Compact Quaternion Algorithms for SQIsign

Won Kim[1], Changmin Lee[1,2⋆], and Hyunwoo Yoo[2]

[1] School of Cybersecurity, Korea University, Seoul, Korea
[2] Department of Cyber Defense, Korea University, Seoul, Korea

Abstract. SQIsign is an isogeny-based post-quantum signature scheme whose public keys and signatures are remarkably compact. However, since SQIsign relies on arithmetic in quaternion algebras over the field of rational numbers, no fixed-precision integer arithmetic for SQIsign had been established until recently, hindering constant-time implementation and deployment on memory-constrained devices. Recent work by Kim et al. instantiated an SQIsign implementation with fixed-precision integer arithmetic by deriving uniform worst-case bounds for the quaternion algorithms used in key generation and signing. Nevertheless, the resulting precision budget remains large, exceeding 13 times the public key size. Consequently, this forces implementations to reserve wide integer buffers throughout the computation. This increases the memory footprint and reduces the suitability of fixed-precision SQIsign for constrained platforms.

In this work, we present compact quaternion algorithms that substantially reduce the fixed-precision memory requirements of SQIsign. First, we modify and analyze quaternion algorithms for SQIsign, in which large intermediate integer values appear. Then, we derive the improved uniform worst-case size bound on integers during the key generation and signing procedures. As a result, we reduce the required precision budgets from 7026/10713/14150 bits to 1665/2521/3319 bits for the NIST-I/III/V security levels, respectively, corresponding to improvements of 76.30%, 76.47%, and 76.54%.

We also provide a fixed-precision implementation of SQIsign applying these improved precision budgets and accelerating previous inefficient implementation of fixed-precision integer arithmetic. Compared with the previous fixed-precision implementation, our implementation achieves performance improvements larger than 99% for both key generation and signing procedures at each security level.

Keywords: SQIsign · Quaternion algebras · Fixed-precision integer arithmetic

1 Introduction

The National Institute of Standards and Technology (NIST) initiated the Post-Quantum Cryptography (PQC) standardization process in response to the threat

⋆ Corresponding Author

that quantum computers pose to widely used public-key systems. NIST standardizes ML-DSA [1], Falcon [2], and SLH-DSA [3] to provide the foundation for most post-quantum signature deployments, but its ongoing additional-signatures process also makes clear that algorithmic diversity and compact public keys and signatures remain important design goals. Compactness of public keys and signatures matters in practice because they are carried in certificates, network-protocol handshakes, software-signing metadata, and secure-boot manifests, so their size directly affects bandwidth, storage, and latency. In particular, NIST has explicitly sought candidates beyond structured lattices, as well as schemes with short signatures and fast verification. In this setting, SQIsign [4] is especially compelling: as an isogeny-based Round-3 candidate, it offers far smaller public keys and signatures than the those of current post-quantum standards and leading candidates, which are often measured in kilobytes. More concretely, Table 1 compares sizes of public keys and signatures between SQIsign and NIST standard post-quantum digital signature schemes.

Scheme	PK (bytes)			Sig (bytes)		
	I	III	V	I	III	V
SQIsign	65	97	129	148	224	292
ML-DSA (Dilithium)	1312	1952	2592	2420	3309	4627
Falcon	897	–	1793	666	–	1280
SLH-DSA (SPHINCS+)	32	48	64	7856	16224	29792

Table 1: Public key (PK) and signature (Sig) sizes in bytes at NIST security levels I/III/V. Falcon defines parameter sets at levels I and V only.

Fixed-precision integer arithmetic [5] is one of the key requirements for practical cryptographic implementations: when all operands have statically bounded sizes, arithmetic routines can be implemented without dynamic memory allocation but with predictable control flow. It is essential for constant-time implementation [6] and for deployment on resource-constrained devices such as Cortex-M. This requirement was particularly challenging for SQIsign, whose key generation and signing algorithms involve arithmetic in a quaternion algebra over the field of rational numbers; intermediate size growth of integers representing quaternion elements is difficult to bound tightly. Consequently, existing SQIsign implementations have traditionally relied on multi-precision integer arithmetic, such as GMP, which complicates constant-time hardening and deployment on low-end devices. Recent work [7] has overcome this obstacle by proving explicit coefficient-growth bounds for SQIsign and using them to realize fixed-precision integer arithmetic, making SQIsign more portable and implementation-friendly [8].

However, the static precision required by the existing fixed-precision approach remains quite large [7, 8, 9], when computing the Hermite normal form (HNF) on integer matrices that SQIsign uses on its quaternion side. Conse-

quently, the global precision budget is driven not by the typical operand sizes arising in most routines, but by rare HNF intermediates whose coefficients can grow much larger than the input generators; this overprovisioning is undesirable in practice. More concretely, the precision required to cover the uniform worst-case bound of every quaternion algorithm for SQIsign, is as large as about 13.5 times the public key size and as about 6 times the signature size, for each NIST security level. More generally, for the security parameter λ and the base prime $p \approx 2^{2\lambda}$, the previous KeyGen bound is $2^{68}p^9$ and the previous Sign bound is $2^{16}p^{28}$ so quaternion operations need much more memory than operations in the finite field $\mathbb{F}_{p^2}$, whose intermediate value sizes are at most p^4. This forces the implementation to carry unnecessarily wide integers throughout arithmetic, slowing down basic arithmetic such as additions, multiplications, and reductions. It also increases the memory footprint and thereby makes the design less suitable for low-end or memory-constrained devices.

1.1 Our contributions

In this work, our contributions are as follows:

– **Improved fixed-precision**: By modifying some quaternion algorithms and analyzing existing algorithms tightly, we derive improved fixed-precision bounds compared with the previous work [7]. Our improvement is shown in Table 2.

Table 2: Comparison of worst-case fixed-precision bit sizes.

Security Level	Previous Work [7]	Ours (Worst-case)	Improvement
NIST-I	7,026	1,665	76.30%
NIST-III	10,713	2,521	76.47%
NIST-V	14,150	3,319	76.54%
General	$\log_2\left(2^{16}p^{28}\right)$	$\log_2\left(2^{35}47\pi^{-2}p^{6.5}\right)$	$\log_2\left(2^{-19}47^{-1}\pi^{2}p^{21.5}\right)$

Public key sizes are 520/776/1032 bits and signature sizes are 1184/1792/2336 bits for the NIST-I/III/V security level, respectively. Previous work achieves around 13.51/13.81/13.71 times the public key sizes and around 5.93/5.98/6.06 times the signature sizes, while ours achieves around 3.21/3.25/3.22 times the public key sizes and around 1.41/1.41/1.42 times the signature sizes, respectively.

– **Performance enhancement of the fixed-precision implementation**: From the reduced precision and optimization of big-integer arithmetic, we give a C implementation of SQIsign with our compact quaternion algorithms proposed in Section 3 and Section 4. Comparing with the previous fixed-precision implementation of SQIsign, we achieve the higher performance as shown in Table 3, achieving improvements of larger than 99% for both key generation and signing procedures, at each security level.

Table 3: Benchmark between two fixed-precision implementations of SQIsign with worst-case bounds. Table reports average wall-clock running time of KeyGen, Sign, and Verify in milliseconds (ms), for both implementations at each NIST security level.

Security level	Operation	Previous work [7]	Ours	Improvement
NIST-I	KeyGen	15,719.26ms	148.19ms	99.06%
	Sign	31,637.71ms	306.93ms	99.03%
	Verify	1.06ms	1.05ms	–
NIST-III	KeyGen	60,294.40ms	404.65ms	99.33%
	Sign	122,501.06ms	875.62ms	99.29%
	Verify	3.30ms	3.31ms	–
NIST-V	KeyGen	172,194.93ms	757.04ms	99.56%
	Sign	369,549.62ms	1,715.75ms	99.54%
	Verify	6.29ms	6.30ms	–

1.2 Technical overview

The goal of this work is to reduce the static precision required by fixed-precision integer arithmetic in SQIsign. Since the precision must cover every intermediate integer, it is determined by the largest size among all values inside quaternion algorithms used during key generation and signing procedures. We therefore identify the routines that dominate this intermediate growth, modify them, and otherwise give tighter size analyses for the existing procedures as follows:

- Replacing the subroutine HNF algorithm with a modified LLL algorithm for ideal operations to reduce the maximum size of intermediate values
- Modifying RandomIdealGivenNorm algorithm with smaller intermediate integer sizes while preserving the distribution of output ideals
- Giving a tighter bound on the reduced norm of output ideals of RandomEquivalentPrimeIdeal algorithm

The result is then propagated through the full SQIsign key generation and signing algorithms to obtain improved uniform worst-case bounds, which directly translate into a smaller fixed-precision memory requirement.

Replacing HNF with MLLL. The main bottleneck of precision growth in the previous fixed-precision analysis is the use of the HNF algorithm as a subroutine of quaternion ideal multiplication, which we call IdealMultiplication. To address this bottleneck, we replace the HNF algorithm with a variant algorithm [10] of the modified LLL (MLLL) algorithm [11], which takes a generating set of a lattice and returns an LLL-reduced basis of the same lattice. As a high-level visualization, this replacement is shown in Figure 1.

Since the role of HNF of ideal operations is to compute a basis of the result ideal, this replacement preserves correctness while reducing the intermediate

IdealMultiplication(I_1, I_2)

Input: Two ideals $I_1 = \langle \alpha_1, \ldots, \alpha_4 \rangle$, $I_2 = \langle \beta_1, \ldots \beta_4 \rangle$.

Output: An ideal $I_1 I_2 = \langle \gamma_1, \gamma_2, \gamma_3, \gamma_4 \rangle$.

1. $r_1 \leftarrow \mathrm{lcm}(r_{\alpha_1}, r_{\alpha_2}, r_{\alpha_3}, r_{\alpha_4}), \quad r_2 \leftarrow \mathrm{lcm}(r_{\beta_1}, r_{\beta_2}, r_{\beta_3}, r_{\beta_4})$
2. $\alpha_i \leftarrow r_1 \alpha_i, \quad \beta_j \leftarrow r_2 \beta_j \qquad$ for $1 \le i, j \le 4$
3. $m \leftarrow \mathrm{nrd}(r_1 I_1)^2 \, \mathrm{nrd}(r_2 I_2)^2$
4. ~~Compute $M \leftarrow \mathbf{HNF}(\alpha_i \beta_j)_{1 \le i,j \le 4}$ over $\mathbb{Z}_m$.~~
   Compute $M \leftarrow \mathbf{MLLL}\big((\alpha_i \beta_j)_{1 \le i,j \le 4}\big)$.
5. **return** $\dfrac{1}{r_1 r_2} M$

Fig. 1: Replacing HNF with MLLL in ideal multiplication.

integer values. More concretely, after clearing denominators, the previous bound of IdealMultiplication of I_1 and I_2 from HNF is

$$\mathrm{nrd}(I_1)^4 \mathrm{nrd}(I_2)^4$$

where $\mathrm{nrd}(I)$ is the reduced norm of I which is proportional to $\sqrt{\det(I)}$. This can be reduced to

$$\frac{4 \cdot 47^2 p^2}{\pi^4} \mathrm{nrd}(I_1) \mathrm{nrd}(I_2)$$

for a SQIsign parameter p, deriving asymptotically $1/4$ bound when p is treated as a constant.

Modifying RandomIdealGivenNorm for prime norm. Replacing HNF with MLLL would suggest a reduction of the dominant ideal-arithmetic bound by roughly a factor of four. However, this improvement is not fully realized because of the algorithm RandomIdealGivenNorm: for prime N, it samples two quaternion elements γ, β with large coefficients and computes the ideal generated by N and $\gamma\beta$, which needs a large coefficient to be represented.

The distribution of the output ideal depends only on the residue class of a product $\gamma\beta$ modulo $N\mathcal{O}_0$, where $\mathcal{O}_0$ is the left order (see Section 2.3) of the output ideal. We therefore modify the routine to sample β directly modulo N subject to $\mathrm{nrd}(\beta) \not\equiv 0 \pmod{N}$, compute $g \equiv \gamma\beta \pmod{N}$, replacing the direct product $\gamma\beta$. As a high-level visualization, this modification is shown in Figure 2.

The modified algorithm preserves the original uniform output distribution while requiring a smaller fixed-precision bound. More concretely, with an input N, the original algorithm requires the intermediate integers of size $O(N^4)$, while the modified one requires $O(N^2)$.

RandomIdealGivenNorm(N)

Input: A prime N.

Output: A uniformly random left $\mathcal{O}_0$-ideal J'.

1. **while** $\gcd\big(\mathrm{nrd}(\gamma), N^2\big) \neq N$ **do**
2. Sample integers $g_1, g_2, g_3 \xleftarrow{\$} [0, N-1]$ uniformly
3. $\gamma \leftarrow g_1 i + g_2 j + g_3 ij$
4. **if** $\big(-\mathrm{nrd}(\gamma) : N\big) = 1$ **then**
5. $\gamma \leftarrow \gamma + \Big(\sqrt{-\mathrm{nrd}(\gamma)} \bmod N\Big)$
6. Uniformly sample integers $a, b, c, d \xleftarrow{\$} [0, N-1]$ such that $a^2 + b^2 + p(c^2 + d^2) \not\equiv 0 \pmod N$
7. $\beta \leftarrow a + bi + cj + dk$
8. **return** $J' \leftarrow$ the left $\mathcal{O}_0$-ideal generated by ~~$\gamma\beta$~~ and N
$\gamma\beta \bmod N\mathcal{O}_0$

Fig. 2: Modification of RandomIdealGivenNorm algorithm.

Rigorous bound on RandomEquivalentPrimeIdeal. The algorithm RandomEquivalentPrimeIdeal first computes a reduced basis $(\alpha_1, \ldots, \alpha_4)$ of an input ideal I, samples

$$\beta = \sum_{i=1}^{4} c_i \alpha_i, \qquad |c_i| \leq \mathrm{QUAT_equiv_bound_coeff},$$

and returns $J = \chi_I(\beta)$ once $\mathrm{nrd}(J)$ is prime. In Section 4.2, we replace the conventional LLL reduction used in this procedure by ML2. The latter maintains exact integral lattice data together with fixed-precision floating-point approximations for the reduction decisions, thereby avoiding exact rational Gram–Schmidt arithmetic and providing a sharper bound on the intermediate integers. More importantly, whereas the previous analysis imposed an empirical bound without proving the size of the reduced basis, we establish that the resulting $(0.86, 0.55)$-LLL-reduced basis satisfies

$$\max_{1 \leq i \leq 4} \frac{\mathrm{nrd}(\alpha_i)}{\mathrm{nrd}(I)} < \frac{47}{\pi^2} p.$$

This yields a fully rigorous worst-case bound for both the basis-reduction step and the reduced norm of the returned ideal; the tighter $O(\sqrt{p})$ estimate is treated separately as a heuristic bound.

1.3 Related works

1.3.1 Efficient quaternion algorithms [9] Leroux recently proposed specialized algorithms for quaternion ideals and orders arising from the Deuring

correspondence that avoid generic linear algebra and tightly control the growth of intermediate integers. These techniques could therefore potentially reduce the precision required by fixed-precision SQIsign implementations. Their direct applicability to SQIsign, however, is limited by restrictive structural assumptions. In particular, the fast ideal-basis computation algorithm for replacing the basis-computation step in ideal multiplication assumes an ideal with an odd prime norm. The product ideals arising in SQIsign cannot have prime norm or left order $\mathcal{O}_0$, and hence this routine cannot replace general ideal multiplication in SQIsign. Similarly, the compact CRT-based intersection method considers coprime-norm left $\mathcal{O}_0$-ideals and obtains its simple size-controlled representation when all prime components are in the inert form. Since the ideals intersected during SQIsign signing are not guaranteed to satisfy these conditions, this method cannot be directly used to derive uniform fixed-precision bounds for general SQIsign ideal intersections.

1.3.2 Attack on the supersingular isogeny problem [12] Wesolowski recently presented, under a heuristic assumption on the smoothness of certain integers, a Las Vegas algorithm that solves the supersingular isogeny problem in expected time and memory $p^{1/3+o(1)}$, improving upon the previous complexity of $p^{1/2}(\log p)^{O(1)}$ [13]. Since this problem underlies the security estimates of SQIsign, the result necessitates a re-evaluation of its concrete parameter sets. Its practical impact, however, remains to be determined because of the superpolynomial overhead hidden in the $o(1)$ term and the $p^{1/3+o(1)}$ memory requirement. Several concrete bounds in this work, affecting the instantiated fixed-precision budgets, depend explicitly on the SQIsign parameters. We will therefore recompute these quantities once revised parameter sets become available.

Organization. The rest of this paper is organized as follows. Section 2 recalls the basic notation and background needed for this work. Section 3 replaces the HNF algorithm with a modified LLL algorithm in ideal operations and proves improved bounds on intermediate integer sizes. Section 4 additionally reduces the precision requirements by modifying RandomIdealGivenNorm and rigorously analyzing the size bound on output-norm of RandomEquivalentPrimeIdeal. Section 5 propagates these local improvements through key generation and signing procedures to obtain uniform fixed-precision bounds, and reports the resulting implementation performance.

2 Preliminaries

In this section, we recall background to understand the paper. We give notation used throughout the paper as shown in Table 4.

Table 4: Notation used throughout the paper.

