SMAUG similarly outperforms Kyber and provides a trade-off with Saber and Sable. For instance, SMAUG-128 has the same ciphertext size as level-3 Sable, a larger public key but a smaller secret key, and is faster than Sable. FireSable has a smaller ciphertext than SMAUG-256; however, it has a classical core-SVP hardness way lower than 260. We refer to Section 6.2 for detailed comparisons.

Paper organization. The rest of the paper is organized as follows. Section 2 defines the notations and summarizes the formal definitions of key encapsulation mechanisms with the relevant works. In Section 3, we introduce the design choices of SMAUG. In Section 4, we introduce SMAUG and its security proofs. We provide concrete security analysis and the recommended parameter sets in Section 5. Lastly, we give the performance result with comparisons to recent KEM schemes and the implementation details in Section 6.

Code availability. We place all software, consisting of the constant-time C reference code of SMAUG and the Python scripts used for security estimation, into the public domain: `www.kpqc.cryptolab.co.kr/smaug`.

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

2.2 Public key encryption and key encapsulation mechanism

We refer to Appendix B for the formalism of Public Key Encryption (PKE) and Key Encapsulation Mechanism (KEM). The advantage functions against IND-CPA and IND-CCA attacks are also defined in the appendix. Note that we only focus on the adaptive IND-CCA attacks, i.e., IND-CCA2 attacks. Here, we give the standard (quantum) security notions for PKE and KEM in the (Q)ROM.

Definition 1 ((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 $\mathsf{Adv}^{\mathsf{ATK}}_{\mathcal{S}}(\mathcal{A}) < \epsilon$, where*

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

2.3 Fujisaki-Okamoto transform

Fujiskai and Okamoto proposed a novel generic transform [36, 37] 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. [41], Bindel et al. [14], and Hövelmanns et al. [43].

The original FO transforms $\mathsf{FO}^{\perp}_m$ 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" ($\mathsf{FO}^{\not\perp}_m$) 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. [14] allowing the KEMs constructed over non-perfect PKEs to have IND-CCA security.

Theorem 1 ([14], 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*

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

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.

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

We construct SMAUG 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 [14, 41, 43].

3.2 MLWE public key and MLWR ciphertext

One of the core designs of SMAUG 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.

Public key. Public key of SMAUG 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. Since the matrix $\mathbf{A}$ is sampled uniformly, it can be stored and transmitted as a seed of an eXtendable Output Function (XOF).

Ciphertext. The ciphertext of SMAUG 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.

3.3 Sparse secret

The sparse secret is widely used in homomorphic encryption to reduce the noise propagation during the homomorphic operations [19,24,39] 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 also improves 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, and 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 nonzero coefficients. We can further use this structure for the polynomial multiplications, which we will describe in Section 3.5.

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 [13,34,57] 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 [30].

Since SMAUG uses a sparse secret key $\mathbf{s}$ and a sparse randomness $\mathbf{r}$, the security of SMAUG 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 can maintain high security, as in Kyber or Saber.

3.4 Sampling algorithms

We use the following algorithms for sampling the randomnesses used in SMAUG: expandA for sampling a uniform random matrix $\mathbf{A}$; HWT_h for sampling sparse secrets, i.e., the secret key $\mathbf{s}$ and the ephemeral secret $\mathbf{r}$ with fixed hamming weights h_s and h_r, respectively; and $\mathsf{dGaussian}_\sigma$ for sampling a discrete Gaussian error with standard deviation σ for MLWE sample.

Uniform random matrix sampler, expandA. We adopt the gen algorithm in Saber [58] for our uniform random matrix sampler expandA. A detailed algorithm is given in Appendix C, Figure 4.

Hamming weight sampler, HWT_h. Our hamming weight sampler, HWT_h, is adapted from the SampleInBall algorithm in Dilithium [32], having a secret-independent running time. It samples a ternary polynomial vector having a hamming weight of h. A detailed algorithm is given in Appendix C, Figure 5.

Discrete Gaussian sampler, dGaussian. Our discrete Gaussian sampler dGaussian is a constant-time sampler outputting a sample from discrete Gaussian distribution. It is constructed upon a Cumulative Distribution Table (CDT) but is not used during sampling, as it is expressed with bit operations.

To make our sampler, 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[4] 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.

