Power Analysis on SQIsign via Ideal Reconstruction from Quaternion Leakages

Won Kim[1], Jaeho Jeon[2], JeongHwan Lee[1], Suseong Lee[2], Young-Sik Kim[2] and Changmin Lee[1*]

[1] School of Cybersecurity, Korea University, Seoul, Korea
[2] Department of Electrical Engineering and Computer Science, DGIST, Daegu, Korea

Abstract. SQIsign is an isogeny-based signature scheme, which is one of the Round-3 candidates for NIST's additional post-quantum digital signature standardization. NIST considers resistance to side-channel attacks as one important factor in evaluating digital signature schemes. However, side-channel analysis of SQIsign remains very limited. To the best of our knowledge, all existing studies target the Round-1 version, so their applicability to the substantially revised Round-3 SQIsign has not been established. Moreover, existing studies either provide experimental validation only for toy parameters or assume exploitable leakage without demonstrating its extraction in practice, leaving their applicability to realistic implementations unclear.
In this work, we identify five side-channel vulnerabilities in the signing procedure of Round-3 SQIsign and provide polynomial-time algebraic algorithms that recover the secret key from the corresponding leaked intermediate values. Additionally, we further provide algorithms that enable the reconstruction of the complete secret key even when the side-channel leakage reveals only partial bits of an algebraic value, for each vulnerability. Finally, we further perform power analysis attack on the SQIsign source code, extracting the necessary leakage or its partial information for each vulnerability.

Keywords: SQIsign · Quaternion algebras · Side-channel analysis · Isogenies · Power analysis · Ideal reconstruction

1 Introduction

SQIsign [DFKL+20, AAA+26] is an isogeny-based post-quantum digital signature scheme. It has been selected as one of the eight candidates remaining in the third round of NIST's Additional Digital Signature Schemes standardization process [ABC+26]. [1] Among the remaining candidates and standard digital signature schemes, SQIsign provides exceptionally compact public key and signature sizes. A detailed comparison is provided in Table 1.[2]

Resistance to side-channel attacks, including the cost of the corresponding countermeasures, is an explicit consideration in NIST's Additional Digital Signature standardization process. This reflects the fact that mathematical security alone does not guarantee implementation security, since physical observations such as timing, power consumption, or

*Corresponding Author
[1]The HAWK team withdrew its scheme from the NIST's Additional Digital Signature Schemes standardization process, because of the recent attack [SW26, NLLM26].
[2]This table is the summarization of the benchmark from the NIST Post-Quantum Signature Zoo (https://pqshield.github.io/nist-sigs-zoo).
[3]This is the new parameter proposed at [BCC+25a] which is secure for the recent attack [Ran26].

Table 1: Public key (PK), signature (Sig), and combined PK and signature sizes in bytes at NIST security levels I, III, and V.

Scheme	PK (bytes)			Sig (bytes)			PK+Sig (bytes)		
	I	III	V	I	III	V	I	III	V
SQIsign	**83**	**129**	**169**	**200**	**306**	**406**	**283**	**435**	**575**
ML-DSA [oST24]	1312	1952	2592	2420	3309	4627	3732	5261	7219
SLH-DSA [Nat24]	32	48	64	7856	16224	29792	7888	16272	29856
Falcon [PFH+20]	897	–	1793	666	–	1280	1563	–	3073
FAEST [BBB+25]	32	48	≥48	≥3906	≥9340	≥17984	≥3938	≥9388	≥18048
MAYO [BCC+25b]	≥1456	2986	5554	≥239[3]	681	964	≥1920	3667	6518
MQOM [BBFR25]	≥52	≥78	≥104	≥2820	≥6280	≥11564	≥2872	≥6358	≥11668
QR-UOV [FIH+25]	≥12266	≥34399	≥58532	≥157	≥232	≥306	≥12701	≥35042	≥59339
SDitH [AMBB+25]	70	98	132	≥3705	≥7964	≥14121	≥3775	≥8062	≥14253
SNOVA [WCD+25]	≥379	≥800	≥3213	≥146	≥210	≥276	≥525	≥1010	≥3489
UOV [BCD+25]	≥43576	≥189232	≥446992	≥96	200	260	≥43672	≥189432	≥447252

electromagnetic emanations may reveal secret-dependent information. Systematically identifying and characterizing such vulnerabilities is therefore essential for assessing real-world security and designing effective countermeasures with acceptable overhead.