Notation	Meaning
Basic parameters and finite fields	
λ	Security parameter.
p	Prime $p = c \cdot 2^f - 1$ where $p \approx 2^{2\lambda}$ and c is the smallest cofactor.
$\mathbb{F}_p, \mathbb{F}_{p^2}$	Finite fields with p and p^2 elements, respectively.
$K, \overline{K}$	A finite field K, and its algebraic closure.
$(a : N)$	Legendre symbol $\left(\frac{a}{N}\right)$ modulo the odd prime N.
Elliptic curves and isogenies	
E_0	Supersingular elliptic curve $E_0 : y^2 = x^3 + x$.
0_E	Identity point of the elliptic curve E.
$[n]$	Multiplication-by-n map on an elliptic curve.
Quaternion algebra, orders, and ideals	
$\mathcal{O}_0$	The maximal order $\mathcal{O}_0 = \mathbb{Z}\left\langle 1,\ i,\ \frac{i+j}{2},\ \frac{1+k}{2} \right\rangle$ in $B_{p,\infty}$
ι, π	Endomorphisms satisfying $\iota^2 = [-1]$, and $\pi^2 = [-p]$.
$\alpha(P)$	Action of the endomorphism corresponding to α on the point P.
r_α	$\min\{r \in \mathbb{Z}_{>0} : r\alpha \in \mathbb{Z} + \mathbb{Z}i + \mathbb{Z}j + \mathbb{Z}k\}$.
$\chi_I(\beta)$	An ideal $I\frac{\bar{\beta}}{\mathrm{nrd}(I)}$ equivalent to I.
Lattices and reduction	
$0_{m\times n}$	An $m \times n$ matrix with all entries 0.
Id_n	An $n \times n$ identity matrix.
$\|\cdot\|_\infty$	Maximum absolute coordinate norm.
$\lfloor x \rceil$	Nearest integer to x.
$\diamond(x)$	Nearest fixed-point value to x.
Sampling and algorithmic parameters	
D_{mix}	The smallest prime larger than $2^{4\lambda}$.
D_{rsp}	SQIsign parameter $2^{\lceil \log_2(\sqrt{p}) \rceil}$.
`QUAT_equiv_bound_coeff`	SQIsign parameter 64.
$\nu_2(x)$	Dyadic valuation of x.

2.1 Elliptic curves and isogenies

We recall the elliptic-curve notions needed to relate the quaternion algorithms used in SQIsign to their geometric meaning. For more details, see [14,15]. Throughout this subsection, let $K = \mathbb{F}_{p^2}$, where $p > 3$, and let $\overline{K}$ denote an algebraic closure of K. An elliptic curve over K is a smooth projective curve of genus one

with a distinguished point 0_E. When a short Weierstrass model is used, we write

$$E : y^2 = x^3 + ax + b, \qquad 4a^3 + 27b^2 \neq 0.$$

The points of E form an abelian group with identity 0_E. For an integer $n \geq 1$, the n-torsion subgroup is

$$E[n] := \{P \in E(\overline{K}) : [n]P = 0_E\}.$$

If $\gcd(n, p) = 1$, then $E[n] \cong (\mathbb{Z}/n\mathbb{Z})^2$.

An isogeny $\phi : E \to E'$ is a non-constant morphism of elliptic curves satisfying $\phi(0_E) = 0_{E'}$; equivalently, it is a group homomorphism with finite kernel. Its degree is denoted by $\deg(\phi)$, and degrees are multiplicative under composition. If ϕ is separable, then $\deg(\phi) = \#\ker(\phi)$. Conversely, for every finite subgroup $G \subset E(\overline{K})$, there is a quotient curve E/G and a separable isogeny $\phi_G : E \longrightarrow E/G$ with kernel G. Hence a separable isogeny can be specified by its kernel, and explicit equations for ϕ_G can be obtained using Vélu's formulas [16]. The dual isogeny $\widehat{\phi} : E' \to E$ satisfies

$$\widehat{\phi} \circ \phi = [\deg(\phi)] \quad \text{and} \quad \phi \circ \widehat{\phi} = [\deg(\phi)].$$

The endomorphism ring of E, denoted $\mathrm{End}(E)$, is the ring of isogenies from E to itself together with the zero map, where addition is defined pointwise and multiplication is composition. An elliptic curve in characteristic p is supersingular if

$$\mathrm{End}(E) \otimes_{\mathbb{Z}} \mathbb{Q} \cong B_{p,\infty},$$

where $B_{p,\infty}$ is the quaternion algebra over $\mathbb{Q}$ ramified exactly at p and at infinity.

2.2 Lattices

We recall only the lattice notions used in our setting. Let V be an n-dimensional real vector space equipped with a positive definite symmetric bilinear form $\langle \cdot, \cdot \rangle$, and write $\|x\| = \sqrt{\langle x, x \rangle}$. A full-rank lattice in V is a discrete subgroup

$$\Lambda = \mathbb{Z}b_1 + \cdots + \mathbb{Z}b_n$$

generated by an $\mathbb{R}$-basis $b_1, \ldots, b_n$ of V. Its Gram matrix is

$$G_\Lambda := (\langle b_i, b_j \rangle)_{1 \leq i,j \leq n},$$

and its covolume, or determinant, is

$$\det(\Lambda) := \sqrt{\det(G_\Lambda)}.$$

Equivalently, if $x = (x_1, \ldots, x_n)^T \in \mathbb{Z}^n$, then the squared length of the lattice vector $\sum_i x_i b_i$ is $x^T G_\Lambda x$.

The dual lattice of Λ, with respect to the same bilinear form, is

$$\Lambda^* = \{\, y \in V : \langle x, y \rangle \in \mathbb{Z} \text{ for all } x \in \Lambda \,\}.$$

If $\hat{b}_1, \ldots, \hat{b}_n$ denotes the basis dual to $b_1, \ldots, b_n$, i.e. $\langle b_i, \hat{b}_j \rangle = \delta_{ij}$, then $\hat{\Lambda} = \mathbb{Z}\hat{b}_1 + \cdots + \mathbb{Z}\hat{b}_n$ and $\det(\hat{\Lambda}) = \det(\Lambda)^{-1}$.

For $r > 0$, let $B_r = \{x \in V : \|x\| \leq r\}$. The i-th successive minimum of Λ is

$$\lambda_i(\Lambda) := \inf\{\, r > 0 : \dim_{\mathbb{R}} \operatorname{span}_{\mathbb{R}}(\Lambda \cap B_r) \geq i \,\}, \qquad 1 \leq i \leq n.$$

Thus $\lambda_1(\Lambda)$ is the length of a shortest nonzero vector of Λ, and

$$0 < \lambda_1(\Lambda) \leq \cdots \leq \lambda_n(\Lambda).$$

Basis reduction We also use LLL-reduced bases [17]. Let $B = (b_1, \ldots, b_n)$ be an ordered basis of a lattice Λ, and let $b_1^*, \ldots, b_n^*$ and $\mu_{i,j}$ be its Gram–Schmidt orthogonalization, so that

$$b_i = b_i^* + \sum_{j<i} \mu_{i,j} b_j^*.$$

The basis B is LLL-reduced, with the standard parameter $\delta = 3/4$, if it is size-reduced,

$$|\mu_{i,j}| \leq \frac{1}{2} \qquad (1 \leq j < i \leq n),$$

and satisfies the Lovász condition

$$\|b_i^* + \mu_{i,i-1} b_{i-1}^*\|^2 \geq \frac{3}{4} \|b_{i-1}^*\|^2 \qquad (2 \leq i \leq n).$$

Such a basis is short and nearly orthogonal in a quantitative sense:

Lemma 1 (LLL basis and successive minima [17, Proposition 1.6]). *Let Λ be a rank-n Euclidean lattice, and let $\lambda_i(\Lambda)$ denote its i-th successive minimum with respect to the Euclidean norm. Let*

$$B = (b_1, \ldots, b_n)$$

be an LLL-reduced basis of Λ. Then, for every $1 \leq i \leq n$,

$$2^{\frac{1-i}{2}} \lambda_i(\Lambda) \leq \|b_i\| \leq 2^{\frac{n-1}{2}} \lambda_i(\Lambda).$$

We define the analogous of the LLL-reudced basis as follows.

Definition 1 ((δ, η)-LLL-reduced basis). *Let Λ be a rank-n Euclidean lattice and let $B = (b_1, \ldots, b_n)$ be an ordered basis of Λ. Write its Gram–Schmidt orthogonalization $b_1^*, \ldots, b_n^*$ satisfying*

$$b_i = b_i^* + \sum_{j<i} \mu_{i,j} b_j^*.$$

For parameters

$$\delta \in (1/4, 1], \qquad \eta \in [1/2, \sqrt{\delta}),$$

the basis B is called (δ, η)-LLL-reduced if

$$|\mu_{i,j}| \leq \eta \qquad (1 \leq j < i \leq n)$$

and it satisfies the Lovász conditions

$$\|b_i^*\|^2 \geq \left(\delta - \mu_{i,i-1}^2\right)\|b_{i-1}^*\|^2 \qquad (2 \leq i \leq n).$$

For the specific (δ, η) parameters, we can bound norm of each basis element.

Lemma 2 (LLL basis vectors and successive minima). *Let Λ be a rank-n Euclidean lattice, and let $B = (b_1, \ldots, b_n)$ be a (δ, η)-LLL-reduced basis of Λ. Set*

$$\alpha_{\delta,\eta} := \frac{1}{\delta - \eta^2}.$$

Then, for every $1 \leq i \leq n$,

$$\|b_i\|^2 \leq \alpha_{\delta,\eta}^{n-1}\lambda_i(\Lambda)^2,$$

where $\lambda_i(\Lambda)$ denotes the i-th successive minimum of Λ.

Proof. Write

$$r_j := \|b_j^*\|^2, \qquad \alpha := \alpha_{\delta,\eta}.$$

The Lovász and size-reduction conditions imply

$$r_j \leq \alpha^{k-j} r_k \qquad (1 \leq j \leq k \leq n).$$

Moreover,

$$\|b_j\|^2 = r_j + \sum_{h<j} \mu_{j,h}^2 r_h \leq \left(1 + \eta^2 \sum_{t=1}^{j-1} \alpha^t\right) r_j \leq \alpha^{j-1} r_j.$$

Indeed, $\eta^2\alpha \leq \alpha - 1$ is equivalent to $\delta \leq 1$, and hence

$$\eta^2 \sum_{t=1}^{j-1} \alpha^t \leq (\alpha - 1) \sum_{t=0}^{j-2} \alpha^t = \alpha^{j-1} - 1.$$

Consequently, we have

$$\|b_j\|^2 \leq \alpha^{k-1} r_k \qquad (j \leq k).$$

Choose linearly independent vectors $x_1, \ldots, x_i \in \Lambda$ satisfying

$$\|x_\ell\| \leq \lambda_i(\Lambda) \qquad (1 \leq \ell \leq i),$$

and write

$$x_\ell = \sum_{k=1}^{n} z_{\ell,k} b_k, \qquad z_{\ell,k} \in \mathbb{Z}.$$

After reordering these vectors, the indices $m_\ell := \max\{k : z_{\ell,k} \neq 0\}$ satisfy $m_1 \leq \cdots \leq m_i$. Their linear independence implies $m_\ell \geq \ell$; otherwise $x_1, \ldots, x_\ell$ would lie in the span of fewer than ℓ basis vectors. Since z_{i,m_i} is a nonzero integer, triangularity of the Gram–Schmidt expansion gives

$$\|x_i\|^2 \geq z_{i,m_i}^2 r_{m_i} \geq r_{m_i}.$$

As $i \leq m_i \leq n$, it follows that

$$\|b_i\|^2 \leq \alpha^{m_i-1} r_{m_i} \leq \alpha^{n-1}\|x_i\|^2 \leq \alpha^{n-1}\lambda_i(\Lambda)^2.$$

□

2.3 Quaternion algebras and Deuring correspondence

We recall the quaternion algebras used to represent supersingular isogenies in SQIsign. For more details, see [18,19].

Quaternion algebras Let $B_{p,\infty}$ denote a quaternion algebra over $\mathbb{Q}$ whose $\mathbb{Q}$-basis is $\{1, i, j, k := ij\}$ with $i^2 = -1, j^2 = -p, ij = -ji$; every quaternion element can be represented by $\alpha = a + bi + cj + dk \in B_{p,\infty}$ for some $a, b, c, d \in \mathbb{Q}$. The canonical conjugate $\bar{\alpha}$ is defined as $a - bi - cj - dk$. The reduced trace and reduced norm are defined by

$$\mathrm{trd}(\alpha) := \alpha + \bar{\alpha},\ \mathrm{nrd}(\alpha) := \alpha\bar{\alpha} = \bar{\alpha}\alpha \in \mathbb{Q}.$$

A *fractional ideal* in $B_{p,\infty}$ is a $\mathbb{Z}$-submodule of rank 4. An *order* $\mathcal{O}$ is a fractional ideal that is also a subring of $B_{p,\infty}$ containing 1. An order $\mathcal{O}$ is *maximal* if it is not properly contained in any other order. A fractional ideal I is called a left $\mathcal{O}$-ideal if $\alpha I \subseteq I$ for all $\alpha \in \mathcal{O}$. It is called a right $\mathcal{O}$-ideal if $I\alpha \subseteq I$ for all $\alpha \in \mathcal{O}$. For a fractional ideal I, its left and right orders are defined by

$$O_L(I) := \{\alpha \in B_{p,\infty} : \alpha I \subseteq I\}, \qquad O_R(I) := \{\alpha \in B_{p,\infty} : I\alpha \subseteq I\}.$$

Its conjugate is defined by $\bar{I} := \{\bar{\alpha} : \alpha \in I\}$ and its reduced norm is defined by $\mathrm{nrd}(I) := \gcd\{\mathrm{nrd}(\alpha) : \alpha \in I\}$. Also, its inverse ideal is defined by $I^{-1} := \{x \in B_{p,\infty} : Ix \subseteq O_L(I)\}$, which satisfies $I^{-1} = \frac{\bar{I}}{\mathrm{nrd}(I)}$ if $O_L(I)$ is maximal (or, equivalently, $O_R(I)$ is).

We say that I is integral if it is contained in its left order (or, equivalently, in its right order). Moreover, I is said to be cyclic if for all prime ℓ we have that $I \not\subset \ell O_L(I)$. Two left $\mathcal{O}$-ideals I and J are said to be *equivalent*, written $I \sim J$, if there exists $\beta \in B_{p,\infty}^\times$ such that $J = I\beta$. In this case,

$$O_L(J) = O_L(I) \qquad \text{and} \qquad O_R(J) = \beta^{-1} O_R(I)\beta.$$

The corresponding equivalence classes form the *left ideal class set* of $\mathcal{O}$, which is the object that appears on the quaternionic side of the Deuring correspondence.

Deuring correspondence. For a supersingular elliptic curve E defined over $\mathbb{F}_{p^2}$, the endomorphism ring $\mathrm{End}(E)$ is isomorphic to a maximal order in $B_{p,\infty}$ [20]. Conversely, every maximal order in $B_{p,\infty}$ occurs as $\mathrm{End}(E)$ for some supersingular elliptic curve E. Thus, on objects, the Deuring correspondence identifies isomorphism classes of supersingular elliptic curves with isomorphism classes of maximal orders in $B_{p,\infty}$.

For a separable isogeny $\varphi : E \to E'$, we define the associated connecting ideal by

$$I_\varphi := \{\alpha \circ \varphi : \alpha \in \mathrm{Hom}(E', E)\}.$$

Choose $\sqrt{-1} \in \mathbb{F}_{p^2}$ such that $\sqrt{-1}^2 = -1$, and define endomorphisms in $\mathrm{End}(E_0)$

$$\iota(x, y) := (-x, \sqrt{-1}y), \qquad \pi(x, y) := (x^p, y^p).$$

Then $\iota^2 = [-1]$, $\pi^2 = [-p]$, and $\iota\pi = -\pi\iota$. The map

$$\rho_0 : \ B_{p,\infty} \longrightarrow \mathrm{End}(E_0) \otimes_{\mathbb{Z}} \mathbb{Q}, \qquad i \mapsto \iota, \quad j \mapsto \pi, \quad k = ij \mapsto \iota\pi,$$

restricts to an isomorphism [21]

$$\rho_0|_{\mathcal{O}_0} : \ \mathcal{O}_0 := \mathbb{Z}\Big\langle 1, i, \frac{i+j}{2}, \frac{1+k}{2} \Big\rangle \xrightarrow{\sim} \mathrm{End}(E_0).$$

We give a table which summarizes the Deuring correspondence, the correspondence between objects of elliptic curves and objects of quaternion algebras.