The algorithms can be found in Appendix C. In addition, we refer to Appendix D for a detailed analysis of the impact of the narrow discrete Gaussian noise on security.

3.5 Polynomial multiplication using sparsity

SMAUG uses the power-of-two moduli to ease the correct scaling and roundings. However, this makes the polynomial multiplications hard to benefit from 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 1, is constant-time and is faster than the previous approach. We also use a similar secret storing technique as RLizard, where only the degrees of non-zero coefficients are stored in the secret key and directly used in polynomial multiplications.

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

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

$c = 0$
for i from 0 to neg_start $- 1$ **do**
    degree $= b[i]$
    for j from 0 to $n - 1$ **do**
        $c[\text{degree} + j] = c[\text{degree} + j] + a[j]$;
for i from neg_start to len$(b) - 1$ **do**
    degree $= b[i]$
    for j from 0 to $n - 1$ **do**
        $c[\text{degree} + j] = c[\text{degree} + j] - a[j]$;
for j from 0 to $n - 1$ **do**
    $c[j] = c[j] - c[n + j]$;
return c

Fig. 1: Polynomial multiplication using sparsity.

4 The SMAUG

4.1 Specification of SMAUG.PKE

We now describe the public key encryption scheme SMAUG.PKE in Figure 2 with the following building blocks:

- Extendable output function XOF 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. $\mathsf{seed}_{\mathbf{r}}$).

We then prove the completeness of SMAUG.PKE.

Theorem 2 (Completeness of SMAUG.PKE). *Let $\mathbf{A}$, $\mathbf{b}$, $\mathbf{s}$, $\mathbf{e}$, and $\mathbf{r}$ are defined as in Figure 2. 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.PKE in Figure 2 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

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

1: $\mathsf{seed} \leftarrow \{0,1\}^{256}$
2: $(\mathsf{seed}_{\mathbf{A}}, \mathsf{seed}_{\mathsf{sk}}, \mathsf{seed}_{\mathbf{e}}) \leftarrow \mathsf{XOF}(\mathsf{seed})$
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. 2: Description of SMAUG.PKE

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. □

4.2 Specification of SMAUG.KEM

We finally introduce the key encapsulation mechanism SMAUG.KEM in Figure 3. SMAUG.KEM is designed following the Fujisaki-Okamoto transform with implicit rejection using the non-perfectly correct public key encryption SMAUG.PKE. The construction of SMAUG.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.

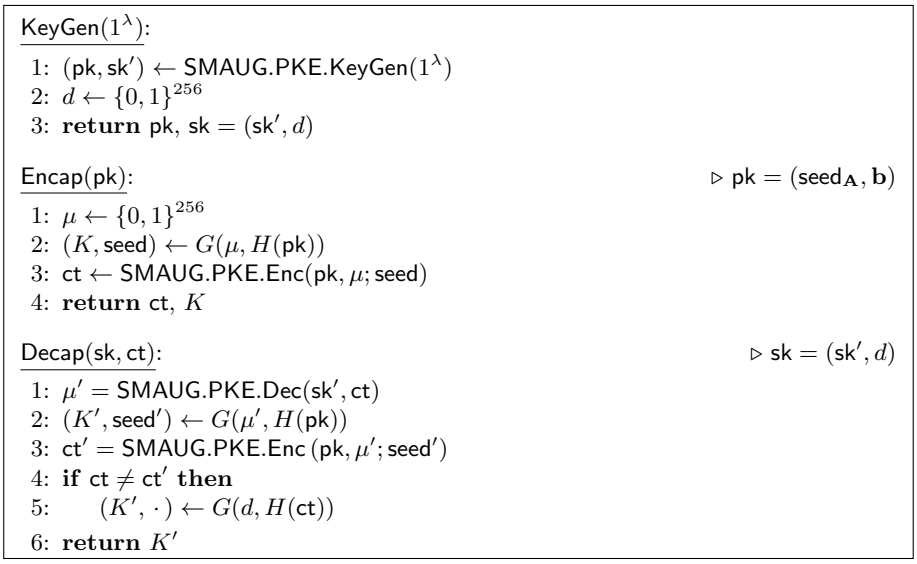

Fig. 3: Description of SMAUG.KEM

The Fujisaki-Okamoto transform used in Figure 3 defers from the $\mathsf{FO}^{\not\perp}_m$ transform in [43] in encapsulation and decapsulation. When generating the sharing key and randomness, SMAUG'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 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 $\mathsf{FO}^{\perp}_m$, recently treated in [45] with a competitive bound.