However, existing side-channel and fault-attack studies on SQIsign have focused on its Round-1 design. Since then, the signing procedure of SQIsign has been substantially redesigned, leaving the applicability of previous attacks to Round-3 uncertain. Moreover, existing countermeasures address only selected subroutines, rather than the complete signing procedure. Consequently, the side-channel attack surface of Round-3 SQIsign, as well as the effectiveness and cost of comprehensive protection, remains insufficiently characterized, posing potential risks to real-world deployment.

1.1 Our contributions

We study how side-channel leakage of intermediate values from quaternion computations in Round-3 SQIsign [AAA+26] can be converted into a valid signing key. This achieves the first power analysis attack on the signing algorithm of Round-3 SQIsign. Our contributions are as follows.

- **Five vulnerabilities and key-recovery attacks (Section 3).** We identify five leakage targets in the signing procedure and give an algebraic recovery strategy for each. To do so, we recover the secret ideal from each leakage:
  - (i) *The secret ideal used in ideal intersection (Section 3.1).* We recover the secret ideal directly from its leaked inert representation, consisting of its norm and two integer coefficients, and check its validity with the public key.
  - (ii) *The intersection with the challenge ideal (Section 3.2.1).* We show that leaking the intersection is sufficient even when the stored secret ideal is not directly exposed. Applying 2-saturation removes the challenge ideal's 2-primary part of the intersection ideal, and thus recovers the secret ideal with odd prime norm.
  - (iii) *The equivalent prime-norm ideal (Section 3.2.2).* For the ideal returned by EquivalentCoprimeIdeal, we compute its associated isogeny and compose it with the dual of the public challenge isogeny from the same signing execution. Then, converting the resulting isogeny into its corresponding ideal yields a signing-equivalent secret ideal.
  - (iv) *The conjugated quaternion witness (Section 3.3.1).* Given the quaternion element after conjugation in SampleQuaternionResponse from side-channel leakage and the

output-ideal norm, we undo the conjugation to obtain an element of the secret ideal. Its norm relation determines the norm of the secret ideal, and the primitive element then generates the secret ideal together with that norm.

(v) *The random response quaternion (Section 3.3.2).* We reconstruct the equivalent prime-norm ideal from the conjugate of the output of RandomBoundedElement and the appropriate prime divisor of its norm. We then apply the preceding prime-ideal recovery method given in Section 3.2.2.

These attacks share a final reconstruction step (Section 3.1.2): for an accepted secret ideal, we reconstruct the remaining torsion-basis change information and obtain a complete signing key for the given public key.

- **Algebraic completion from partial MSB leakage (Section 4).** We develop completion algorithms for leaky values when only the correct most significant bits (MSBs) of their integer components are leaked. We prove that the correct value is retained under sufficiently many correct leakage bits and express the time and space complexity.
- **Experimental evaluation of power-trace leakage (Section 5).** We evaluate whether the intermediate values required by these attacks can be extracted from measured power traces of SQIsign signing implementation. For each evaluated target, we report the observed leakage, recovery accuracy, and implications for the corresponding algebraic attacks. This evaluation connect physical leakage extraction to partial-MSB completion and signing-key reconstruction.

1.2 Related works

Although SQIsign has emerged as a prominent post-quantum signature scheme, research on its physical security remains extremely limited. To the best of our knowledge, only three prior studies have investigated side-channel or fault attacks against SQIsign. We review these works and their limitations with respect to the current SQIsign specification.