Elliptic-curve world	**Quaternion-algebra world**
Supersingular elliptic curve E	$\mathcal{O} = \mathrm{End}(E) \subset B_{p,\infty}$
$\varphi : E \to E'$	I_φ, a $\mathcal{O}$,$\mathcal{O}' = \mathrm{End}(E')$-connecting ideal
$\deg(\varphi)$	$\mathrm{nrd}(I_\varphi)$
$\varphi, \psi : E \to E'$	$I_\varphi \sim I_\psi$
$\widehat{\varphi} : E' \to E$	$I_{\widehat{\varphi}} = \overline{I_\varphi}$
$\psi \circ \varphi$	$I_\varphi \cdot I_\psi$

Quaternion lattices. For fixed-precision analysis, these ideals are viewed as rank-four Euclidean lattices. The reduced norm is a positive-definite quadratic form on $B_{p,\infty} \otimes_{\mathbb{Q}} \mathbb{R}$. Its associated symmetric bilinear form is

$$\langle \alpha, \beta \rangle := \frac{1}{2} \left(\mathrm{nrd}(\alpha + \beta) - \mathrm{nrd}(\alpha) - \mathrm{nrd}(\beta) \right),$$

and we define

$$\|\alpha\| := \sqrt{\mathrm{nrd}(\alpha)}.$$

Thus,

$$\|\alpha\|^2 = \langle \alpha, \alpha \rangle = \mathrm{nrd}(\alpha).$$

Hence a quaternion lattice $\Lambda = \mathbb{Z}\alpha_1 + \mathbb{Z}\alpha_2 + \mathbb{Z}\alpha_3 + \mathbb{Z}\alpha_4$ with $\mathbb{Q}$-linearly independent $\alpha_i \in B_{p,\infty}$ is a rank-four Euclidean lattice with Gram matrix

$$G_\Lambda = \left(\frac{\mathrm{nrd}(\alpha_i + \alpha_j) - \mathrm{nrd}(\alpha_i) - \mathrm{nrd}(\alpha_j)}{2} \right)_{1 \le i,j \le 4}.$$

Thus, for $x = (x_1, \ldots, x_4)^T \in \mathbb{Z}^4$,

$$\left\| \sum_{i=1}^{4} x_i \alpha_i \right\|^2 = \mathrm{nrd}\left(\sum_{i=1}^{4} x_i \alpha_i \right) = x^T G_\Lambda x.$$

Consequently, enumerating or sampling quaternion elements of bounded reduced norm is equivalent to enumerating or sampling vectors in this positive definite lattice with respect to the quadratic form $x^T G_\Lambda x$.

Throughout this paper, a quaternion ideal is represented by a coordinate matrix with respect to the fixed ambient basis $(1, i, j, k)$ of $B_{p,\infty}$. More precisely, for a basis $I = \mathbb{Z}\alpha_1 + \mathbb{Z}\alpha_2 + \mathbb{Z}\alpha_3 + \mathbb{Z}\alpha_4$, we write each basis element as

$$\alpha_\ell = M_{1,\ell} + M_{2,\ell} i + M_{3,\ell} j + M_{4,\ell} k \qquad (1 \le \ell \le 4),$$

and define its coordinate matrix by

$$M_I := (M_{m,\ell})_{1 \le m,\ell \le 4}.$$

Thus the ℓ-th column of M_I is the coordinate vector of α_ℓ, and

$$(\alpha_1, \alpha_2, \alpha_3, \alpha_4) = (1, i, j, k) M_I.$$

When the coefficients are rational, we write $M_I = A_I / r_I$, where $A_I \in \mathbb{Z}^{4\times 4}$ and $r_I > 0$, so that

$$(\alpha_1, \alpha_2, \alpha_3, \alpha_4) = (1, i, j, k) \frac{A_I}{r_I}.$$

We also define the reduced-norm dual lattice

$$I^\# := \{\beta \in B_{p,\infty} : \langle \alpha, \beta \rangle \in \mathbb{Z} \text{ for all } \alpha \in I\},$$

which is the dual lattice of I with respect to the Euclidean inner product associated with the reduced norm.

For any maximal order $\mathcal{O}$ in $B_{p,\infty}$ and any left $\mathcal{O}$-ideal I, we define

$$q_I : I \to \mathbb{Z},\ \alpha \mapsto \frac{\mathrm{nrd}(\alpha)}{\mathrm{nrd}(I)}.$$

Now we state some lemmas used repeatedly in the later fixed-precision analysis.

Lemma 3. *Let I be a left ideal of a maximal order $\mathcal{O} \subset B_{p,\infty}$, and let (b_1, b_2, b_3, b_4) be a $(0.86, 0.55)$-LLL-reduced basis of I with respect to the Euclidean norm $\|x\|^2 = \mathrm{nrd}(x)$. Then, for every $1 \leq i \leq 4$ we have,*

$$\mathrm{nrd}(b_i) < \frac{47p}{\pi^2}\mathrm{nrd}(I).$$

Proof. Let $\lambda_i(I)$ denote the i-th successive minimum of I, and set

$$\alpha := \frac{1}{\delta - \eta^2}.$$

By Lemma 2, we have

$$\|b_i\|^2 \leq \alpha^{n-1}\lambda_i(I)^2.$$

Indeed, if $r_j = \|b_j^*\|^2$, the Lovász and size-reduction conditions imply

$$\|b_i\|^2 \leq \alpha^{i-1} r_i, \qquad r_i \leq \alpha^{n-i}\lambda_i(I)^2.$$

Here

$$n = 4, \qquad \delta = \frac{43}{50}, \qquad \eta = \frac{11}{20}, \qquad \alpha = \left(\frac{43}{50} - \left(\frac{11}{20}\right)^2\right)^{-1} = \frac{400}{223}.$$

Moreover, by [22, Lemma 48], the last successive minimum of a quaternion ideal satisfies

$$\lambda_4(I)^2 \leq \frac{8p}{\pi^2}\mathrm{nrd}(I).$$

Therefore, for every i,

$$\mathrm{nrd}(b_i) = \|b_i\|^2 \leq \left(\frac{400}{223}\right)^3 \lambda_i(I)^2 \leq 8\left(\frac{400}{223}\right)^3 \frac{p}{\pi^2}\mathrm{nrd}(I).$$

Finally,

$$8\left(\frac{400}{223}\right)^3 \approx 46.1696 < 47,$$

which gives the claimed statement. □

Lemma 4 ([7, Lemma 5]). *Let $\mathcal{O}$ be a maximal order in $B_{p,\infty}$ with a $\mathcal{O}_0, \mathcal{O}$-connecting ideal I s.t. $\mathrm{nrd}(I) = N$. Then, for any $\alpha \in \mathcal{O}$, $r_\alpha \leq 2N$.*

Lemma 5 ([7, Lemma 9]). *Let $\mathcal{O}$ be a maximal order in $B_{p,\infty}$ and I_1, I_2 be two left $\mathcal{O}$-ideals. Then,*

$$\mathrm{nrd}(I_1 \cap I_2) \leq \mathrm{nrd}(I_1)\mathrm{nrd}(I_2).$$

Lemma 6 ([7, Lemma 15]). *Let I be a left/right-ideal of a maximal order $\mathcal{O}$ in $B_{p,\infty}$. Then,*

$$\det(I) = \det(\mathcal{O})\mathrm{nrd}(I)^2.$$

Lemma 7 ([7, Lemma 16]). $\det(\mathcal{O}_0) = \frac{p}{4}$

Lemma 8. *Let*

$$x = \sum_{r=1}^{4} u_r\alpha_r, \qquad y = \sum_{s=1}^{4} v_s\beta_s,$$

where

$$u_r, v_s \in \mathbb{Q}, \qquad \alpha_r, \beta_s \in B_{p,\infty},$$

and define

$$t_{r,s} := \mathrm{trd}(\alpha_r\overline{\beta_s}).$$

Then

$$\begin{aligned}\mathrm{nrd}(x+y) &= \mathrm{nrd}(x) + \mathrm{nrd}(y) + \mathrm{trd}(x\overline{y}) \\ &= \mathrm{nrd}(x) + \mathrm{nrd}(y) + \sum_{r,s=1}^{4} u_r v_s t_{r,s}.\end{aligned}$$

Proof. By the definitions of the reduced norm and reduced trace,

$$\mathrm{nrd}(z) = z\overline{z}, \qquad \mathrm{trd}(z) = z + \overline{z}$$

for every $z \in B$. Hence

$$\mathrm{nrd}(x+y) = (x+y)\overline{(x+y)} = (x+y)(\overline{x}+\overline{y}) = \mathrm{nrd}(x) + \mathrm{nrd}(y) + x\overline{y} + y\overline{x}.$$

Canonical conjugation is an involutive anti-automorphism, so

$$\overline{x\overline{y}} = \overline{\overline{y}}\,\overline{x} = y\overline{x}.$$

Consequently,

$$x\overline{y} + y\overline{x} = x\overline{y} + \overline{x\overline{y}} = \mathrm{trd}(x\overline{y}),$$

which proves

$$\mathrm{nrd}(x+y) = \mathrm{nrd}(x) + \mathrm{nrd}(y) + \mathrm{trd}(x\overline{y}).$$

Since canonical conjugation fixes $\mathbb{Q}$ and is $\mathbb{Q}$-linear,

$$\overline{y} = \sum_{s=1}^{4} v_s\overline{\beta_s}.$$

Using the bilinearity of multiplication and the $\mathbb{Q}$-linearity of the reduced trace, we obtain

$$\begin{aligned}\mathrm{trd}(x\overline{y}) &= \mathrm{trd}\left(\left(\sum_{r=1}^{4} u_r\alpha_r\right)\left(\sum_{s=1}^{4} v_s\overline{\beta_s}\right)\right) = \mathrm{trd}\left(\sum_{r,s=1}^{4} u_r v_s \alpha_r\overline{\beta_s}\right)\\ &= \sum_{r,s=1}^{4} u_r v_s \mathrm{trd}(\alpha_r\overline{\beta_s}) = \sum_{r,s=1}^{4} u_r v_s t_{r,s}.\end{aligned}$$

Substituting this identity into the preceding polarization formula proves the claim. □

2.4 SQIsign

SQIsign [4] is an isogeny-based digital signature scheme over supersingular elliptic curves. Conceptually, it is obtained by applying the Fiat–Shamir transform to a proof of knowledge of hidden endomorphism information associated with a public supersingular curve. For more details, see [23].

SQIsign is a digital signature scheme specified by four algorithms

$$\mathsf{SQIsign} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}),$$

where Setup generates public parameters, KeyGen outputs a public/secret key pair, Sign produces a signature on a message, and Verify checks its validity. At a high-level, each algorithm can be explained as follows:

Setup(1^λ)

1. Choose public parameters for the security level λ, including a prime p.
2. Fix the prescribed degrees and encodings for key, commitment, challenge, and response isogenies, together with torsion-basis conventions.
3. Fix a hash function H, modeled as a random oracle in the Fiat–Shamir view, and output the public parameters pp.

KeyGen(pp)

1. Sample a secret isogeny
$$\varphi_{\mathsf{sk}} : E_0 \longrightarrow E_{\mathsf{pk}},$$
internally represented through the quaternion-ideal machinery.
2. Set the public key to be the codomain curve, together with auxiliary public decoding data:
$$\mathsf{pk} = (E_{\mathsf{pk}}, \mathsf{hint}_{\mathsf{pk}}).$$

3. Store as secret key a compact representation of φ_{sk}, equivalently enough information to recover the hidden endomorphism structure of E_{pk}, together with auxiliary basis data used for signing.

$\underline{\mathsf{Sign}(\mathsf{sk}, m)}$.

1. *Commitment.* Sample a fresh random commitment isogeny
$$\varphi_{\mathsf{com}} : E_0 \longrightarrow E_{\mathsf{com}}.$$
2. *Challenge.* Compute the Fiat–Shamir challenge
$$c := H(\mathsf{pk} \,\|\, j(E_{\mathsf{com}}) \,\|\, m),$$
and interpret c as defining a challenge isogeny
$$\varphi_{\mathsf{chl}} : E_{\mathsf{pk}} \longrightarrow E_{\mathsf{chl}}.$$
3. *Response.* Using the secret isogeny φ_{sk} and the commitment data, compute a compact descriptor R of a response isogeny
$$\varphi_{\mathsf{rsp}} : E_{\mathsf{com}} \longrightarrow E_{\mathsf{chl}},$$
subject to SQIsign's no-backtracking condition.
4. Output
$$\sigma = (c, R).$$
In the concrete scheme, R is encoded using auxiliary curves, small integers, torsion-basis hints, and change-of-basis matrices rather than by explicitly transmitting the full isogeny.

$\underline{\mathsf{Verify}(\mathsf{pk}, m, \sigma)}$

1. Parse $\sigma = (c, R)$. Reject if the encoded curves, integers, hints, or matrices are malformed, or if the relevant curves are not supersingular.
2. From pk and c, reconstruct the challenge isogeny
$$\varphi_{\mathsf{chl}} : E_{\mathsf{pk}} \longrightarrow E_{\mathsf{chl}}.$$
3. From the response descriptor R, evaluate the claimed response structure and recover the corresponding commitment curve E_{com} up to isomorphism.
4. Check that the response data is consistent with an isogeny $E_{\mathsf{com}} \to E_{\mathsf{chl}}$ and satisfies the required no-backtracking and kernel-validity conditions.

5. Recompute
$$c' := H(\mathsf{pk} \,\|\, j(E_{\mathsf{com}}) \,\|\, m),$$
and accept iff $c' = c$ and all previous consistency checks pass.

3 Ideal operations without HNF

In SQIsign, a left $\mathcal{O}_0$-ideal is represented by a rank-4 $\mathbb{Z}$-lattice in a fixed integral basis of $\mathcal{O}_0$. The standard implementation performs ideal arithmetic by first producing a generating set for the resulting lattice and then normalizing it by computing its HNF. While HNF is exact and canonical, computing it is not well suited to fixed-precision arithmetic: elimination steps of the HNF algorithm may create intermediate coefficients that are substantially larger than those of the input generators.

In this section, we remove this source of coefficient growth. We replace the HNF algorithm by the modified LLL algorithm [10, 11], which also takes a generating set of the lattice as input and then outputs its basis. This does not alter the represented ideals; rather, it produces a reduced basis of the same lattice while keeping the size of intermediate integers within significantly smaller sizes than the HNF algorithm. We then combine these bounds with the framework of [7] to derive sufficient bit sizes for the fixed-precision integer arithmetic used in SQIsign.

3.1 Modified LLL algorithm

The modified LLL algorithm [11] is particularly convenient in our setting because it takes as input an arbitrary generating family, possibly linearly dependent, and directly outputs an LLL-reduced basis of the lattice that it spans. This matches the output of ideal operations in SQIsign, which naturally produce redundant generating sets. Hence modified LLL can be used as a direct replacement for the HNF algorithm for ideal operations.

To reduce the size bound on the algorithm, we use the modified LLL algorithm with floating-point arithmetic [10], which we call ML2 or modified LLL algorithm, for the sake of simplicity. We briefly recall the algorithm for completeness. The ML2 algorithm is given in Algorithm 1. We also call LLL as the special case of MLLL algorithm, when $d = 4$ in Algorithm 1.

First, we claim the correctness of algorithm 1.

Lemma 9 (Correctness of ML2 in rank four). *Let L be a positive-definite lattice of rank 4. Consider*
$$\bar{\delta} = \diamond\left(\frac{\diamond(3/4) + 1}{2}\right) = \frac{7}{8}, \qquad \eta = 0.55.$$
Then, the nonzero output vectors form a $(0.86, 0.55)$-LLL-reduced basis of L.

Algorithm 1 $\mathsf{ML2}(b_1, \dots, b_d)$ [10, Figure 9]

Input: Nonzero vectors $(b_1, \dots, b_d) \subset \mathbb{Z}^4$ generating a rank-4 lattice, and fp-precision $\ell + 1 = 53$.
Output: A (δ, η)-LLL-reduced basis $(b'_1, \dots, b'_4)$.

$\bar{\delta} \leftarrow \diamond(\diamond(3/4) + 1)/2, \quad \bar{\eta} \leftarrow \diamond\left(\frac{\diamond(0.55)+1/2}{2}\right), \quad \bar{r}_{1,1} \leftarrow \diamond(\langle b_1, b_1 \rangle), \quad \kappa \leftarrow 2, \quad \zeta \leftarrow 0.$
while $\kappa \leq d$ **do**
  $(b_\kappa, G, \bar{r}_{\kappa,*}, \bar{\mu}_{\kappa,*}, \bar{s}_*^{(\kappa)}) \leftarrow \mathsf{LazySizeReduce}(\kappa, \zeta, (b_1, \dots, b_d), G_{(b_1,\dots,b_d)}, \bar{r}, \bar{\mu})$. ▷ Algorithm 8 in Appendix A.1
  $\kappa' \leftarrow \kappa$.
  while $\kappa \geq \zeta + 2$ **and** $\diamond(\bar{\delta} \cdot \bar{r}_{\kappa-1,\kappa-1}) \geq \bar{s}_{\kappa-1}^{(\kappa')}$ **do**
    $\kappa \leftarrow \kappa - 1$.
  end while
  for $i = \zeta + 1$ **to** $\kappa - 1$ **do**
    $\bar{\mu}_{\kappa,i} \leftarrow \bar{\mu}_{\kappa',i}, \quad \bar{r}_{\kappa,i} \leftarrow \bar{r}_{\kappa',i}$.
  end for
  $\bar{r}_{\kappa,\kappa} \leftarrow \bar{s}_\kappa^{(\kappa')}$.
  Insert $b_{\kappa'}$ right before b_κ and update G accordingly.
  if $b_\kappa = 0$ **then**
    $\zeta \leftarrow \zeta + 1$.
  end if
  $\kappa \leftarrow \max(\zeta + 2, \kappa + 1)$.
end while
return $(b_{\zeta+1}, \dots, b_d)$.