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

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

Theorem 3 (Completeness of SMAUG.KEM). *We borrow the notations and assumptions from Theorem 2 and Figure 3. Then* SMAUG.KEM *in Figure 3 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.PKE. The completeness of SMAUG.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.KEM has a tight reduction from the IND-CPA security of the underlying PKE, SMAUG.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 [14, 41, 43] in the QROM. Therefore, we prove the IND-CCA security of SMAUG.KEM based on the non-tight QROM reduction of [14] as explained in Section 2 by proving the IND-CPA security of SMAUG.PKE.

Theorem 4 (IND-CPA security of SMAUG.PKE). *Assuming pseudorandomness of the underlying sampling algorithms, the* IND-CPA *security of* SMAUG.PKE *can be tightly reduced to the decisional MLWE and MLWR problems. Specifically, for any* IND-CPA*-adversary* $\mathcal{A}$ *of* SMAUG.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,*

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

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 $\mathsf{Adv}_0 = \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{SMAUG.PKE}}(\mathcal{A})$ 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

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

□

The classical IND-CCA security of SMAUG.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.KEM in the quantum random oracle model.

5 Parameter selection and concrete security

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

5.1 Concrete security estimation

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

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 [23, 40, 56]. BKZ has been used in various lattice-based schemes [3, 16, 32, 35, 58]. 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. [9] 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 in core-SVP hardness.

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 use the lattice estimator for estimating such attacks.

We also focus on the attacks not considered in the lattice estimator, specifically those that target sparse secrets, such as Meet-LWE [51] attack. This attack is inspired by Odlyzko's Meet-in-the-Middle approach and involves using representations of ternary secrets in additive shares. We use a Python script to estimate the cost of the Meet-LWE attack, following the original description in [51].

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. 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. We also provide a detailed justification for using the narrow discrete Gaussian noise in a more conservative manner, in Appendix D.

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. [8] 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 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 set SMAUG parameters to make SMAUG at least as safe

as Saber. We first set our ring dimension to $n = 256$ and plaintext modulus to $t = 2$ to have a 256-bit message space (or sharing key space). Then we search for parameters that offer the smallest ciphertext size with enough security. Starting from parameters having a tiny ciphertext size, we increase the ciphertext size, h_s, h_r, and σ and search for the parameters having enough security. Once we have them, we compute the DFP. If it is enough low, 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 the ciphertext.

Table 3 shows the three parameter sets of SMAUG, 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.

Parameters sets	SMAUG-128	SMAUG-192	SMAUG-256
Security level	1	3	5
n	256	256	256
k	2	3	5
(q, p, p', t)	$(1024, 256, 32, 2)$	$(2048, 256, 256, 2)$	$(2048, 256, 64, 2)$
(h_s, h_r)	$(140, 132)$	$(198, 151)$	$(176, 160)$
σ	1.0625	1.453713	1.0625
Classical core-SVP	120.0	181.7	264.5
Quantum core-SVP	105.6	160.9	245.2
Beyond core-SVP	144.7	202.0	274.6
DFP	-119.6	-136.1	-167.2
Secret key	176	236	218
Public key	672	1088	1792
Ciphertext	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), DFP (in $\log_2$), and sizes (in bytes) of SMAUG.

The core-SVP hardness is estimated[5] via the lattice estimator [2] using the cost model "ADPS16" introduced in [5] and "MATZOV" [50]. 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

[5] There are some suspicions on the unsubstantiated dual-sieve attacks assuming the flawed heuristic [33]. However, we hereby estimate the security of SMAUG following the methods in Kyber, Saber, and Sable for a fair comparison.

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 have similar to or lower than that of Saber's.

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 [28, Figure 6 (a)], the quantum cost for finding a single failing ciphertext of SMAUG security level 3 is much higher than 2^{160}, as desired[6]. For security level 5, we refer to Figure 7(a) in [28], 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.