Jacquemin et al. [JMKR23] presented the first side-channel analysis of SQIsign, targeting Cornacchia's algorithm used within the quaternion arithmetic of the signing procedure to represent an integer as a sum of two squares. They observed that its half-GCD subroutine relies on a non-constant-time Euclidean algorithm whose number of division steps depends on the input integer m. They further showed that, if the input m to Cornacchia's algorithm is precisely recovered, the SQIsign signing key can be reconstructed in polynomial time, and proposed two constant-time variants of the algorithm. Their key-recovery analysis, however, assumes precise knowledge of m rather than demonstrating its recovery from physical leakage.

Mukherjee et al. [MCJ+25] presented a simple power analysis (SPA) attack against SQIsign that targets Cornacchia's algorithm during the signing procedure. In particular, their attack exploits power leakage from the modular exponentiation performed during the modular square-root computation and recovers the secret-dependent input integer m. They then reconstruct the signing key by backtracking through the hierarchy of quaternion subroutines used to construct the response ideal. Their physical experiments were demonstrated only for toy parameters with a 54-bit modulus because of memory limitations on the target microcontroller.

Lee et al. [LHK+24] presented fault attacks against SQIsign targeting both the commitment ideal generation and the subsequent isogeny computations. They showed that faults injected during commitment ideal generation can produce exploitable malformed ideals, and that the iterative structure of isogeny computation enables key-recovery attacks through loop-abort faults. Their evaluation was based on probabilistic simulations under prescribed fault models rather than physical fault-injection experiments. Moreover, their

attacks require either terminating an isogeny computation at a designated iteration or inducing two precisely positioned faults within a single signature generation.

These prior attacks no longer directly apply to the current SQIsign specification. In particular, the algebraic backtracking technique of Mukherjee et al. [MCJ+25] does not directly extend to the current signing procedure, where intermediate values in the two-dimensional (2D) isogeny computation are rerandomized. The attacks of Jacquemin et al. [JMKR23] and Lee et al. [LHK+24] were analyzed against earlier SQIsign specifications, and their applicability to the current specification remains to be evaluated.

2 Preliminaries

In this section, we recall some preliminaries necessary to understand the paper. We give notation used throughout the paper as shown in Table 2.

Table 2: Notation used throughout the paper.

Notation	Meaning
Scheme parameters and isogenies	
λ	Security parameter.
p	A prime of the form $p = c \cdot 2^f - 1$, where c is a small positive integer, and p is a prime satisfying $p \equiv 3 \bmod 4$ with $p \approx 2^{3\lambda}$.
pk	Public key $\mathsf{pk} = \left(E_{\mathsf{pk}}, \mathsf{hint}_{\mathsf{pk}}\right)$.
sk	Secret signing key $\mathsf{sk} = \left(E_{\mathsf{pk}}, \mathsf{hint}_{\mathsf{pk}}, I_{\mathsf{sk}}, M_{\mathsf{sk}}\right)$.
e_{rsp}	Response exponent $\left\lceil \frac{\log_2 p}{2} \right\rceil + \frac{\lambda}{4} + 1$.
HASH, $\|$	Hash function, and concatenation of bit strings, respectively.
msg	Message to be signed.
Elliptic curves and isogenies	
E_0	Fixed starting curve $E_0 : y^2 = x^3 + x$ over $\mathbb{F}_{p^2}$, whose endomorphism ring is identified with $\mathcal{O}_0$.
E_{pk}	Public-key curve and codomain of the secret isogeny.
E_{com}	Commitment curve produced during the commitment phase.
E_{chl}	Challenge curve, which is the common codomain of the challenge and response isogenies.
φ_{sk}	Secret isogeny $\varphi_{\mathsf{sk}} : E_0 \to E_{\mathsf{pk}}$.
$j(E)$	j-invariant of the elliptic curve E.
ζ_n^E	n-Weil pairing on a curve E
Quaternion algebras	
I_{sk}	Integral left $\mathcal{O}_0$-ideal representing the secret ideal class and corresponding to the secret isogeny $\varphi_{\mathsf{sk}} : E_0 \to E_{\mathsf{pk}}$.
$I_{\mathrm{com}}^{(t)}$	Commitment ideal generated in the t-th signing execution and corresponding to the commitment isogeny $\varphi_{\mathsf{com}}^{(t)}$.
$I_{\mathrm{chl}}^{(t)}$	Integral left $\mathcal{O}_0$-ideal of reduced norm 2^λ obtained from the decomposed challenge kernel in the t-th signing execution.
$[J]_* I$	Pushforward of I through J, defined by $[J]_* I := J^{-1}(J \cap I)$.
$\alpha_{\mathsf{rsp}}^{(t)}$	Response quaternion element from the t-th signing execution.
Attack and algorithmic conventions	
$X^{(t)}$	Value X arising from the t-th signing execution.
$\widehat{X}$	Value reconstructed from side-channel leakage or retained as an attack candidate.
$\mathrm{Mat}_{m\times n}(R)$	Set of $m \times n$ matrices with entries in the ring R.
$\perp$	Failure symbol, indicating that an algorithm did not produce a valid candidate.