Proof. Set

$$\rho := \frac{(1 + 0.55)^2 + 1/2}{3/4 - 0.55^2} = \frac{1161}{179}.$$

For every final Lovász test, let r and s denote the corresponding exact Gram–Schmidt quantities and let $\bar{r}, \bar{s}$ be their floating-point approximations. The error bounds give

$$|\bar{r} - r| \leq Er, \qquad |\bar{s} - s| \leq Ar + Bs,$$

where, using the worst-case index $j \leq 4$,

$$E = 4\rho^4 2^{-50} < 6.3{\cdot}10^{-12}, \qquad A = 4\rho^4 2^{-43} < 8.1{\cdot}10^{-10}, \qquad B = 4{\cdot}2^{-52} < 8.9{\cdot}10^{-16}.$$

At termination,

$$\bar{s} > \diamond\left(\frac{7}{8}\bar{r}\right) \geq (1 - 2^{-53})\frac{7}{8}\bar{r}.$$

Consequently,

$$\frac{s}{r} > \frac{(1 - 2^{-53})\frac{7}{8}(1 - E) - A}{1 + B} > 0.8749999991 > 0.86.$$

It remains to prove the size-reduction condition. Consider a call to LazySizeReduce on a nonzero current vector b_κ, and let $\mu_{\kappa,j}$ and $\bar{\mu}_{\kappa,j}$ denote, respectively,

its exact and computed Gram–Schmidt coefficients with respect to the active nonzero prefix

$$(b_{\zeta+1}, \ldots, b_{\kappa-1}).$$

Whenever the stopping test fails, the subroutine computes

$$X_i = \lfloor \bar{\mu}_{\kappa,i} \rceil \qquad (\zeta + 1 \le i < \kappa)$$

in decreasing order of i, and performs the exact update

$$b'_\kappa = b_\kappa - \sum_{i=\zeta+1}^{\kappa-1} X_i b_i.$$

Since the preceding vectors are unchanged, so are their Gram–Schmidt vectors, and the exact coefficients of b'_κ satisfy

$$\mu'_{\kappa,j} = \mu_{\kappa,j} - X_j - \sum_{i=j+1}^{\kappa-1} X_i \mu_{i,j}.$$

Thus, this update is the usual nearest-plane size-reduction step. The lazy variant recomputes the coefficients from the exact updated Gram matrix and repeats this step because the integers X_i are selected from floating-point approximations. More precisely, the subroutine returns only when

$$\max_{\zeta+1 \le j < \kappa} |\bar{\mu}'_{\kappa,j}| \le \bar{\eta}, \qquad \bar{\eta} := \diamond\left(\frac{\diamond(0.55) + 1/2}{2}\right).$$

By the lazy size-reduction error analysis of [10, Theorems 5 and 10], specialized to active rank 4, precision $\ell = 52$, and

$$\rho = \frac{(1 + 0.55)^2 + 1/2}{3/4 - 0.55^2} = \frac{1161}{179},$$

the exact and computed coefficients at the terminating iteration satisfy

$$\left|\mu'_{\kappa,j} - \bar{\mu}'_{\kappa,j}\right| \le 4\rho^3 2^{-48}.$$

Moreover, correct rounding gives

$$\bar{\eta} \le \frac{0.55 + 1/2}{2} + 2^{-52}.$$

Consequently,

$$\begin{aligned} |\mu'_{\kappa,j}| &\le |\bar{\mu}'_{\kappa,j}| + \left|\mu'_{\kappa,j} - \bar{\mu}'_{\kappa,j}\right| \le \bar{\eta} + 4\rho^3 2^{-48} \\ &< 0.525 + 2^{-52} + 4\left(\frac{1161}{179}\right)^3 2^{-48} < 0.55. \end{aligned}$$

The modified L^2 loop maintains that its active nonzero prefix is size-reduced and reprocesses every vector affected by an insertion. Hence, at termination,

$$|\mu_{i,j}| \leq 0.55 \qquad (1 \leq j < i \leq 4).$$

Together with the preceding Lovász-condition estimate, this proves that the nonzero output basis is $(0.86, 0.55)$-LLL-reduced. □

Now, we analyze the size bound of integers in modified LLL algorithm as follows:

Lemma 10. *During the execution of Algorithm 1, the maximum size of integers is at most* $4 \max_{1 \leq i \leq d} ||b_i||^2$.

Proof. See Appendix A.1. □

Lemma 10 shows that modified LLL algorithm does not create integers larger than the largest squared Euclidean norm of an input generator. This is precisely the feature needed for our ideal-arithmetic routines: once a given ideal operation outputs generators with controlled norm, it preserves that control. In this sense, modified LLL replaces HNF algorithm while avoiding the coefficient growth that would otherwise dominate the precision requirements.

Remark 1 (Floating-point storage overhead). Lemma 10 bounds only the size of the exact integer objects appearing in Algorithm 1. However, Algorithm 1 also stores the floating-point quantities. Since we use fp-precision $\ell + 1 = 53$, each floating-point quantity must be stored with a constant-size binary floating-point word supporting a 53-bit significand. For instance, this is exactly the significand precision of standard double precision, whose explicit fraction field has 52 bits. Hence the floating-point part contributes only a constant-size storage overhead per floating-point quantity, independent of the input size, but this overhead should still be counted in a concrete implementation of Algorithm 1.

3.2 Ideal multiplication

We now replace the HNF step in ideal multiplication by a reduction step based on modified LLL. In the standard approach, one forms the product lattice from the sixteen pairwise products of basis elements and then computes an HNF basis of the resulting lattice. However, ideal multiplication only requires some basis of the product lattice. This makes modified LLL a natural substitute, since it accepts a redundant generating set and returns a reduced basis of the same lattice.

We propose a new algorithm for ideal multiplication replacing the HNF algorithm with the modified LLL algorithm. The idea of the algorithm is as follows:

– We first clear denominators so that all generators lie in the ambient integral lattice,

– then shorten the two input bases by LLL reduction,
– form the sixteen products $\alpha_i\beta_j$,
– and finally apply modified LLL to these generators. The output basis is then rescaled by the common denominator r_1r_2.

Our compact ideal multiplication algorithm is given in Algorithm 2.

Algorithm 2 $\mathsf{CompactIdealMultiplication}(I_1, I_2)$

Input: A left $\mathcal{O}_1$-ideal $I_1 = \langle \alpha_1, \alpha_2, \alpha_3, \alpha_4 \rangle$ and a left $\mathcal{O}_2$-ideal $I_2 = \langle \beta_1, \beta_2, \beta_3, \beta_4 \rangle$, such that $O_R(I_1) = O_L(I_2)$.
Output: A left $\mathcal{O}_1$-ideal $I_1I_2 = \langle \gamma_1, \gamma_2, \gamma_3, \gamma_4 \rangle$.
1: $r_1 \leftarrow \mathrm{lcm}(r_{\alpha_1}, r_{\alpha_2}, r_{\alpha_3}, r_{\alpha_4})$, $r_2 \leftarrow \mathrm{lcm}(r_{\beta_1}, r_{\beta_2}, r_{\beta_3}, r_{\beta_4})$
2: $\alpha_i \leftarrow r_1\alpha_i$, $\beta_j \leftarrow r_2\beta_j$ for $1 \le i, j \le 4$
3: $(\alpha_1, \alpha_2, \alpha_3, \alpha_4) \leftarrow \mathsf{L2}(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$
4: $(\beta_1, \beta_2, \beta_3, \beta_4) \leftarrow \mathsf{L2}(\beta_1, \beta_2, \beta_3, \beta_4)$
5: $M \leftarrow (\alpha_i\beta_j)_{1\le i\le 4, 1\le j\le 4}$
6: $(b_1, b_2, b_3, b_4) \leftarrow \mathsf{ML2}(M)$
7: **return** $\frac{1}{r_1r_2}\langle b_1, b_2, b_3, b_4 \rangle$

Correctness is immediate from the lattice interpretation of ideal multiplication. After clearing denominators, the sixteen vectors $\alpha_i\beta_j$ generate the lattice

$$(r_1I_1)(r_2I_2) = r_1r_2(I_1I_2).$$

Since $\mathsf{ML2}$ returns a basis of the lattice spanned by its input vectors, the vectors (b_1, b_2, b_3, b_4) form a basis of $r_1r_2(I_1I_2)$. Dividing by r_1r_2 therefore yields a basis of the product ideal I_1I_2.

Before deriving a size bound for Algorithm 2, we first relate the size of the input generators to the intrinsic size of the input ideals; in SQIsign, the quantity naturally controlled by the preceding routines is the reduced norm of an ideal, whereas an arbitrary basis of that ideal need not consist of short elements. For this reason, Algorithm 2 applies LLL reduction to the input bases before invoking MLLL, for the sake of bounding the (reduced) norm of each basis element depending only on $\mathrm{nrd}(I_1)$ and $\mathrm{nrd}(I_2)$.

Remark 2. Algorithm 2 needs input bases which are LLL-reduced, in order to compute the uniform bound of the maximum size of integers during the execution of the algorithm. To compute LLL-reduced input bases, one can apply LLL algorithm to each basis of an ideal to be multiplied. The maximum size of integers during the execution of LLL algorithm is the norm of each basis element. So, we assume that every basis element has its reduced norm less than the square of the reduced norm of the ideal and use rejection sampling for the basis of each ideal, before applying LLL algorithm. We experimentally claim that the probability that rejection sampling occurs is negligible in Appendix B.1.

We now bound the largest integer that can occur during the execution of Algorithm 2. The main point is that, once the input bases are LLL-reduced, the entries of the product matrix M are already controlled by the norms of I_1 and I_2 by Lemma 3; Lemma 10 then propagates this control through the final ML2 reduction.

Lemma 11. *During the execution of Algorithm 2, the size of integers is at most*

$$\frac{4 \cdot 47^2 p^2}{\pi^4} r_1^2 r_2^2 \,\mathrm{nrd}(I_1)\mathrm{nrd}(I_2).$$

Proof. See Appendix A.2

□

Lemma 11 shows that the multiplication routine operates at the natural scale of the product ideal, up to the explicit factor $\frac{47^2 p^2}{\pi^4}$. In particular, the reduction step based on modified LLL does not reintroduce the coefficient growth that is typical of HNF-based normalization.

Remark 3. In order to reduce the size bound of the norm of input bases elements, we propose computing the LLL-reduced basis, before computing the ideal multiplication with modified LLL algorithm. Assuring that input bases of modified LLL algorithm are LLL-reduced, we can bound their norm as $O(\mathrm{nrd}(I_1 I_2))$, which is essentially smaller than the state-of-the-art $O(\mathrm{nrd}(I_1 I_2)^4)$ [7, Lemma 12].

From the result of this subsection, we also compute a size bound of integers occuring during the execution of algorithm IdealToIsogeny, which uses ideal multiplication as a subroutine. See Appendix C.

3.3 Ideal intersection

We also refine the integer size analysis for ideal intersection. We also replace the HNF step by an ML2 reduction as in the case of ideal multiplication, which preserves the generated lattice up to a unimodular change of basis while keeping the intermediate coefficients smaller.

However, computing the dual lattice of an ideal is necessary for the original ideal intersection algorithm [23]. Therefore, the larger precision budget is necessary, compared to the ideal multiplication. So, we propose two other approaches to compute an ideal intersection.

The first approach is to compute the ideal multiplication of I_1, I_2 satisfying $O_L(I_1) = O_R(I_2)$ and $\gcd(\mathrm{nrd}(I_1), \mathrm{nrd}(I_2)) = 1$.

Lemma 12 (Intersection of composable coprime ideals). *Let B be a quaternion algebra over $\mathbb{Q}$, and let I_1 and I_2 be nonzero integral invertible ideals in B such that $O_R(I_1) = O_L(I_2)$. If $\gcd(\mathrm{nrd}(I_1), \mathrm{nrd}(I_2)) = 1$, then*

$$I_1 \cap I_2 = I_1 I_2.$$

Proof. Set $N_1 := \mathrm{nrd}(I_1),\ N_2 := \mathrm{nrd}(I_2)$. Since I_1 and I_2 are integral and

$$O_R(I_1) = O_L(I_2) =: \mathcal{O},$$

we have

$$I_1 \subseteq \mathcal{O}, \qquad I_2 \subseteq \mathcal{O}.$$

Therefore,

$$I_1 I_2 \subseteq I_1 \mathcal{O} \subseteq I_1,$$
$$I_1 I_2 \subseteq \mathcal{O} I_2 \subseteq I_2.$$

Hence

$$I_1 I_2 \subseteq I_1 \cap I_2.$$

Conversely, let $z \in I_1 \cap I_2$. In particular,

$$N_1 \in I_1, \qquad N_2 \in I_2.$$

Since $\gcd(N_1, N_2) = 1$, there exist $u, v \in \mathbb{Z}$ such that

$$uN_1 + vN_2 = 1.$$

Consequently,

$$z = (uN_1 + vN_2)z = u(N_1 z) + v(zN_2),$$

where we used that $N_2 \in \mathbb{Z}$ is central in B. Since $N_1 \in I_1,$ and $z \in I_2$, we have

$$N_1 z \in I_1 I_2.$$

Similarly, since $z \in I_1,$ and $N_2 \in I_2$, we have

$$zN_2 \in I_1 I_2.$$

As $I_1 I_2$ is a $\mathbb{Z}$-module, it follows that

$$z \in I_1 I_2.$$

Thus

$$I_1 \cap I_2 \subseteq I_1 I_2,$$

and hence

$$I_1 \cap I_2 = I_1 I_2.$$

□

Algorithm 3 computes a reduced basis of intersection of two left $\mathcal{O}_0$-ideals without computing a dual lattice or an HNF.

First, we prove the correctness of Algorithm 3.

Lemma 13. *Algorithm 3 returns a* $(0.86, 0.55)$*-LLL-reduced basis* $(\gamma_1, \ldots, \gamma_4)$ *of* $I_1 \cap I_2$.

Algorithm 3 CompactIdealIntersection(I_1, I_2)

Input: Integral left $\mathcal{O}_0$-ideals $I_1 = \langle \alpha_1, \ldots, \alpha_4 \rangle_{\mathbb{Z}}$, $I_2 = \langle \beta_1, \ldots, \beta_4 \rangle_{\mathbb{Z}}$.
Output: An LLL-reduced $\mathbb{Z}$-basis $(\gamma_1, \ldots, \gamma_4)$ of $I_1 \cap I_2$.

1: $(\alpha_1, \alpha_2, \alpha_3, \alpha_4) \leftarrow \mathsf{ML2}(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$
2: $(\beta_1, \beta_2, \beta_3, \beta_4) \leftarrow \mathsf{ML2}(\beta_1, \beta_2, \beta_3, \beta_4)$
3: $t_{r,s} \leftarrow \mathrm{trd}(\alpha_r \overline{\beta_s})$ for all $1 \le r, s \le 4$
4: $d \leftarrow \gcd(\mathrm{nrd}(I_1), \mathrm{nrd}(I_2), (t_{r,s})_{1 \le r,s \le 4})$
5: $a \leftarrow \mathrm{nrd}(I_1)/d$, $b \leftarrow \mathrm{nrd}(I_2)/d$
6: $(\gamma_1, \ldots, \gamma_4) \leftarrow \mathsf{ML2}(b\alpha_1, \ldots, b\alpha_4, a\beta_1, \ldots, a\beta_4)$
7: **return** $(\gamma_1, \ldots, \gamma_4)$

Proof. Let $N_1 = \mathrm{nrd}(I_1), N_2 = \mathrm{nrd}(I_2)$. For any arbitrary

$$x = \sum_{r=1}^{4} u_r \alpha_r \in I_1, \qquad y = \sum_{s=1}^{4} v_s \beta_s \in I_2,$$

by applying Lemma 8,

$$\begin{aligned} \mathrm{nrd}(x+y) &= \mathrm{nrd}(x) + \mathrm{nrd}(y) + \mathrm{trd}(x\overline{y}) \\ &= \mathrm{nrd}(x) + \mathrm{nrd}(y) + \sum_{r,s=1}^{4} u_r v_s t_{r,s}. \end{aligned}$$

Since $N_1 \mid \mathrm{nrd}(x)$ and $N_2 \mid \mathrm{nrd}(y)$, the integer $d = \gcd(N_1, N_2, (t_{r,s})_{r,s})$ divides the norm of every element of $I_1 + I_2$. Hence $d \mid \mathrm{nrd}(I_1 + I_2)$.
Conversely, since $I_1, I_2 \subseteq I_1 + I_2$, the integer $\mathrm{nrd}(I_1 + I_2)$ divides N_1 and N_2. Moreover,

$$t_{r,s} = \mathrm{nrd}(\alpha_r + \beta_s) - \mathrm{nrd}(\alpha_r) - \mathrm{nrd}(\beta_s),$$

so $\mathrm{nrd}(I_1 + I_2)$ divides every $t_{r,s}$. Therefore

$$\mathrm{nrd}(I_1 + I_2) = d.$$

The lattice-index identity

$$[\mathcal{O}_0 : I_1 \cap I_2]\,[\mathcal{O}_0 : I_1 + I_2] = [\mathcal{O}_0 : I_1]\,[\mathcal{O}_0 : I_2]$$

and

$$[\mathcal{O}_0 : J] = \mathrm{nrd}(J)^2$$

for integral left $\mathcal{O}_0$-ideals J imply

$$\mathrm{nrd}(I_1 \cap I_2) = \frac{N_1 N_2}{d} = bN_1 = aN_2.$$

We now prove that $I_1 \cap I_2 = bI_1 + aI_2$. Since $I_2 \subseteq I_1 + I_2$, the ideal

$$J_2 := (I_1 + I_2)^{-1} I_2$$

is an integral left $O_R(I_1 + I_2)$-ideal of norm $\mathrm{nrd}(J_2) = \frac{N_2}{d} = b$. The standard containment $\mathrm{nrd}(J_2)O_R(I_1 + I_2) \subseteq J_2$ therefore gives

$$bO_R(I_1 + I_2) \subseteq (I_1 + I_2)^{-1} I_2.$$

Multiplying on the left by $I_1 + I_2$ yields

$$b(I_1 + I_2) \subseteq I_2.$$

In particular, we get $bI_1 \subseteq I_1 \cap I_2$. The same argument with I_1 and I_2 exchanged gives $aI_2 \subseteq I_1 \cap I_2$. Thus

$$bI_1 + aI_2 \subseteq I_1 \cap I_2.$$

Applying the preceding norm-of-a-sum calculation to $bI_1 + aI_2$ gives

$$\begin{aligned} \mathrm{nrd}(bI_1 + aI_2) &= \gcd\big(b^2 N_1,\, a^2 N_2,\, (ab\, t_{r,s})_{r,s}\big) \\ &= \mathrm{nrd}(I_1 \cap I_2) \cdot \gcd(b,\, a,\, (t_{r,s}/d)_{r,s})\,. \end{aligned}$$

By the definition of d,

$$1 = \frac{1}{d} \gcd(N_1, N_2, (t_{r,s})_{r,s}) = \gcd(a, b, (t_{r,s}/d)_{r,s})\,,$$

and hence

$$\mathrm{nrd}(bI_1 + aI_2) = N.$$

Since

$$bI_1 + aI_2 \subseteq I_1 \cap I_2$$

and both ideals have norm N, they have the same index in $\mathcal{O}_0$. Consequently,

$$bI_1 + aI_2 = I_1 \cap I_2.$$

Finally, the eight input vectors to ML2 generate

$$\begin{aligned} \big\langle b\alpha_1, \ldots, b\alpha_4,\, a\beta_1, \ldots, a\beta_4 \big\rangle_{\mathbb{Z}} &= bI_1 + aI_2 \\ &= I_1 \cap I_2. \end{aligned}$$

By Lemma 9, $(\gamma_1, \ldots, \gamma_4)$ is a $(0.86, 0.55)$-LLL-reduced basis of the same lattice. □

Lemma 14. *During the execution of Algorithm 3, the size of every integers is at most*

$$\frac{2^4 \cdot 47 \cdot p}{\pi^2} \frac{\mathrm{nrd}(I_1)\mathrm{nrd}(I_2) \max\{\mathrm{nrd}(I_1), \mathrm{nrd}(I_2)\}}{\mathrm{nrd}(I_1 + I_2)^2}.$$

Proof. See Appendix A.3.