6 Implementation

In this section, we give the implementation performance for each parameter set. We compare the sizes and the reported performance with prior works such as Kyber, Saber, and Sable. The constant-time reference implementation of SMAUG, along with the supporting scripts, can be found in our website: `www.kpqc.cryptolab.co.kr/smaug`.

[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 larger than 2^{93}. The quantum cost is given as $1/\beta\sqrt{\alpha}$.

6.1 Performance

We instantiate the hash functions G, H and the extendable output function XOF with the following symmetric primitives: G is instantiated with SHAKE256, H is instantiated with SHA3-256, and XOF is instantiated with SHAKE128.

Table 4 presents the performance results of SMAUG. For a fair comparison, we also performed measurements on the same system with identical settings of the reference implementation of Kyber, Saber, and Sable[7].

	Cycles			**Cycles** (ratio)		
Schemes	KeyGen	Encap	Decap	KeyGen	Encap	Decap
Kyber512	131 560	162 472	189 30	1.7	2.1	2.03
LightSaber	93 752	122 176	133 764	1.21	1.58	1.44
LightSable	85 274	114 822	128 990	1.1	1.48	1.39
SMAUG-128	77 220	77 370	92 916	1	1	1
Kyber768	214 160	251 308	285 378	1.38	1.84	1.75
Saber	187 22	224 686	239 590	1.21	1.64	1.47
Sable	170 400	211 290	237 24	1.1	1.55	1.45
SMAUG-192	154 862	136 616	163 354	1	1	1
Kyber1024	332 470	371 854	415 498	1.25	1.38	1.36
FireSaber	289 278	347 900	382 326	1.08	1.29	1.25
FireSable	275 156	337 322	371 486	1.03	1.25	1.22
SMAUG-256	266 704	270 123	305 452	1	1	1

Table 4: Median cycle counts of 1000 executions for Kyber, Saber, Sable, and SMAUG (and their ratios). Cycle counts are obtained on one core of an Intel Core i7-10700k, with TurboBoost and hyperthreading disabled, using the C implementations without AVX optimizations.

6.2 Comparison to prior/related work

In this section, we compare the sizes, security, and performance of SMAUG to the recent lattice-based KEM schemes.

KEMs using unstructured lattices. Most of the recent KEM schemes are designed over the ring lattices or module lattices. The schemes based on the unstructured lattices have a tighter security reduction and achieve conservative security. However, the ciphertext size is much larger and has a lower performance than the schemes based on structured lattices. For e.g., Lizard [27] has a 10,896-byte ciphertext, and FrodoKEM [17] has a 9,752-byte ciphertext for the lowest level ($\approx$ 150 classical core-SVP hardness) whereas the security level-2 RLizard [27] has a 4,144-byte ciphertext.

[7] From `github.com/pq-crystals/kyber` (`518de24`), `github.com/KULeuven-COSIC/SABER` (`f7f39e4`), and .`github.com/josebmera/scabbard` (`4b2b5de`), respectively.

	Sizes (bytes)			**Sizes** (ratio)			**Security**	
Schemes	**sk**	**pk**	**ct**	**sk**	**pk**	**ct**	**Classic.**	**DFP**
Kyber512	1,632	800	768	9.4	1.2	1.1	118	-139
LightSaber	832	672	736	4.8	1	1.1	118	-120
LightSable	800	608	672	4.6	0.9	1	114	-139
SMAUG-128	176	672	672	1	1	1	120	-120
Kyber768	2,400	1,184	1,088	10.4	1.1	1.1	183	-164
Saber	1,248	992	1,088	5.4	0.9	1.1	189	-136
Sable	1,152	896	1,024	5	0.8	1	185	-143
SMAUG-192	236	1,088	1,024	1	1	1	181	-136
Kyber1024	3,168	1,568	1,568	15.2	0.9	1.1	256	-174
FireSaber	1,664	1,312	1,472	8	0.7	1	260	-165
FireSable	1,632	1,312	1,376	7.8	0.7	0.9	223	-208
SMAUG-256	218	1,792	1,472	1	1	1	264	-167

Table 5: Comparison of Kyber, Saber, Sable, and SMAUG. Sizes are given in bytes, and the ratios are given relative to the sizes of SMAUG. Security is provided in the classical core-SVP hardness with DFP (in logarithm base two).