Continued on the next page

Notation	Meaning
$\mathrm{odd}(n)$	Largest odd divisor of a positive integer n.
$\mathrm{bitlen}(n)$	Binary bit length of a positive integer n: $\mathrm{bitlen}(n) = \lfloor \log_2 n \rfloor + 1$.

2.1 Isogenies and quaternion algebras

In this subsection, we briefly introduce the basic notions about isogenies used in this paper. For more details, see [Sil09, DF17].

Throughout this paper, every elliptic curve is supersingular and defined over $\mathbb{F}_{p^2}$, unless otherwise stated. The normalized Montgomery form of an elliptic curve E is the elliptic curve representation $E_A : y^2 = x^3 + Ax^2 + x, A \in \mathbb{F}_{p^2}$ with the largest $A^2 \neq 4$ in a given order, satisfying $E \cong E_A$.

An *isogeny* $\varphi : E \to E'$ between elliptic curves E and E' is a non-constant morphism, which is a surjective group homomorphism with a finite kernel. We write $\mathrm{Hom}(E, E')$ for the group of isogenies between E and E', and $\mathrm{End}(E) := \mathrm{Hom}(E, E)$ for the endomorphism ring of E. For an isogeny φ, we denote its domain curve by $\mathrm{Dom}(\varphi)$ and its image curve by $\mathrm{Img}(\varphi)$.

The *degree* of an isogeny φ is its degree as a finite morphism, denoted by $\deg(\varphi)$. We say that φ is a d-isogeny when $\deg(\varphi) = d$. If $\psi : E' \to E''$ is another isogeny, then $\deg(\psi \circ \varphi) = \deg(\psi)\deg(\varphi)$. An isogeny φ is *separable* if the induced extension of function fields is separable; for separable isogenies, we have $\#\ker(\varphi) = \deg(\varphi)$. In particular, every isogeny of degree coprime to p is separable.

We write $[m]_E$ for the multiplication-by-m map on E, or simply $[m]$ when E is clear in context, and we set $E[m] := \ker([m]_E)$. If $\gcd(m, p) = 1$, then $E[m] \cong (\mathbb{Z}/m\mathbb{Z})^2$. More generally, for every finite subgroup $G \subseteq E$ there exists a quotient isogeny $\varphi_G : E \to E/G$, unique up to isomorphism of the codomain, with kernel G [Vél71].