□

4 Compact random ideal sampling algorithms

After replacing HNF algorithm by MLLL in the ideal operations, the ideal operations no longer dominate the fixed-precision budget. Instead, the largest remaining intermediate values arise from the prime-norm branch of RandomIdealGivenNorm. Also, the previous analysis of the algorithm RandomEquivalentPrimeIdeal uses a conservative bound on the norm of the sampled equivalent ideal.

We therefore revisit these routines. In subsection 4.1, we introduce the modified algorithm of RandomIdealGivenNorm, by reducing the maximum size of intermediate values of the algorithm. In subsection 4.2, we give a tighter size analysis. These refinements improve the final uniform fixed-precision bounds during the key generation and signing procedures of SQIsign.

4.1 Modification of RandomIdealGivenNorm

We first consider the prime-norm branch of RandomIdealGivenNorm algorithm [23]. In the original algorithm, after sampling γ and β, one returns

$$J' = \mathcal{O}_0(\gamma\beta) + N\mathcal{O}_0.$$

However, this ideal depends only on the residue class of $\gamma\beta$ modulo $N\mathcal{O}_0$. There is therefore no need to materialize the unreduced product: it suffices to compute a representative

$$g \equiv \gamma\beta \pmod{N\mathcal{O}_0}$$

and return $\mathcal{O}_0 g + N\mathcal{O}_0$. In particular, the final multiplication may be carried out modulo N, so the coefficients handled by the algorithm remain reduced modulo N.

The same observation also simplifies the sampling of β. In the original algorithm, the rejection condition is exactly $\mathrm{nrd}(\beta) \not\equiv 0 \pmod{N}$. Since only $\beta \bmod N\mathcal{O}_0$ is used afterwards, we may sample β directly from residue classes with nonzero reduced norm modulo N. This yields the compact variant introduced in Algorithm 4.

By the correctness of the original algorithm **RandomIdealGivenNorm**, we claim the correctness of Algorithm 4 by showing that the output gives the same distribution with **RandomIdealGivenNorm**, as the following lemma.

Lemma 15. *Let N be a prime and $\beta, \gamma \in \mathcal{O}_0$ such that $N \mid \mathrm{nrd}(\gamma)$, $N \nmid \mathrm{nrd}(\beta)$. Define*

$$J'_{\mathrm{old}} := \mathcal{O}_0(\gamma\beta) + N\mathcal{O}_0, \qquad J'_{\mathrm{new}} := \mathcal{O}_0(\overline{\gamma\beta}) + N\mathcal{O}_0,$$

where $\overline{\gamma\beta}$ is any lift in $\mathcal{O}_0$ of the class of $\gamma\beta$ modulo $N\mathcal{O}_0$. Then

$$J'_{\mathrm{old}} = J'_{\mathrm{new}}.$$

In particular, replacing $\gamma\beta \bmod N$ by $\gamma\beta$ in the last line of Algorithm 4 does not change the output distribution.

Algorithm 4 RandomIdealGivenPrimeNorm(N)

Input: A prime number $N > 3$.
Output: A random left $\mathcal{O}_0$-ideal J' of norm N.

$\gamma \leftarrow 0 + 0i + 0j + 0k$
while $\gcd\left(\mathrm{nrd}(\gamma), N^2\right) \neq N$ **do**
  Sample integers $g_1, g_2, g_3 \xleftarrow{\$} [0, N-1]$ uniformly
  $\gamma \leftarrow g_1 i + g_2 j + g_3 ij$
  if $(-\mathrm{nrd}(\gamma) : N) = 1$ **then**
    $\gamma \leftarrow \gamma + \left(\sqrt{-\mathrm{nrd}(\gamma)} \bmod N\right)$
  end if
end while
Sample integers $a, b, c, d \xleftarrow{\$} [0, N-1]$ s.t. $a^2+b^2+p(c^2+d^2) \neq 0 \pmod N$ uniformly
$\beta \leftarrow a + bi + cj + dk$
$g \leftarrow \gamma\beta \bmod N\mathcal{O}_0$
$J' \leftarrow$ the left O_0-ideal generated by g and N
return J'

Proof. If $a, a' \in \mathcal{O}_0$ satisfy $a - a' \in N\mathcal{O}_0$, then

$$\mathcal{O}_0 a + N\mathcal{O}_0 = \mathcal{O}_0 a' + N\mathcal{O}_0.$$

Indeed, writing $a = a' + N\delta$ with $\delta \in \mathcal{O}_0$, we get

$$\mathcal{O}_0 a \subseteq \mathcal{O}_0 a' + N\mathcal{O}_0,$$

hence

$$\mathcal{O}_0 a + N\mathcal{O}_0 \subseteq \mathcal{O}_0 a' + N\mathcal{O}_0.$$

The reverse inclusion follows symmetrically from $a' = a - N\delta$.
Now take $a = \gamma\beta$ and $a' = \widetilde{\gamma\beta}$. By construction,

$$\gamma\beta - \widetilde{\gamma\beta} \in N\mathcal{O}_0.$$

Therefore

$$\mathcal{O}_0(\gamma\beta) + N\mathcal{O}_0 = \mathcal{O}_0(\widetilde{\gamma\beta}) + N\mathcal{O}_0.$$

Thus, for every realization of the random choices, the modified and the original algorithms return exactly the same left $\mathcal{O}_0$-ideal. Consequently their output distributions are identical. □

We now derive the intermediate size bound on our proposed algorithm RandomIdealGivenPrimeNorm.

Lemma 16. *During the execution of Algorithm 4, the maximum size of integers is less than* $16p^2N^2$.

Proof. See Appendix A.4

□

4.2 Output norm of RandomEquivalentPrimeIdeal

We recall the crude worst-case bound for RandomEquivalentPrimeIdeal algorithm used in previous analyses [7]. The bound on output norm was derived in a coarse empirical manner: one directly observed the norm of the ideal returned by Algorithm 5 and chose a sufficiently large global upper bound as p without rigorous proof, while the observed output norm appears to be $O(\sqrt{p})$. In contrast, we provide two additional approaches on this algorithm for the rigorous size bound. First, we give a rigorous proof for an upper bound on the size of integers during the execution of Algorithm 5. Second, we replace the LLL reduction with ML2 reduction algorithm, to eliminate rational numbers representing the Gram matrix of a lattice, which can merely occur extremely large integers.

Algorithm 5 RandomEquivalentPrimeIdeal(I)

Input: I, a left $\mathcal{O}$-ideal.
Output: $J \sim I$ of small prime norm, or raise an exception if unsuccessful.

1: Initialize *counter* $\leftarrow 0$
2: $(\alpha_1, \alpha_2, \alpha_3, \alpha_4) \leftarrow \mathsf{ML2}(I)$
3: **while** *counter* $< (2 \cdot \texttt{QUAT_equiv_bound_coeff} + 1)^4$ **do**
4: *counter* $\leftarrow$ *counter* $+ 1$
5: Sample c_1, c_2, c_3, c_4 uniformly at random from $[-b, \ldots, b]$, for bound $b =$ `QUAT_equiv_bound_coeff`
6: $\beta \leftarrow \sum_{i=1}^{4} c_i \alpha_i$
7: $J \leftarrow \chi_I(\beta)$
8: **if** nrd(J) is prime **then**
9: **return** J
10: **end if**
11: **end while**
12: **raise** Exception("RandomEquivalentPrimeIdeal failed")

Lemma 17. *The reduced norm of output ideal of Algorithm 5 is less than*

$$\frac{2^4 \cdot 47 \cdot QUAT_equiv_bound_coeff^2}{\pi^2} p$$

when given an input ideal generated by Algorithm 4 with given norm D_{mix}.

Proof. Let I be the output of Algorithm 4 and $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ be its $(0.86, 0.55)$-LLL-reduced basis as in line 2. By Lemma 3, we have

$$\frac{\mathrm{nrd}(\alpha_4)}{\mathrm{nrd}(I)} \leq \frac{47}{\pi^2} \cdot p.$$

Hence, we have

$$\begin{aligned}\mathrm{nrd}(J) &\leq \frac{\mathrm{nrd}\left(\mathrm{QUAT_equiv_bound_coeff} \cdot \sum_{i=1}^{4} \alpha_i\right)}{\mathrm{nrd}(I)} \\ &\leq \frac{\mathrm{QUAT_equiv_bound_coeff}^2 \cdot \max_{1\leq i\leq 4} \mathrm{nrd}(4\alpha_i)}{\mathrm{nrd}(I)} \\ &= \frac{16 \cdot \mathrm{QUAT_equiv_bound_coeff}^2 \cdot \max_{1\leq i\leq 4} \mathrm{nrd}(\alpha_i)}{\mathrm{nrd}(I)} \\ &\leq \frac{2^4 \cdot 47 \cdot \mathrm{QUAT_equiv_bound_coeff}^2}{\pi^2}\, p \ .\end{aligned}$$

□

5 Improved fixed-precision integer arithmetic

In this section, we combine the local bounds obtained in the previous sections into global fixed-precision requirements for SQIsign. In subsection 5.1, we propagate these bounds through the full key generation and signing procedures, thereby deriving new uniform worst-case precision bounds that are sufficient for fixed-precision integer arithmetic. In subsection 5.2, we evaluate how the reduced precision affects the practical performance of SQIsign.

5.1 Improved uniform bounds

First, we derive the uniform bound on the key generation of SQIsign with our compact quaternion algorithms. (Algorithm 6)

Theorem 1 (Improved KeyGen bound). *During the execution of Algorithm 6, the maximum size of integers is less than* $2^{34}p^6$.

Proof. By Lemma 16, the maximum size of integers in line 2 is at most

$$16p^2 D_{\mathrm{mix}}^2 < 2^{34}p^6.$$[3]

By Lemma 17 and Lemma 22, the maximum size of integers in line 5 is at most

$$\frac{1024p^2}{\pi^4}\log p \cdot \frac{2^{16}\cdot 47}{\pi^2}p = 2^{26}47\pi^{-6}p^3 \log p.$$

For other lines, following the proof of [7, Theorem 1], the maximum size of integers is less than $2^{34}p^6$. This proves the claim. □

We next derive the uniform bound on the signing process of SQIsign with our compact quaternion algorithms. (Algorithm 7)

[3] One can easily check that $D_{\mathrm{mix}} < 2^{15}p^2$ by a simple computation.
[4] $I_{\mathrm{sk}} \cap I'_{\mathrm{chl}} = I_{\mathrm{sk}} I_{\mathrm{sk}}^{-1}(I_{\mathrm{sk}} \cap I'_{\mathrm{chl}}) = I_{\mathrm{sk}} \cdot [I_{\mathrm{sk}}]_* I'_{\mathrm{chl}}$

Algorithm 6 KeyGen(pp)

Input : The public parameter pp.
Output: Secret key sk and public key pk.

while true **do**
  $I_{\mathrm{sk}} \leftarrow \mathsf{RandomIdealGivenPrimeNorm}(D_{\mathrm{mix}})$
  try
    $I_{\mathrm{sk}} \leftarrow \mathsf{RandomEquivalentPrimeIdeal}(I_{\mathrm{sk}})$
    $E_{\mathrm{pk}}, \varphi_{\mathrm{sk}}(P_0), \varphi_{\mathrm{sk}}(Q_0) \leftarrow \mathsf{IdealToIsogeny}(I_{\mathrm{sk}})$
  except
    continue
  $P_{\mathrm{pk}}, Q_{\mathrm{pk}}, hint_{\mathrm{pk}} \leftarrow \mathsf{TorsionBasisToHint}(E_{\mathrm{pk}})$
  $M_{\mathrm{sk}} \leftarrow \mathsf{ChangeOfBasis}_{2^f}\Big(E_{\mathrm{pk}}, \big(\varphi_{\mathrm{sk}}(P_0), \varphi_{\mathrm{sk}}(Q_0)\big), \big(P_{\mathrm{pk}}, Q_{\mathrm{pk}}\big)\Big)$
  $pk \leftarrow \big(E_{\mathrm{pk}}, hint_{\mathrm{pk}}\big)$
  $sk \leftarrow \big(E_{\mathrm{pk}}, hint_{\mathrm{pk}}, I_{\mathrm{sk}}, M_{\mathrm{sk}}\big)$
  return sk, pk
end while

Theorem 2 (Improved worst-case Sign bound). *During the execution of Algorithm 7, the maximum size of integers is less than* $2^{35}47\pi^{-2}p^{6.5}$.

Proof. We follow the proof of [7, Theorem 2] with improved bounds on some algorithms introduced in Section 3 and Section 4. Operations and subroutines which are not analyzed in this proof is already rigorously proven in the proof of [7, Theorem 2]. We also use the fact that $2^f < p/4$ and QUAT_equiv_bound_coeff $= 64$, which are clear by [23]. By Lemma 16, Lemma 22, and the same argument as in the proof of [7, Theorem 2] with lemmas proved in [7], it is trivial that the maximum size of integers does not appear in lines except for the following lines:

- **Line 4**: By Lemma 16, the maximum size of integers during computing I_{com} is less than $2^{34}p^6$.
- **Line 13**: During the computation of $I_{\mathrm{sk}} \cap I'_{\mathrm{chl}}$, since $\gcd(\mathrm{nrd}(I_{\mathrm{sk}}), 2^f) = 1$, by Lemma 14, the maximum size of integers is less than

$$\begin{aligned}&\frac{2^4 \cdot 47 \cdot p}{\pi^2} \frac{\mathrm{nrd}(I_{\mathrm{sk}})\mathrm{nrd}(I'_{\mathrm{chl}})\max(\mathrm{nrd}(I_{\mathrm{sk}}), \mathrm{nrd}(I'_{\mathrm{chl}}))}{\mathrm{nrd}(I_{\mathrm{sk}} + I'_{\mathrm{chl}})^2} \\ &\leq \frac{2^4 \cdot 47p}{\pi^2} \cdot \frac{2^{16} \cdot 47}{\pi^2} p \cdot 2^f \cdot \frac{2^{16} \cdot 47}{\pi^2} p \leq 2^{34} 47 \pi^{-6} p^4.\end{aligned}$$

  Also, by letting $I_{\mathrm{chl}} = [I_{\mathrm{sk}}]_* I'_{\mathrm{chl}}$, since $\overline{I_{\mathrm{com}}} \subset \mathcal{O}_0$, by Lemma 4, Lemma 11, Lemma 12, and Lemma 17, the maximum size of integers during the computation of the intersection is less than

$$\begin{aligned}&\frac{4 \cdot 47^2 p^2}{\pi^4} \cdot 2^2 \cdot 2^2 \cdot \mathrm{nrd}(I_{\mathrm{com}}) \cdot \mathrm{nrd}(I_{\mathrm{sk}} \cdot I_{\mathrm{chl}}) \\ &\leq \frac{4 \cdot 47^2 p^2}{\pi^4} 2^2 \cdot 2^2 \cdot \frac{2^{16}47}{\pi^2} p \cdot \left(\frac{2^{16}47}{\pi^2} p \cdot 2^f \right) \\ &< 2^{36} 47^4 \pi^{-8} p^5,\end{aligned}$$

since $\mathrm{nrd}(I_{\mathrm{com}}, I_{\mathrm{sk}} \cdot I_{\mathrm{chl}}) = 1$ with an overwhelming probability.