Ring-based KEMs. When KEMs are designed over structured lattices, there is a higher risk of decryption failure. To ensure security against the decryption failure attacks [29, 44], a high enough DFP is necessary. As a result, the ring-based schemes take large parameters like RLizard or use Error Correction Codes (ECC) to reduce the failure rate. However, employing ECC makes it vulnerable to Side-Channel Attacks (SCA) as masking ECC against decryption failure probing is yet an open problem.

Moreover, it is difficult to achieve a message space of 256 bits while keeping a compact ciphertext size and low DFP for the underlying ring-based PKEs. Especially for low-security levels, balancing between DFP and the message space size is challenging. As a result, the KEMs such as Round5 [12] or Tiger [53] have small message spaces, which does not provide enough entropy for the shared key to be used in later quantum-secure protocols.

Module-based KEMs. When constructing KEMs over unstructured lattices or ring-variants, there are certain limitations to consider. As a result, module-based KEMs have become an attractive option for practical use due to their scalability in security, sizes, and DFP. Module-based KEMs Kyber and Saber, selected as a standard and a finalist by NIST, respectively, achieve ciphertext sizes of 768 and 736 bytes for security level 1, similar to SMAUG's.

Kyber and SMAUG. Tables 4 and 5 demonstrate that SMAUG outperforms Kyber in almost every aspect, except for the DFP and public key size in level 5. Compared to Kyber-512 [16], SMAUG-128 has a 16% and 12% smaller public key and ciphertext, respectively. Additionally, the secret key size of SMAUG is significantly smaller than that of Kyber. This makes it easy to manage secure zones in restricted IoT devices, as it is tiny and ready to use. Note that most of

the KEMs can store a secret key as a seed, having 32 bytes; however, this makes them more vulnerable to side-channel attacks.

Furthermore, SMAUG-128 achieves a 110% and 103% speed-up in encapsulation and decapsulation, respectively, while maintaining high security and low DFP. Similar results are presented for security levels 3 and 5, except that the public key size of Kyber is shorter than SMAUG's in level 5. We note that the speed-ups decrease in higher security parameters.

Saber, Sable, and SMAUG. When compared to Saber, one of NIST's round 3 finalists, SMAUG-128 has a 9% smaller ciphertext and the same public key size as LightSaber. The secret key is significantly smaller, with a 58% and 44% speed up in encapsulation and decapsulation, respectively. Compared to Sable, an efficient variant of Saber, SMAUG-128, has the same ciphertext size but a larger public key size. This can be seen as a trade-off between smaller public keys versus faster running time. SMAUG-128 achieves 48% and 39% faster encapsulation and decapsulation, a smaller secret key, and a bit higher security at the cost of a larger public key.

This trade-off is also observed in NIST's security levels 3 and 5. SMAUG offers a trade-off with Saber and Sable between the public key size versus the secret key size and running time. In NIST's level 3, the ciphertext size of SMAUG-192 is smaller than Saber and the same as Sable. The encapsulation and decapsulation speed outperforms by 44% to 64% with a much smaller, ready-to-use secret key.

In security level 5, FireSable has a classical core-SVP hardness of 223, much lower than 260. It achieves a smaller public key and ciphertext than SMAUG-256 but still with slower speeds. SMAUG-256 has the same ciphertext size as FireSaber, and a similar trade-off is observed.

6.3 Security against physical attacks

We justify the security of SMAUG against physical attacks based on the profiled Differential Power Analysis (DPA). Specifically, Simple Power Analysis (SPA) can profile the key generation and encapsulation processes since they only occur once or without a secret key. However, multi-trace attacks are possible for decapsulation due to"re-encryption." As Kyber and Saber share many design aspects with SMAUG, we can follow recent works on masking Kyber and Saber [10, 18] to add SCA countermeasures. Our new sampler, dGaussian, is expressed with bit operations, so adding SCA countermeasures like boolean masking is easy. While 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. The new multiplication method may be vulnerable to memory access patterns, but we can efficiently mask it using coefficient-wise shuffling.

Acknowledgments This work was submitted to the 'Korean Post-Quantum Cryptography Competition' (`www.kpqc.or.kr`). Part of this work was done while MinJune Yi was in CryptoLab Inc.

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. 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)
9. 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
10. 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
11. Bernstein, D.J., Chuengsatiansup, C., Lange, T., Van Vredendaal, C.: Ntru prime. IACR Cryptol. ePrint Arch. **2016**, 461 (2016)
12. Bhattacharya, S., Garcia-Morchon, O., Laarhoven, T., Rietman, R., Saarinen, M.J.O., Tolhuizen, L., Zhang, Z.: Round5: Kem and pke based on glwr. Cryptology ePrint Archive, Paper 2018/725 (2018), https://eprint.iacr.org/2018/725
13. 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
14. 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
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, C., Danba, O., Hoffstein, J., Hülsing, A., Rijneveld, J., Schanck, J.M., Saito, T., Schwabe, P., Whyte, W., Xagawa, K., Yamakawa, T., Zhang, Z.: Ntru: Algorithm specifications and supporting documentation (2020), nIST PQC Round 3 Submision
23. 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
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. Cheon, J.H., Park, S., Lee, J., Kim, D., Song, Y., Hong, S., Kim, D., Kim, J., Hong, S.M., Yun, A., et al.: Lizard public key encryption (2018), nIST PQC Round 1 Submision
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, Heidelberg (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, Heidelberg (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. 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
33. Ducas, L., Pulles, L.: Does the dual-sieve attack on learning with errors even work? Cryptology ePrint Archive, Paper 2023/302 (2023), https://eprint.iacr.org/2023/302
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, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-65277-7_20
35. 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)
36. 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
37. 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
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. 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
40. 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)
41. 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
42. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the fujisaki-okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) Theory of Cryptography. pp. 341–371. Springer International Publishing, Cham (2017)

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. Hövelmanns, K., Hülsing, A., Majenz, C.: Failing gracefully: Decryption failures and the fujisaki-okamoto transform. Cryptology ePrint Archive, Paper 2022/365 (2022), https://eprint.iacr.org/2022/365
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. Lindner, R., Peikert, C.: Better key sizes (and attacks) for LWE-based encryption. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 319–339. Springer, Heidelberg (Feb 2011). https://doi.org/10.1007/978-3-642-19074-2_21
50. MATZOV: Report on the Security of LWE: Improved Dual Lattice Attack (Apr 2022). https://doi.org/10.5281/zenodo.6493704
51. 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
52. Mera, J.M.B., Karmakar, A., Kundu, S., Verbauwhede, I.: Scabbard: a suite of efficient learning with rounding key-encapsulation mechanisms. IACR TCHES **2021**(4), 474–509 (2021). https://doi.org/10.46586/tches.v2021.i4.474-509, https://tches.iacr.org/index.php/TCHES/article/view/9073
53. 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
54. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) 37th ACM STOC. pp. 84–93. ACM Press (May 2005). https://doi.org/10.1145/1060590.1060603
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, Heidelberg (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. 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
58. Vercauteren, I.F., Sinha Roy, S., D'Anvers, J.P., Karmakar, A.: Saber: Mod-lwr based kem, nIST PQC Round 3 Submision

A Related work: LWE/LWR-based PKEs

We focus on the LWE/LWR-based IND-CPA secure PKEs, which can be turned into IND-CCA secure KEM by applying FO transforms. The original Regev's public key encryption [54], followed by most recent LWE- and LWR-based PKE constructions, bases its security on the LWE assumption. It generates an LWE sample as a public key and returns a ciphertext of a binary message by scaling and adding a public key multiplied by a random vector. In more detail, the public key is $(\mathbf{A}, \mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \bmod q) \in \mathbb{Z}_q^{k \times \ell} \times \mathbb{Z}_q^k$, where $\mathbf{s}$ is a secret key and $\mathbf{e}$ is an error term. The encryption of a binary message μ is a ciphertext $(\mathbf{r}^\top \cdot \mathbf{A}, \mathbf{r}^\top \cdot \mathbf{b} + \lfloor q/2 \cdot \mu \rceil)$, where $\mathbf{r}$ is a random vector, called an ephemeral secret. The ciphertext is then statistically uniform over the range due to the leftover hash lemma; however, it makes the size of the matrix $\mathbf{A}$ too massive for even a binary message to be used efficiently.