– **Line 23**: During the computation of $I_{\mathrm{com,rsp}} \cap I_{\mathrm{aux}}$, since $2^{e_{\mathrm{rsp}}} \leq \sqrt{p}+1$ [23], by Lemma 14, the maximum size of integers is less than

$$\begin{aligned}
&\frac{2^4 \cdot 47 \cdot p}{\pi^2} \frac{\mathrm{nrd}(I_{\mathrm{com,rsp}})\mathrm{nrd}(I_{\mathrm{aux}}) \max\{\mathrm{nrd}(I_{\mathrm{com,rsp}}), \mathrm{nrd}(I_{\mathrm{aux}})\}}{\mathrm{nrd}(I_{\mathrm{com,rsp}} + I_{\mathrm{aux}})^2} \\
&\leq \frac{2^4 \cdot 47p}{\pi^2} \cdot (q_{\mathrm{rsp}} D_{\mathrm{mix}}) \cdot (2^e_{\mathrm{rsp}}) \cdot (q_{\mathrm{rsp}} D_{\mathrm{mix}}) \\
&\leq \frac{2^4 \cdot 47p}{\pi^2} \cdot D_{\mathrm{rsp}} D_{\mathrm{mix}} \cdot 2^{e_{\mathrm{rsp}}} \cdot D_{\mathrm{rsp}} D_{\mathrm{mix}} \\
&\leq \frac{2^4 \cdot 47p}{\pi^2} D_{\mathrm{mix}}^2 2^{3e_{\mathrm{rsp}}} \leq \frac{2^4 \cdot 47p}{\pi^2} \cdot (2^{15} p^2)^2 \cdot 2p^{1.5} \\
&< 2^{35} 47 \pi^{-2} p^{6.5}.
\end{aligned}$$

□

5.2 Implementation and performance analysis

To evaluate the practical impact of the reduced precision bounds derived in Section 5.1, we compare two fixed-precision integer-arithmetic implementations of SQIsign: the implementation of previous work [7],[5] which uses the precision budgets of 7026/10713/14150 bits for the NIST-I/III/V parameter sets, and our implementation,[6] which uses the precision budgets of 1665/2521/3319 bits implied by Theorem 1 and Theorem 2. Both implementations are built on top of the original SQIsign source code.[7] The source code of our implementation is publicly available at:

https://github.com/pqc-lab-ku/compact-SQIsign

Results. Table 5 compares the average running time of KeyGen, Sign, and Verify for both implementations at each NIST security level.

Each value averages 100 runs on an Intel Core i9-14900KF under Ubuntu 24.04.4, using GCC 13.3, -O3 -DNDEBUG, and the broadwell finite-field backend. Each run used a separate process, a deterministic seed, a 32-byte message, pinned CPU affinity, and CLOCK_MONOTONIC wall-clock timing. Diagnostic I/O was disabled.

Acknowledgements. The authors used generative AI tools, to assist with the following tasks:

– We used ChatGPT Pro 5.5/5.6 for grammar and spelling of author-written manuscript text.

[5] https://github.com/munsanwon2/SQIsign-Fixed-Precision

[6] Our implementation is based on the previous work (https://github.com/munsanwon2/SQIsign-Fixed-Precision).

[7] https://github.com/SQIsign/the-sqisign

Table 5: Benchmark comparison at NIST security levels I, III, and V. All timings are reported in milliseconds (ms).

Security level	Operation	Previous work [7]	Ours	Original SQIsign [23]
NIST-I	KeyGen	15,719.26ms	148.19ms	9.23ms
	Sign	31,637.71ms	306.93ms	21.16ms
	Verify	1.06ms	1.05ms	1.07ms
NIST-III	KeyGen	60,294.40ms	404.65ms	25.71ms
	Sign	122,501.06ms	875.62ms	57.47ms
	Verify	3.30ms	3.31ms	3.33ms
NIST-V	KeyGen	172,194.93ms	757.04ms	39.48ms
	Sign	369,549.62ms	1,715.75ms	90.47ms
	Verify	6.29ms	6.30ms	6.30ms

– We used ChatGPT Pro 5.5/5.6 for generating LaTeX tables with contents written by authors.
– We used ChatGPT Pro 5.5/5.6 for drafting and revising experimental source code based on the authors' experimental design.
– We used Claude Opus 4.7 and ChatGPT Pro 5.6 for drafting and revising portions of the implementation source code based on the authors' specifications and intended algorithms.

All AI-assisted text and code were reviewed, edited, tested, and validated by the authors. The authors take full responsibility for the correctness and appropriateness of all material.

Algorithm 7 $\mathsf{Sign}(sk, msg)$

Input: Secret signing key sk and message $msg \in \{0,1\}^*$.
Output: Signature σ.

Parse sk as $(E_{\mathrm{pk}},\ hint_{\mathrm{pk}},\ I_{\mathrm{sk}},\ M_{\mathrm{sk}})$
$P_{\mathrm{pk}}, Q_{\mathrm{pk}} \leftarrow \mathsf{TorsionBasisFromHint}(E_{\mathrm{pk}}, hint_{\mathrm{pk}})$
while true **do**
  $I_{\mathrm{com}} \leftarrow \mathsf{RandomIdealGivenPrimeNorm}(D_{\mathrm{mix}}, true)$
  try
    $I_{\mathrm{com}} \leftarrow \mathsf{RandomEquivalentPrimeIdeal}(I_{\mathrm{com}})$
    $E_{\mathrm{com}}, P_{\mathrm{com}}, Q_{\mathrm{com}} \leftarrow \mathsf{IdealToIsogeny}(I_{\mathrm{com}})$
  except
    continue
  $chl \leftarrow \mathsf{HASH}\big(pk \,\|\, j(E_{\mathrm{com}}) \,\|\, msg\big)$
  $(c_1, c_2) \leftarrow M_{\mathrm{sk}} \cdot (1, chl)$
  $I'_{\mathrm{chl}} \leftarrow \mathsf{KernelDecomposedToIdeal}_{2^f}(c_1, c_2)$
  $\alpha_{\mathrm{rsp}} \leftarrow \mathsf{RandomEquivalentQuaternion}(\overline{I_{\mathrm{com}}} \cap (I_{\mathrm{sk}} \cap I'_{\mathrm{chl}}))$[4]
  $\alpha_{\mathrm{rsp}},\ n_{\mathrm{bt}} \leftarrow \mathsf{ComputeBacktrackingAndNormalize}(\alpha_{\mathrm{rsp}})$
  $d_{\mathrm{rsp}} \leftarrow \mathrm{nrd}(\alpha_{\mathrm{rsp}})/D_{\mathrm{mix}}^2 \cdot 2^{f-n_{\mathrm{bt}}}$ // d_{rsp} *is the degree of* φ_{rsp}
  $r_{\mathrm{rsp}} \leftarrow \mathsf{DyadicValuation}(d_{\mathrm{rsp}})$
  $q_{\mathrm{rsp}} \leftarrow d_{\mathrm{rsp}}/2^{r_{\mathrm{rsp}}}$
  $I_{\mathrm{com,rsp}} \leftarrow \mathcal{O}\alpha_{\mathrm{rsp}} + \mathcal{O}(q_{\mathrm{rsp}} D_{\mathrm{mix}})$ // $I_{\mathrm{com,rsp}}$ *corresponds to* $\varphi_{\mathrm{com}} \circ \varphi_{\mathrm{rsp}}$
  $e'_{\mathrm{rsp}} \leftarrow e_{\mathrm{rsp}} - r_{\mathrm{rsp}} - n_{\mathrm{bt}}$ // e'_{rsp} *is the degree of the two-dimensional isogeny*
  if $e'_{\mathrm{rsp}} > 0$ **then**
    try
      $I_{\mathrm{aux}} \leftarrow \mathsf{RandomIdealGivenNorm}(2^{e'_{\mathrm{rsp}}} - q_{\mathrm{rsp}}, false)$
      $E'_{\mathrm{aux}}, P'_{\mathrm{aux}}, Q'_{\mathrm{aux}} \leftarrow \mathsf{IdealToIsogeny}(I_{\mathrm{com,rsp}} \cap I_{\mathrm{aux}})$
    except
      continue
    $(E_{\mathrm{aux}}, P_{\mathrm{aux}}, Q_{\mathrm{aux}}, E_{\mathrm{chl}}, P_{\mathrm{chl}}, Q_{\mathrm{chl}}) \leftarrow \mathsf{SplitAuxiliaryIsogeny}$
    $(E_{\mathrm{com}}, E'_{\mathrm{aux}}, P_{\mathrm{com}}, Q_{\mathrm{com}}, P'_{\mathrm{aux}}, Q'_{\mathrm{aux}}, q_{\mathrm{rsp}}, e'_{\mathrm{rsp}}, r_{\mathrm{rsp}}, n_{\mathrm{bt}})$
  else
    try
      $E_{\mathrm{chl}}, P_{\mathrm{chl}}, Q_{\mathrm{chl}} \leftarrow \mathsf{IdealToIsogeny}(I_{\mathrm{com,rsp}})$
      $E_{\mathrm{aux}}, P_{\mathrm{aux}}, Q_{\mathrm{aux}} \leftarrow E_{\mathrm{chl}}, P_{\mathrm{chl}}, Q_{\mathrm{chl}}$
    except
      continue
  end if
  if $r_{\mathrm{rsp}} > 0$ **then**
    $E_{\mathrm{chl}}, P_{\mathrm{chl}}, Q_{\mathrm{chl}} \leftarrow \mathsf{ComputeEvenNonBacktrackingResponse}$
    $(E_{\mathrm{chl}}, P_{\mathrm{chl}}, Q_{\mathrm{chl}}, \alpha_{\mathrm{rsp}}, e'_{\mathrm{rsp}}, r_{\mathrm{rsp}})$
  end if
  $E_{\mathrm{chl}}, P_{\mathrm{chl}}, Q_{\mathrm{chl}} \leftarrow \mathsf{ComputeChallengeIsogeny}$
  $(E_{\mathrm{pk}}, chl, P_{\mathrm{pk}}, Q_{\mathrm{pk}}, E_{\mathrm{chl}}, P_{\mathrm{chl}}, Q_{\mathrm{chl}}, n_{\mathrm{bt}})$
  $M_{\mathrm{chl}},\ hint_{\mathrm{aux}},\ hint_{\mathrm{chl}} \leftarrow \mathsf{SetChangeOfBasisMatrix}$
  $(E_{\mathrm{aux}}, E_{\mathrm{chl}}, P_{\mathrm{aux}}, Q_{\mathrm{aux}}, P_{\mathrm{chl}}, Q_{\mathrm{chl}}, e'_{\mathrm{rsp}} + r_{\mathrm{rsp}})$
  $\sigma \leftarrow (E_{\mathrm{aux}}, n_{\mathrm{bt}}, r_{\mathrm{rsp}}, M_{\mathrm{chl}}, chl, hint_{\mathrm{aux}}, hint_{\mathrm{chl}})$
  return σ
end while

References

1. National Institute of Standards and Technology. Module-Lattice-Based digital signature standard. Technical report, Gaithersburg, MD, 2024.
2. Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon. Technical report, National Institute of Standards and Technology, 2020.
3. National Institute of Standards and Technology (US). Stateless hash-based digital signature standard. Technical report, Washington, D.C., August 2024.
4. Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. SQISign: compact post-quantum signatures from quaternions and isogenies. In *International conference on the theory and application of cryptology and information security*, pages 64–93. Springer, 2020.
5. Randy Yates. Fixed-point arithmetic: An introduction. *Digital Signal Labs*, 81(83):198, 2009.
6. José Bacelar Almeida, Manuel Barbosa, Gilles Barthe, François Dupressoir, and Michael Emmi. Verifying {Constant-Time} implementations. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 53–70, 2016.
7. Won Kim, Jeonghwan Lee, Hyeonhak Kim, and Changmin Lee. SQIsign with Fixed-Precision Integer Arithmetic. *Cryptology ePrint Archive*, 2025.
8. Andrea Basso, Chenfeng He, David Jacquemin, Fatna Kouider, Péter Kutas, Anisha Mukherjee, Sina Schaeffler, and Sujoy Sinha Roy. Constant-time quaternion algorithms for SQIsign. Cryptology ePrint Archive, Paper 2025/2192, 2025.
9. Antonin Leroux. Efficient quaternion algorithms for the deuring correspondence, and application to the evaluation of modular polynomials. *Cryptology ePrint Archive*, 2026.
10. Phong Q Nguyen and Damien Stehlé. An LLL algorithm with quadratic complexity. *SIAM Journal on Computing*, 39(3):874–903, 2009.
11. Michael Pohst. A modification of the LLL reduction algorithm. *Journal of Symbolic Computation*, 4(1):123–127, 1987.
12. Benjamin Wesolowski. The supersingular isogeny problem in time and memory $p^{1/3+o(1)}$. Cryptology ePrint Archive, Paper 2026/1486, 2026.
13. Marius A Aardal, Diego F Aranha, Yansong Feng, Yiming Gao, and Yanbin Pan. Better bounds for finding fixed-degree isogenies via coppersmith's method. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 541–572. Springer, 2026.
14. Joseph H Silverman. *The arithmetic of elliptic curves*, volume 106. Springer, 2009.
15. Luca De Feo. Mathematics of isogeny based cryptography. *arXiv preprint arXiv:1711.04062*, 2017.
16. Jacques Vélu. Isogénies entre courbes elliptiques. *Comptes-Rendus de l'Académie des Sciences*, 273:238–241, 1971.
17. Arjen K Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. 1982.
18. John Voight. *Quaternion algebras*. Springer, 2021.
19. Antonin Leroux. Quaternion algebra and isogeny-based cryptography. *Theses, Ecole doctorale de l'Institut Polytechnique de Paris*, 2022.
20. Max Deuring. Die typen der multiplikatorenringe elliptischer funktionenkörper. In *Abhandlungen aus dem mathematischen Seminar der Universität Hamburg*, volume 14, pages 197–272. Springer Berlin/Heidelberg, 1941.

21. Tomoyoshi Ibukiyama. On maximal orders of division quaternion algebras over the rational number field with certain optimal embeddings. *Nagoya Mathematical Journal*, 88:181–195, 1982.
22. Pierrick Dartois, Antonin Leroux, Damien Robert, and Benjamin Wesolowski. SQISignHD: new dimensions in cryptography. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 3–32. Springer, 2024.
23. Marius A. Aardal, Gora Adj, Diego F. Aranha, Andrea Basso, Isaac Andrés Canales Martínez, Jorge Chávez-Saab, Maria Corte-Real Santos, Pierrick Dartois, Luca De Feo, Max Duparc, Jonathan Komada Eriksen, Tako Boris Fouotsa, Décio Luiz Gazzoni Filho, Basil Hess, David Kohel, Antonin Leroux, Patrick Longa, Luciano Maino, Michael Meyer, Kohei Nakagawa, Hiroshi Onuki, Lorenz Panny, Sikhar Patranabis, Christophe Petit, Giacomo Pope, Krijn Reijnders, Damien Robert, Francisco Rodríguez-Henríquez, Sina Schaeffler, and Benjamin Wesolowski. SQIsign. Technical report, National Institute of Standards and Technology, 2025.
24. Giacomo Borin, Maria Corte-Real Santos, Jonathan Komada Eriksen, Riccardo Invernizzi, Marzio Mula, Sina Schaeffler, and Frederik Vercauteren. Qlapoti: Simple and efficient translation of quaternion ideals to isogenies. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 174–205. Springer, 2025.

A Additional Proofs

In this section, we provide proofs deferred from the main body of this paper.

A.1 Proof of Lemma 10

The exact integer objects in Algorithm 1 are the vectors b_i, or equivalently the exact integral Gram matrix G. The quantities $\bar{r}_{i,j}$, $\bar{\mu}_{i,j}$, $\bar{s}_j^{(\kappa)}$, and $\bar{\delta}$ are fixed-precision floating-point numbers, and the indices κ, ζ do not affect the integer-size bound.

We check the only steps that can affect exact integer data.

(1) Initialization. In line 1, the algorithm computes

$$G_{i,j} = \langle b_i, b_j \rangle.$$

By Cauchy–Schwarz,

$$|G_{i,j}| \le \|b_i\| \, \|b_j\|.$$

(2) Lazy size-reduction. Line 3 calls Algorithm 8. By Lemma 18, this subroutine does not increase the size of any exact integer object beyond $4B_{\text{in}}$.

(3) Insertion and remaining updates. Lines 5–12 only compare or copy floating-point quantities. Line 13 inserts $b_{\kappa'}$ right before b_κ and updates G accordingly; this is just a permutation of the current vectors and of the corresponding rows and columns of G, so it creates no new integer magnitude. Lines 14–17 only test whether $b_\kappa = 0$ and update the indices ζ and κ.

Therefore every iteration of Algorithm 1 preserves the invariant that all exact integer objects are bounded by B_{in}. Since the invariant holds after initialization, it holds throughout the algorithm. Hence the exact integer size in Algorithm 1 is bounded by

$$4 \max_{1 \le i \le d} \|b_i\|^2.$$

□

Lemma 18 (Integer-size bound for LazySizeReduce). *Let*

$$B_{\text{in}} := \max_{1 \le i \le d} \|b_i\|^2$$

be the initial input bound of Algorithm 1. Assume that before a call to Algorithm 8, every exact integer object currently stored by the algorithm, namely the vector representation or equivalently the exact Gram-matrix representation, is bounded by B_{in}. Then the returned objects

$$b_\kappa, \qquad G, \qquad \bar{r}_{\kappa,*}, \quad \bar{\mu}_{\kappa,*}, \quad \bar{s}_*^{(\kappa)}$$

do not increase the exact integer size beyond $4B_{\text{in}}$.