The approaches after that tried to gain more efficiency. Lindner and Peikert [49] introduce an encryption using an LWE sample as a ciphertext, i.e., $(\mathbf{r}^\top \cdot \mathbf{A} + \mathbf{e}_1, \mathbf{r}^\top \cdot \mathbf{b} + e_2 + \lfloor q/2 \cdot \mu \rceil) = \mathbf{r}^\top \cdot \mathsf{pk} + \mathbf{e} + (0, \lfloor q/2 \cdot \mu \rceil)$. Frodo [17], an instantiation of [49], introduces a narrower error for more efficiency. Lizard [26] uses LWR-based encryption, which can be viewed as an LWE sample with deterministic rounding error: $\left(\left\lfloor \frac{p}{q} \cdot \mathbf{r}^\top \cdot \mathbf{A} \right\rceil, \left\lfloor \frac{p}{q} \cdot \mathbf{r}^\top \cdot \mathbf{b} + \frac{p}{2} \cdot \mu \right\rceil \right) = \frac{p}{q} \cdot (\mathbf{r}^\top \cdot \mathbf{A} + \mathbf{e}, \mathbf{r}^\top \cdot \mathbf{b} + \frac{q}{2} \cdot \mu + e)$, where the coefficients of $\mathbf{e}$ and e are in $\mathbb{Z} \cap (-\frac{q}{2p}, \frac{q}{2p}]$. The scaling factor reduces the ciphertext size and the running time since the encryption process skips a complex error sampling procedure.

More efficient approaches are usually based on the assumptions over structured lattices such as NewHope [5] over RLWE, RLizard [48] over RLWR, and Kyber [16] over MLWE assumptions. Additionally, Kyber uses a Centered Binomial Distribution (CBD) as an MLWE error instead of a complex discrete Gaussian error, which makes the encryption much faster. Saber [30] expands the use of LWR also to the public key generation as $(\mathbf{A}, \mathbf{b} = \lfloor p/q \cdot \mathbf{A} \cdot \mathbf{s} \rceil \bmod p) \in \mathbb{Z}_q^{k \times \ell} \times \mathbb{Z}_p^k$ over module lattices. The extensions to rings and modules provide a wider message space, smaller sizes, and faster implementation based on the ring structure but with larger DFPs.

B Missing definitions

B.1 Lattice assumptions

We recall the lattice assumptions MLWE and MLWR in the structured Euclidean lattices, which security of SMAUG underlies.

Definition 2 (Decisional MLWE). *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-*$\mathsf{MLWE}_{n,q,k,\ell,\chi_s,\chi_e}$ *problem is*

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

Definition 3 (Decisional MLWR). *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-*$\mathsf{MLWR}_{n,p,q,k,\ell,\chi_r}$ *problem is*

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

B.2 Public key encryption and key encapsulation mechanism

Definition 4 (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 5 (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 (IND-CPA).

Definition 6 (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* IND-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 (IND-CPA) as in PKE and the INDistinguishability under (adaptively) Chosen Ciphertext Attacks (IND-CCA).

Definition 7 (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* IND-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* IND-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 of PKE and KEM in Definition 1.

C Missing algorihtms

Uniform random matrix sampler. The matrix sampling algorithm expandA given in Figure 4 is adapted from the pseudorandom generator gen in Saber [30]. This pseudorandom generator samples a public matrix $\mathbf{A}$ from a uniformly random distribution over $\mathcal{R}_q^{k \times k}$.

expandA(seed): ▷ seed $\in \{0,1\}^{256}$

1: buf $\leftarrow$ XOF(seed)
2: **for** i from 0 to $k-1$ **do**
3: $\quad \mathbf{A}[i] = \mathtt{bytes\_to\_Rq}(\mathsf{buf} + \mathsf{polybytes} \cdot i)$ ▷ Convert to ring elements
4: **return A**

Fig. 4: Uniform random matrix sampler, expandA.

Hamming weight sampler. The hamming weight sampler, HWT_h in Figure 5, is adapted from the SampleInBall algorithm in Dilithium [32], having a secret-independent running time. It samples a ternary polynomial vector having a hamming weight of h.

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

1: $\mathsf{count} = 0$
2: $\mathsf{buf} \leftarrow \mathsf{XOF}(\mathsf{seed})$
3: **for** i from $n-h$ to $n-1$ **do**
4: **repeat**
5: $\mathsf{degree} = \mathsf{buf[idx]} \wedge \mathsf{mask}$
6: **until** $\mathsf{degree} < i$
7: $\mathsf{res[i]} = \mathsf{res[degree]}$
8: $\mathsf{res[degree]} = ((\mathsf{buf[idx]} \gg 14) \wedge \mathtt{0x02}) - 1$
9: **return** $\mathtt{convToIdx}(\mathbf{s})$ ▷ Storing the indexes

Fig. 5: Hamming weight sampler, HWT_h.

Discrete Gaussian sampler. We describe dGaussian with $\sigma = 1.0625$ in Figure 6 and $\sigma = 1.453713$ in Figure 7.

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

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: $s_0 \mathrel{+}= (x_6x_7\overline{x_8}) + (\overline{x_5}\overline{x_6}x_8) + (\overline{x_4}\overline{x_6}x_8) + (\overline{x_7}x_8)$
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. 6: Discrete Gaussian sampler with $\sigma = 1.0625$, $\mathsf{dGaussian}_\sigma$.

D Narrow 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]. We give some theoretical results based on Rényi divergence to justify the security of SMAUG, considering the narrow discrete

$\underline{\mathsf{dGaussian}_\sigma(x)}$:

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

$s = s_2s_1s_0 = 000 \in \{0,1\}^3$

$s_0 = (x_0x_1x_2x_3x_5x_7x_8) + (x_1x_2x_3x_5\overline{x_6}x_7x_9) + (\overline{x_1x_2x_3}x_6x_7x_8)$

$s_0 \mathrel{+}= (\overline{x_1x_2x_3x_5x_8}x_9) + (\overline{x_0x_2x_3x_5x_8}x_9)$

$s_0 \mathrel{+}= (x_4x_5\overline{x_6}x_7x_9) + (x_3x_4x_8\overline{x_9}) + (\overline{x_5}x_6x_7x_8) + (\overline{x_4}x_6x_7x_8) + (\overline{x_4x_5x_8}x_9)$

$s_0 \mathrel{+}= (x_5x_8\overline{x_9}) + (x_6x_8\overline{x_9}) + (x_7x_8\overline{x_9}) + (\overline{x_7x_8}x_9) + (\overline{x_6x_8}x_9)$

$s_1 = (x_0x_1x_4\overline{x_5}x_6x_7x_9) + (x_2x_4\overline{x_5}x_6x_7x_9) + (x_3x_4\overline{x_5}x_6x_7x_9) + (x_5x_6x_7\overline{x_8}x_9)$

$s_1 \mathrel{+}= (\overline{x_1x_2x_3}x_8x_9) + (\overline{x_7}x_8x_9) + (\overline{x_6}x_8x_9) + (\overline{x_5}x_8x_9) + (\overline{x_4}x_8x_9)$

$s_2 = (x_1x_4x_5x_6x_7x_8x_9) + (x_2x_4x_5x_6x_7x_8x_9) + (x_3x_4x_5x_6x_7x_8x_9)$

$s = (-1)^{x_{10}} \cdot s$ ▷ $\cdot$ is the arithmetic multiplication

return s

Fig. 7: Discrete Gaussian sampler with $\sigma = 1.453713$, $\mathsf{dGaussian}_\sigma$.

Gaussian noise. We note that the narrow Gaussian noise is already taken into account when estimating the concrete security (given in Section 5) using the explained estimators. The analysis given here provides a more conservative security preventing some 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 or Saber.

In SMAUG, 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 $\mathsf{dGaussian}$ only nk times and that the public key is generated only once.

The bound for our typical parameter set for security level 1 (levels 3 and 5, resp.) increases from $2^{-120.0}$ ($2^{-181.7}$ and $2^{-264.5}$, resp.) to $2^{-118.2}$ ($2^{-176.9}$ and $2^{-260.2}$, resp.) with $\alpha = 200$ (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, this new bound provides a more conservative security preventing some possible future attacks that target the noise distribution.

By using one more bit for $\mathsf{dGaussian}$ algorithm, we can decrease the advantage to $2^{-119.6}$ ($2^{-181.2}$ and $2^{-263.6}$, resp.) with $\alpha = 500$. This modification will slightly decrease only the speed of key generation by less than 1.1x.