Algorithm 8 $\mathsf{LazySizeReduce}(\kappa, \zeta, (b_1, \ldots, b_d), G, \bar{r}, \bar{\mu})$ [10, Figure 5]

Input: An index κ, vectors $(b_1, \ldots, b_d)$, their exact Gram matrix $G = (G_{a,b})_{1 \le a,b \le d}$, floating-point values $\bar{r}_{i,j}$ and $\bar{\mu}_{i,j}$ for $\zeta + 1 \le j \le i < \kappa$, and fp-precision $\ell + 1 = 53$.

Output: The updated vector b_κ, the updated Gram matrix G, the values $\bar{r}_{\kappa,j}$ for $\zeta{+}1 \le j \le \kappa$, the values $\bar{\mu}_{\kappa,j}$ for $\zeta{+}1 \le j < \kappa$, and the values $\bar{s}_j^{(\kappa)}$ for $\zeta{+}1 \le j \le \kappa$.

$\bar{\eta} \leftarrow \diamond(\diamond(0.55) + 1/2)\,/2$
while true do
  $\big((\bar{r}_{\kappa,j})_{\zeta+1 \le j \le \kappa}, (\bar{\mu}_{\kappa,j})_{\zeta+1 \le j < \kappa}, (\bar{s}_j^{(\kappa)})_{\zeta+1 \le j \le \kappa}\big) \leftarrow \mathsf{PartialCholesky}(\kappa, \zeta, G, \bar{r}, \bar{\mu})$
  if $\max_{\zeta+1 \le j < \kappa} |\bar{\mu}_{\kappa,j}| \le \bar{\eta}$ **then**
    return $\big(b_\kappa, G, (\bar{r}_{\kappa,j})_{\zeta+1 \le j \le \kappa}, (\bar{\mu}_{\kappa,j})_{\zeta+1 \le j < \kappa}, (\bar{s}_j^{(\kappa)})_{\zeta+1 \le j \le \kappa}\big)$
  end if
  for $i \leftarrow \kappa - 1$ **downto** $\zeta + 1$ **do**
    $X_i \leftarrow \lfloor \bar{\mu}_{\kappa,i} \rceil$
    for $j \leftarrow \zeta + 1$ **to** $i - 1$ **do**
      $\bar{\mu}_{\kappa,j} \leftarrow \diamond(\bar{\mu}_{\kappa,j} - \diamond(X_i \bar{\mu}_{i,j}))$
    end for
  end for
  $G' \leftarrow G$
  for $h \leftarrow 1$ **to** d **do**
    if $h \ne \kappa$ **then**
      $G'_{\kappa,h} \leftarrow G_{\kappa,h} - \sum_{i=\zeta+1}^{\kappa-1} X_i G_{i,h}$
      $G'_{h,\kappa} \leftarrow G'_{\kappa,h}$
    end if
  end for
  $G'_{\kappa,\kappa} \leftarrow G_{\kappa,\kappa} - 2\sum_{i=\zeta+1}^{\kappa-1} X_i G_{\kappa,i} + \sum_{i=\zeta+1}^{\kappa-1} \sum_{j=\zeta+1}^{\kappa-1} X_i X_j G_{i,j}$
  $b_\kappa \leftarrow b_\kappa - \sum_{i=\zeta+1}^{\kappa-1} X_i b_i$
  $G \leftarrow G'$
end while

Proof. Algorithm 8 manipulates two types of data: exact integer data, namely b_κ, the coefficients X_i, and the Gram matrices G and G', and fixed-precision floating-point data, namely $\bar{r}$, $\bar{\mu}$, $\bar{s}$, and $\bar{\eta}$. The floating-point quantities do not contribute to the integer-size bound.
Line 1 only assigns the floating-point value $\bar{\eta}$, and Line 2 only starts the main loop. Line 3 invokes Algorithm 9, which does not increase the size bound by Lemma 19. Lines 4–6 only test the resulting floating-point values and possibly return. Hence Lines 1–6 do not create any new exact integer magnitude.
The only nontrivial exact-integer computations occur in Line 8 and Lines 13–22, where

$$X_i \leftarrow \lfloor \bar{\mu}_{\kappa,i} \rceil, \qquad b_\kappa \leftarrow b_\kappa - \sum_{i=\zeta+1}^{\kappa-1} X_i b_i,$$

and the corresponding row and column of the exact Gram matrix are computed in G' before assigning $G \leftarrow G'$. Size bound of X_i are precisely analyzed in [10, Theorem 4], concluding that it does not increase the size bound of Algorithm 1 with our SQIsign parameters. Also, size bound of each $||b_\kappa||$ satisfies the claimed statement by Lemma 20. By the SizeReduce integer-size bound, this update does not increase the fixed exact-integer budget: after the update, the reduced vector b_κ, or equivalently the updated row and column of G, is again represented within the bound B_{in}. All other rows and columns of G are unchanged.
Finally, Line 23 closes the loop and, if no return has occurred, repeats the same procedure from Line 2. Therefore, by induction over the iterations of Algorithm 8, the subroutine returns without increasing the exact integer size beyond B_{in}. □

Lemma 19. *Let $d \geq 1$, let*

$$C = (c_1, \ldots, c_d) \in \mathbb{Z}^{4\times d},$$

and let

$$G = \left(c_a^T G_{\mathcal{O}_0} c_b\right)_{1\leq a,b\leq d} \in \mathbb{Z}^{d\times d},$$

where $G_{\mathcal{O}_0}$ is positive definite. Define

$$B_C := \max_{1\leq a\leq d} c_a^T G_{\mathcal{O}_0} c_a = \max_{1\leq a\leq d} G_{a,a}.$$

For every $0 \leq \zeta < \kappa \leq d$, each exact Gram-matrix entry read during Algorithm 9 has absolute value at most B_C, and the algorithm creates no new exact integer.

Proof. The only exact arithmetic data read by Algorithm 9 are the entries $G_{\kappa,j}$ for $\zeta + 1 \leq j < \kappa$ and $G_{\kappa,\kappa}$. Since $G_{\mathcal{O}_0}$ is positive definite, the Cauchy–Schwarz inequality for the inner product

$$\langle x, y\rangle_{\mathcal{O}_0} := x^T G_{\mathcal{O}_0} y$$

Algorithm 9 PartialCholesky $(\kappa, \zeta, G, \bar{r}, \bar{\mu})$

Input: Indices $0 \le \zeta < \kappa \le d$, an exact integral Gram matrix $G = (G_{a,b})_{1 \le a,b \le d} \in \mathbb{Z}^{d \times d}$, and previously computed floating-point values $\bar{r}_{j,h}$ $(\zeta + 1 \le h \le j < \kappa)$ and $\bar{\mu}_{j,h}$ $(\zeta + 1 \le h < j < \kappa)$.
Output: Floating-point values $\bar{r}_{\kappa,j}$ $(\zeta + 1 \le j \le \kappa)$, $\bar{\mu}_{\kappa,j}$ $(\zeta + 1 \le j < \kappa)$, and $\bar{s}_j^{(\kappa)}$ $(\zeta + 1 \le j \le \kappa)$.

for $j \leftarrow \zeta + 1$ **to** $\kappa - 1$ **do**
  $\bar{r}_{\kappa,j} \leftarrow \diamond(G_{\kappa,j})$
  for $h \leftarrow \zeta + 1$ **to** $j - 1$ **do**
    $\bar{r}_{\kappa,j} \leftarrow \diamond(\bar{r}_{\kappa,j} - \diamond(\bar{\mu}_{j,h}\bar{r}_{\kappa,h}))$
  end for
  $\bar{\mu}_{\kappa,j} \leftarrow \diamond(\bar{r}_{\kappa,j}/\bar{r}_{j,j})$
end for
$\bar{s}_{\zeta+1}^{(\kappa)} \leftarrow \diamond(G_{\kappa,\kappa})$
for $j \leftarrow \zeta + 2$ **to** κ **do**
  $\bar{s}_j^{(\kappa)} \leftarrow \diamond\Big(\bar{s}_{j-1}^{(\kappa)} - \diamond(\bar{\mu}_{\kappa,j-1}\bar{r}_{\kappa,j-1})\Big)$
end for
$\bar{r}_{\kappa,\kappa} \leftarrow \bar{s}_\kappa^{(\kappa)}$
return $\Big((\bar{r}_{\kappa,j})_{\zeta+1 \le j \le \kappa}, (\bar{\mu}_{\kappa,j})_{\zeta+1 \le j < \kappa}, (\bar{s}_j^{(\kappa)})_{\zeta+1 \le j \le \kappa}\Big)$

gives, for every $1 \le a, b \le d$,

$$\begin{aligned} |G_{a,b}| &= \left|c_a^T G_{\mathcal{O}_0} c_b\right| \\ &\le \sqrt{c_a^T G_{\mathcal{O}_0} c_a}\sqrt{c_b^T G_{\mathcal{O}_0} c_b} \\ &\le B_C. \end{aligned}$$

After each accessed Gram-matrix entry is converted by $\diamond$, all remaining arithmetic in the subroutine is performed on floating-point values. Hence no additional exact integer is created. □

Lemma 20. *Let*

$$(b_1^{(1)}, \ldots, b_d^{(1)}) \subset \mathbb{Z}^n$$

be nonzero vectors generating a lattice of rank $k \le d$, and set

$$B_{\text{in}} := \max_{1 \le h \le d} \|b_h^{(1)}\|^2.$$

Assume that Algorithm 1 is executed with a valid reduction pair and sufficient floating-point precision. At the beginning of every outer loop iteration t, every current vector satisfies

$$\|b_i^{(t)}\|^2 \le 4B_{\text{in}}.$$

Consequently, every entry of the exact current Gram matrix satisfies

$$\left|\langle b_i^{(t)}, b_j^{(t)} \rangle\right| \le 4B_{\text{in}}.$$

Proof. Theorem 6(2) of [10] proves the corresponding statement for ordinary L^2, with the number of basis vectors in place of k. The proof of Theorem 10 of [10] states that Theorem 6 remains valid for modified L^2 and observes that the active prefix contains at most k nonzero vectors. Repeating the final norm estimate in the proof of Theorem 6 over these nonzero Gram–Schmidt components gives

$$\|b_i^{(t)}\|^2 \le 4 \max_h \|b_h^{(1)*}\|^2 \le 4B_{\mathrm{in}}.$$

The Gram-matrix bound follows from the Cauchy–Schwarz inequality. □

A.2 Proof of Lemma 11

(1) Lines 3–4. Computing LLL-reduced basis, the maximum size of integers is at most

$$\max\left(\max_{1\le i\le 4} \mathrm{nrd}(\alpha_i), \max_{1\le j\le 4} \mathrm{nrd}(\beta_j)\right).$$

(2) Lines 5–6. By Lemma 3, for every $1 \le i, j \le 4$,

$$\mathrm{nrd}(\alpha_i) \le \frac{47p}{\pi^2}\mathrm{nrd}(r_1 I_1), \qquad \mathrm{nrd}(\beta_j) \le \frac{47p}{\pi^2}\mathrm{nrd}(r_2 I_2).$$

On the other hand, for each $1 \le i, j \le 4$, $\|\alpha_i\beta_j\|$ is equal to

$$\sqrt{\mathrm{nrd}(\alpha_i)\mathrm{nrd}(\beta_j)}.$$

Therefore every entry of the matrix $M = (\alpha_i\beta_j)_{1\le i,j\le 4}$ in line 5 is bounded by

$$\sqrt{\mathrm{nrd}(\alpha_i)\mathrm{nrd}(\beta_j)} \le \frac{47p}{\pi^2}\sqrt{\mathrm{nrd}(r_1 I_1)\mathrm{nrd}(r_2 I_2)}.$$

By Lemma 10, during the execution of ML2, the maximum size is at most

$$\begin{aligned} &4 \max_{1\le i,j\le 4} ||\alpha_i\beta_j||_\infty^2 \le 4 \max_{1\le i,j\le 4} \mathrm{nrd}(\alpha_i\beta_j) \\ &\le 4 \cdot \left(\frac{47p}{\pi^2}\right)^2 \mathrm{nrd}(r_1 I_1)\mathrm{nrd}(r_2 I_2) = \frac{4 \cdot 47^2 p^2}{\pi^4}\, r_1^2 r_2^2\, \mathrm{nrd}(I_1)\mathrm{nrd}(I_2). \end{aligned}$$

(3) Line 7. Finally, line 7 only rescales the output by the denominator $r_1 r_2$, which is smaller than the above bound. Hence, throughout Algorithm 2, the size of integers is at most

$$\frac{4 \cdot 47^2 p^2}{\pi^4}\, r_1^2 r_2^2\, \mathrm{nrd}(I_1)\mathrm{nrd}(I_2).$$

□

A.3 Proof of Lemma 14

After LLL-reductions in lines 1–2, we set

$$B_1 := \max_{1\le r\le 4} \mathrm{nrd}(\alpha_r), \qquad B_2 := \max_{1\le s\le 4} \mathrm{nrd}(\beta_s),$$

and

$$B := \max\left\{b^2 B_1,\, a^2 B_2\right\}.$$

We first show that every exact integer occurring before the ML2 call is bounded by $4B$.

In line 3, since $I_1, I_2 \subseteq \mathcal{O}_0$, write

$$\alpha_r = \frac{x_0 + x_1 i + x_2 j + x_3 k}{2}, \qquad \beta_s = \frac{y_0 + y_1 i + y_2 j + y_3 k}{2},$$

with integral numerator coordinates. Then

$$4\mathrm{nrd}(\alpha_r) = x_0^2 + x_1^2 + p(x_2^2 + x_3^2),$$
$$4\mathrm{nrd}(\beta_s) = y_0^2 + y_1^2 + p(y_2^2 + y_3^2),$$

and

$$2t_{r,s} = x_0 y_0 + x_1 y_1 + p(x_2 y_2 + x_3 y_3).$$

By the weighted Cauchy–Schwarz inequality,

$$\begin{aligned}|x_0 y_0| + |x_1 y_1| + p\left(|x_2 y_2| + |x_3 y_3|\right) &\le \sqrt{x_0^2 + x_1^2 + p(x_2^2 + x_3^2)}\sqrt{y_0^2 + y_1^2 + p(y_2^2 + y_3^2)} \\ &= 4\sqrt{\mathrm{nrd}(\alpha_r)\mathrm{nrd}(\beta_s)} \le 4\sqrt{B_1 B_2}.\end{aligned}$$

The gcd computation in line 4 creates no integer larger than its inputs, and no larger than squares of its inputs occurs during the computation. Moreover, since $\mathrm{nrd}(I_1)$ and $\mathrm{nrd}(I_2)$ divide the reduced norms of all elements of I_1 and I_2, respectively,

$$\mathrm{nrd}(I_1) \le B_1, \qquad \mathrm{nrd}(I_2) \le B_2.$$

Consequently,

$$a \le \mathrm{nrd}(I_1) \le B, \qquad b \le \mathrm{nrd}(I_2) \le B.$$

The exact divisions in line 5 do not increase integer size, and

$$N = b \cdot \mathrm{nrd}(I_1) \le b^2 B_1 \le B,$$

so lines 4–6 are also bounded by $4B$.

It remains to bound the size of integers representing an output basis. The eight input generators satisfy

$$\mathrm{nrd}(b\alpha_r) = b^2 \mathrm{nrd}(\alpha_r) \le b^2 B_1 \le B$$

and

$$\mathrm{nrd}(a\beta_s) = a^2 \mathrm{nrd}(\beta_s) \le a^2 B_2 \le B.$$

Since these generators belong to $\mathcal{O}_0$, their common denominator in the basis $(1, i, j, k)$ divides 2. Equivalently, the ML2 call may be performed after multiplying all generators by a common denominator $r \leq 2$. Their squared norms are then at most

$$r^2 B \leq 4B.$$

By Lemma 10, no exact integer occurring during the execution of ML2 exceeds the 4 times of largest squared norm of an input generator.

By Lemma 3, the LLL-reduced input bases satisfy

$$B_1 \leq \frac{47p}{\pi^2}\mathrm{nrd}(I_1), \qquad B_2 \leq \frac{47p}{\pi^2}\mathrm{nrd}(I_2).$$

It follows that

$$B \leq \frac{47p}{\pi^2}\max\left\{b^2\mathrm{nrd}(I_1),\, a^2\mathrm{nrd}(I_2)\right\} = \frac{47p}{\pi^2}\max\left\{bN,\, aN\right\} = \frac{47p}{\pi^2}N\max\{a,b\}.$$

Therefore,

$$16B \leq \frac{16\cdot 47p}{\pi^2}N\max\{a,b\}.$$

Finally,

$$N = \frac{\mathrm{nrd}(I_1)\mathrm{nrd}(I_2)}{d} \text{ and } \max\{a,b\} = \frac{\max\{\mathrm{nrd}(I_1),\mathrm{nrd}(I_2)\}}{d} \text{ and } d = \mathrm{nrd}(I_1{+}I_2),$$

which gives the equivalent expression of the size bound

$$\frac{2^4\cdot 47\cdot p}{\pi^2}\frac{N_1N_2\max\{\mathrm{nrd}(I_1),\mathrm{nrd}(I_2)\}}{\mathrm{nrd}(I_1+I_2)^2}.$$

□

A.4 Proof of Lemma 16

(1) Lines 3–6. Since $g_1, g_2, g_3 < N$, each size of an integer appearing in these lines is less than N^2 for the computation of $\sqrt{-\mathrm{nrd}(\gamma)} \bmod N$.

(2) Lines 9–12. Since $a, b, c, d < N$, β has coefficients less than N. Hence, in line 11, $\gamma\beta$ has coefficients less than N^2. Computing the ideal $\mathcal{O}_0 g + \mathcal{O}_0 N$ is as follows:

1. multiply $\mathcal{O}_0 = \langle 1, i, \frac{1+k}{2}, \frac{i+j}{2}\rangle$ by $\gamma\beta$ and encode it to a matrix with

$$\left(1\cdot g, i\cdot g, \frac{1+k}{2}\cdot g, \frac{i+j}{2}\cdot g\right).$$

   The maximum size of integers to represent each coefficient is less than pN.

2. By the same procedure, multiply $\mathcal{O}_0$ by N. The maximum size of integers to represent each coefficient is less than N.

3. Concatenate two vectors and then apply Algorithm 1, ensuring the output matrix is a basis matrix of $I = \mathcal{O}_0 g + \mathcal{O}_0 N$. In this procedure, the maximum size of integers is less than $16p^2N^2$ by Lemma 10.

□

B Experiments

In this section, we report two experiments supporting the heuristic estimates used in the main body of the paper.

The first experiment tests the basis-norm assumption used in the analysis of compact ideal multiplication: for ideals produced by the relevant SQIsign subroutines, we measure the largest reduced norm of their basis elements relative to the reduced norm of the ideal.

The second experiment studies the trace-dual lattice of ideals sampled by RandomIdealGivenPrimeNorm; by estimating the shortest vector in the dual lattice and applying transference, it provides experimental evidence for the $\Theta(\sqrt{p})$ scale used in the analysis of RandomEquivalentPrimeIdeal.

The source code used for these experiments is publicly available at:

https://github.com/munsanwon2/compact-SQIsign-Experiment

B.1 The reduced norm of basis elements before ideal multiplication

Goal. In order to derive a clean upper bound for ideal multiplication, one first needs to control the reduced norms of the basis elements of the ideals that appear *before* the multiplication step. The purpose of this experiment is therefore to estimate, for each relevant ideal-generation procedure, how large the reduced norm of a basis element can be relative to the reduced norm of the ideal itself.

Experimental setting. For each sampled left $\mathcal{O}_0$-ideal $I = \langle \alpha_1, \alpha_2, \alpha_3, \alpha_4 \rangle$, we computed

$$R(I) := \frac{\max_{1 \le i \le 4} \mathrm{nrd}(\alpha_i)}{\mathrm{nrd}(I)},$$

where $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ denotes the natural lattice basis used in the implementation, and we checked whether $R(I)$ exceeds the prescribed threshold $\frac{64p^2}{\pi^2}\mathrm{nrd}(I)$ used in the multiplication analysis. The experiment was carried out for all concrete SQIsign parameter sets of each security parameter $\lambda \in \{128, 192, 256\}$, with 10,000 independent trials for each algorithm and for each parameter set. In accordance with the actual protocol path immediately preceding ideal multiplication, we excluded the standalone call Algorithm 4, since its output is not used directly as an input to the multiplication step, but we retained Algorithm 5 applied after this call. Thus, the tested procedures were as follows:

- RandomEquivalentPrimeIdeal∘RandomIdealGivenPrimeNorm
- RandomIdealGivenNorm
- KernelDecomposedToIdeal

Result. Across all 10,000 trials for every included algorithm and every parameter set, we did not observe a single sampled ideal whose basis contained an element with reduced norm exceeding the chosen threshold $\frac{64p^2}{\pi^4}\mathrm{nrd}(I)$. Hence, this provides strong heuristic evidence that the ideals entering the multiplication step satisfy the basis-norm bound required for the derivation of the ideal-multiplication estimate.

C Bound on IdealToIsogeny

We now revisit the integer-size bound for the algorithm IdealToIsogeny. The dominant arithmetic in this procedure comes from the ideal operations used to prepare suitable short representatives. In the previous analysis, this contribution was bounded using a coarse worst-case estimate for ideal multiplication. This was sufficient for the correctness and asymptotic structure of the algorithm, since the remaining steps were already bounded by the same dominant term. However, after refining the analysis of ideal multiplication, the same improvement can be propagated through the subroutines used by IdealToIsogeny.

C.1 Bound on SuitableIdeals

The first step is to update the bound for the subroutine SuitableIdeals. This algorithm is shown as Algorithm 10. The search phase itself does not introduce the largest integers: as in the previous analysis, the maximum size occurs during the ideal multiplication required to compute the reduced basis. Therefore, replacing the old ideal multiplication estimate by Lemma 11 immediately yields a tighter bound for SuitableIdeals.

The following lemma records the resulting refined bound. Compared with the previous accounting, the algorithm is unchanged; only the size analysis is updated. The improvement comes from using the sharper bound for the intermediate ideal product appearing in the computation of the Minkowski reduced basis.

Lemma 21. *During the execution of Algorithm 10, the maximum size of integers is less than*

$$\frac{1024p^2}{\pi^4}\log p \cdot \mathrm{nrd}(I).$$

Proof. For each $t \in \{0, \ldots, n_{\mathrm{orders}}\}$, set $K_t = \overline{J_t} I$. In Lines 2–6, the algorithm computes a basis $(b_1, \ldots, b_4)$ of K_t, forms the Gram matrix $G = (\langle b_r, b_s \rangle)_{1 \le r,s \le 4}$, and applies the $\mathsf{L2}_{\delta,\eta}$ reduction algorithm to this basis and Gram matrix.
Assuming a similar heuristic in Appendix B.1, each basis element of J_t and I has reduced norm less than $\frac{64p^2}{\pi^4}$ times the reduced norm of its own ideal. Hence, the maximum size of integers does not occur in lines 2–3 because of the ideal multiplication in line 4.
By the same argument as in the proof of [7, Lemma 6], the maximum size of the integers occurring after the construction of K_t; namely, in the Gram-matrix computation, the $\mathsf{L2}_{\delta,\eta}$ reduction, the enumeration of L_t, the norm evaluations, the sorting step, and the subsequent modular arithmetic, is bounded by the maximum size already reached during the ideal multiplication computing $K_t = \overline{J_t} I$ in Line 4. Hence it suffices to bound the integers arising in this ideal multiplication.

Algorithm 10 SuitableIdeals(I)

Input: A left $\mathcal{O}_0$-ideal I.
Output: Positive integers u, v, e, indices s, t, and $\beta_1 \in \overline{J_s}I$, $\beta_2 \in \overline{J_t}I$ such that $ud_1 + vd_2 = 2^e$, $\gcd(ud_1, vd_2) = 1$, and $e \leq f$, where $d_1 = \mathrm{nrd}(\beta_1)/\mathrm{nrd}(\overline{J_s}I)$, $d_2 = \mathrm{nrd}(\beta_2)/\mathrm{nrd}(\overline{J_t}I)$

for t from 0 up to n_{orders} **do**
  $J_t \leftarrow \mathsf{LLL}(J_t)$
  $I \leftarrow \mathsf{LLL}(I)$
  $(b_1, \ldots, b_4) \leftarrow \mathsf{CompactIdealMultiplication}(\overline{J_t}, I)$
  Compute the Gram matrix $G = \big(\langle b_r, b_s \rangle\big)_{1 \leq r,s \leq 4}$
  $(\alpha_1, \ldots, \alpha_4), _ \leftarrow \mathsf{L2}_{\delta,\eta}\big((b_1, \ldots, b_4), G\big)$.
  $L_t \leftarrow \left[\sum_{s=1}^{4} x_s \alpha_s \;\middle|\; (x_1, \ldots, x_4) \in [-m, \ldots, m]^4\right]$ for $m = 2 + \lfloor (\log_2(p) - \nu_2(p+1))/4 \rfloor$
  Sort L_t by increasing norm
end for
for $(s, t) \in [0, \ldots, n_{\text{orders}}]^2$ **do** ▷ by increasing lexicographic order
  for $(\beta_1, \beta_2) \in L_s \times L_t$ **do** ▷ by increasing lexicographic order
    $d_1 \leftarrow \mathrm{nrd}(\beta_1)/\mathrm{nrd}(\overline{J_s}I)$ and $d_2 \leftarrow \mathrm{nrd}(\beta_2)/\mathrm{nrd}(\overline{J_t}I)$.
    if $d_1 \equiv 1 \pmod 2$ and $d_2 \equiv 1 \pmod 2$ and $\gcd(d_1, d_2) = 1$ **then**
      $u \leftarrow 2^f d_1^{-1} \pmod{d_2}$.
      $v \leftarrow (2^f - ud_1)/d_2$.
      if $v > 0$ **then**
        $e \leftarrow \mathrm{DyadicValuation}(u)$
        $u \leftarrow u/2^e$, $v \leftarrow v/2^e$, and $e \leftarrow f - e$.
        return $u, v, e, s, t, \beta_1, \beta_2$.
      end if
    end if
  end for
end for
raise Exception(“`SuitableIdeals failed`”).

By Lemma 11, Lemma 4, and the fact that the SQIsign precomputed ideals satisfy $\mathrm{nrd}(\overline{J_t}) = \mathrm{nrd}(J_t) < (\log p)^2$ (see [7, Appendix E]), the integers occurring in the computation of $\overline{J_t}I$ are, uniformly in t, less than

$$\frac{1024p^2}{\pi^4} \log p \cdot \mathrm{nrd}(I).$$

Therefore the same bound holds for every integer occurring during the execution of Algorithm 10. □

We next propagate this refined estimate to Algorithm 11.

The improved bound for ideal multiplication, once inserted into SuitableIdeals, also gives the refined bound for the full IdealToIsogeny procedure.

Algorithm 11 IdealToIsogeny(I)

Input: A left $\mathcal{O}_0$-ideal I.
Output: The codomain E_I of φ_I, $\varphi_I(P_0)$, and $\varphi_I(Q_0)$.

$u, v, e, s, t, \beta_1, \beta_2 \leftarrow$ SuitableIdeals(I)
$d_1 \leftarrow \mathrm{nrd}(\beta_1)/\mathrm{nrd}(\overline{J_s}I)$
$d_2 \leftarrow \mathrm{nrd}(\beta_2)/\mathrm{nrd}(\overline{J_t}I)$ // $ud_1 + vd_2 = 2^e$, with $\gcd(ud_1, vd_2) = 1$
$E_u, \varphi_u(P_s), \varphi_u(Q_s) \leftarrow$ FixedDegreeIsogeny(s, u) // $\varphi_u : E_s \to E_u$ is an isogeny of degree u
$E_v, \varphi_v(P_t), \varphi_v(Q_t) \leftarrow$ FixedDegreeIsogeny(t, v) // $\varphi_v : E_t \to E_v$ is an isogeny of degree v
$[P, Q]^T \leftarrow \left[\frac{1}{\mathrm{nrd}(I)\mathrm{nrd}(J_t)}\right] \mathbf{M}_{\beta_1}\mathbf{M}_{\beta_2}[\varphi_v(P_t), \varphi_v(Q_t)]^T$
$K_P \leftarrow [2^{f-e}]([d_1]\varphi_u(P_s), P)$
$K_Q \leftarrow [2^{f-e}]([d_1]\varphi_u(Q_s), Q)$
$E \times E', [(P, P'), (Q, Q')] \leftarrow$ Isogeny22Chain$(K_P, K_Q, [(\varphi_u(P_s), 0_{E_v}), (\varphi_u(Q_s), 0_{E_v})])$
if $e_{2f}(P, Q) = e_{2f}(P_s, Q_s)^{u^2 d_1}$ **then** // We identify the right curve E_I by ensuring it is d_1-isogeneous to E_0
    $E_I \leftarrow E$, $P_I \leftarrow P$, and $Q_I \leftarrow Q$
else
    $E_I \leftarrow E'$, $P_I \leftarrow P'$, and $Q_I \leftarrow Q'$
    $[P_I, Q_I]^T \leftarrow \left[\frac{1}{ud_1}\right] \mathbf{M}_{\beta_1}[P_I, Q_I]^T$
end if
return $E_I, (P_I, Q_I)$

Lemma 22. *During the execution of Algorithm 11, the maximum size of integers is less than*

$$\frac{1024p^2}{\pi^4} \log p \cdot \mathrm{nrd}(I).$$

Proof. By following the same process of the proof of [7, Lemma 7], we have the same bound with Algorithm 10. □

C.2 Bound on Qlapoti

Qlapoti [24] is an algorithm for finding two ideals I_1, I_2 which are equivalent to an input ideal I and satisfying $\mathrm{nrd}(I_1) + \mathrm{nrd}(I_2) = 2^e$ for the fixed positive integer e. This algorithm is shown in Algorithm 12.

As in [7, Appendix D], the analysis on the algorithm SuitableIdeals can be analogously applied to the bound analysis on the algorithm Qlapoti. By the same argument, we derive the bound on Qlapoti which is same with the bound on SuitableIdeals.

Lemma 23. *During the execution of Algorithm 12, the maximum size of integers is less than*

$$\frac{1024p^2}{\pi^4} \log p \cdot \mathrm{nrd}(J).$$

Algorithm 12 Qlapoti(J)

Input: A left $\mathcal{O}_0$-ideal J.
Output: Equivalent ideals I_1 and I_2 with $\mathrm{nrd}(I_1) + \mathrm{nrd}(I_2) = 2^e$.

1: $I \leftarrow \mathsf{SmallestEquivIdeal}(J)$
2: $n \leftarrow \mathrm{nrd}(I)$
3: $\alpha \leftarrow \mathsf{SmallGenerator}(I)$ ▷ write $\alpha = a_\alpha + b_\alpha i + c_\alpha j + d_\alpha k$ for $a_\alpha, b_\alpha, c_\alpha, d_\alpha \in \frac{1}{2}\mathbb{Z}$
4: $r \leftarrow \mathrm{nrd}(\alpha)/n$
5: **if** $\gcd(2a_\alpha, n) \neq 1$ **and** $\gcd(2b_\alpha, n) \neq 1$ **then**
6: **goto** line 3
7: **end if**
8: Compute the shortest (s, t) such that $2a_\alpha s + 2b_\alpha t \equiv 2^e - 2r \pmod{n}$
9: Set $z := \dfrac{2\left(2^e - 2r - 2a_\alpha s - 2b_\alpha t\right)}{n} - s^2 - t^2$
10: **if** $z < 0$ **then**
11: **goto** line 3
12: **end if**
13: **if** $z \equiv 0 \pmod 4$ **and not** $s = t = 0 \pmod 2$ **then**
14: **goto** line 3
15: **end if**
16: **if** $z \equiv 1 \pmod 4$ **and not** $s \neq t \pmod 2$ **then**
17: **goto** line 3
18: **end if**
19: **if** $z \equiv 2 \pmod 4$ **and not** $s = t = 1 \pmod 2$ **then**
20: **goto** line 3
21: **end if**
22: **if** $z \equiv 3 \pmod 4$ **then**
23: **goto** line 3
24: **end if**
25: $\mathrm{sol} \leftarrow \mathsf{Cornacchia}(1, z)$ ▷ May return $\perp$ if no solution is found.
26: **if** $\mathrm{sol} = \perp$ **then**
27: **goto** line 3
28: **end if**
29: $z_0, z_1 \leftarrow \mathrm{sol}$ ▷ $z_0^2 + z_1^2 = z$
30: **if** $z_0 \not\equiv s \pmod 2$ **then**
31: swap z_0, z_1
32: **end if**
33: $a_1 \leftarrow (z_0 + s)/2, \quad b_1 \leftarrow (z_1 + t)/2$
34: $a_2 \leftarrow s - a_1, \quad b_2 \leftarrow t - b_1$
35: $\beta_1 \leftarrow n(a_1 + b_1 i) + \alpha, \quad \beta_2 \leftarrow n(a_2 + b_2 i) + \alpha$
36: $I_1 \leftarrow \chi_I(\beta_1), \quad I_2 \leftarrow \chi_I(\beta_2)$
37: **return** I_1, I_2

Proof. It is trivial that the maximum size of integers occurs in one of these lines: line 1, line 3, line 8, line 9, lines 33–36.

(1) Lines 1–3. By applying the analysis of Qlapoti in previous work [7, Appendix D] with our compact quaternion algorithms in Section 3 and Section 4,

the maximum size of integers in line 1 is less than

$$\frac{1024p^2}{\pi^4}\log p \cdot \mathrm{nrd}(I),$$

and the maximum size of integers in line 3 is less than

$$\max\left(\frac{8p}{\pi^2}\mathrm{nrd}(I), \mathrm{nrd}(I^2)\right) = \frac{16\sqrt{2}\,p^{1.5}}{\pi} \quad ,$$

by [22, Lemma 12].

(2) Lines 8–9. The maximum size of integers in line 8 is less than $\mathrm{nrd}(J)^2$ since each integer operation is performed in $\mathbb{Z}_{\mathrm{nrd}(J)}$. Hence the maximum size of integers in line 9 is less than $\mathrm{nrd}(J)$ or $2^{e+1} \leq 2(p+1)$.

(3) Lines 33–36. Since $z_0^2 + z_1^2 = z$, $z_0 + s < z + s < 2^e \leq p + 1$ and $z_1 + t < z + t < 2^e \leq p + 1$. So the maximum size of integers in lines 33–34 are less than $p+1$. Hence, the maximum size of integers in line 35 is less than $n \cdot p = p\,\mathrm{nrd}(I)$. □