For every nonzero isogeny $\varphi : E \to E'$ there exists a unique *dual isogeny* $\widehat{\varphi} : E' \to E$ such that $\widehat{\varphi} \circ \varphi = [\deg(\varphi)]_E$, and $\varphi \circ \widehat{\varphi} = [\deg(\varphi)]_{E'}$. Finally, if $\varphi : E \to E_1$ and $\psi : E \to E_2$ are separable isogenies with $\gcd(\deg(\varphi), \deg(\psi)) = 1$, then $\varphi(\ker\psi) \subseteq E_1$ is a finite subgroup, and the quotient isogeny $[\varphi]_*\psi : E_1 \to E_1/\varphi(\ker\psi)$ is called the *pushforward* of ψ along φ.

Let n be coprime to p. For an elliptic curve E, the n-Weil pairing is a non-degenerate alternating bilinear pairing

$$\zeta_n^E : E[n] \times E[n] \longrightarrow \mu_n,$$

where μ_n denotes the group of n-th roots of unity. In particular, if (P, Q) is a basis of $E[n]$, then $\zeta_n^E(P, Q)$ has order n. For an isogeny $\psi : E \to E'$ with dual $\widehat{\psi}$,

$$\zeta_n^{E'}(\psi(P), R) = \zeta_n^E(P, \widehat{\psi}(R)), \qquad P \in E[n],\ R \in E'[n].$$

Hence, using $\widehat{\psi} \circ \psi = [\deg\psi]$, for $P, Q \in E[n]$,

$$\zeta_n^{E'}(\psi(P), \psi(Q)) = \zeta_n^E(P, Q)^{\deg\psi}.$$

Quaternion algebras. We recall the basic notions about quaternion algebras used for the efficient computations of supersingular isogenies in SQIsign. For more details, see [Voi21, Ler22].

Let $B_{p,\infty}$ denote a quaternion algebra over $\mathbb{Q}$ which is ramified only at p and ∞, whose $\mathbb{Q}$-basis is $\{1, i, j, k := ij\}$ with $i^2 = -1, j^2 = -p, ij = -ji$. Then, every quaternion element in $B_{p,\infty}$ 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

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

A nonzero element $\alpha \in \mathcal{O}$ is *primitive in* $\mathcal{O}$ if $\alpha \notin m\mathcal{O}$ for every integer $m > 1$. Equivalently, its integer coordinates in any $\mathbb{Z}$-basis of $\mathcal{O}$ have greatest common divisor 1. 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 a $\mathbb{Z}$-module $\mathrm{nrd}(I) := \mathbb{Z}\langle \mathrm{nrd}(\alpha) : \alpha \in I\rangle$, or simply its generator if it is principal. Also, its inverse is defined by $I^{-1} := \{x \in B_{p,\infty} : Ix \subseteq O_L(I)\}$. We have $I^{-1} = \frac{\bar{I}}{\mathrm{nrd}(I)}$ 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). 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,

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

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 $B_{p,\infty}$ [Deu41]. 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 [Ibu82]

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

For an integral left $\mathcal{O}_0$-ideal I with $p \nmid \mathrm{nrd}(I)$, define $E_0[I] := \bigcap_{\alpha \in I} \ker\big(\rho_0(\alpha)\big)$. The corresponding isogeny $\varphi_I : E_0 \to E_0/E_0[I]$ is separable and has degree $\mathrm{nrd}(I)$.

Table 3 summarizes the Deuring correspondence between geometric objects on supersingular elliptic curves and algebraic structures in quaternion algebras.

An integral left $\mathcal{O}_0$-ideal I of reduced norm N is *inert* if it admits the $\mathbb{Z}$-basis $\langle N, Ni, x + yi + \frac{i+j}{2}, -y + xi + \frac{k-1}{2}\rangle$ for integers $0 \le x, y < N$ [LS26]. Its *normalized inert representation* is the tuple $(N, (x, y))$.

Now we recall a lemma and a heuristic about inert ideals.

Lemma 1. *There exists a deterministic polynomial-time algorithm* EquivalentInertFromBasis *that, given a $\mathbb{Z}$-basis B_L of an integral left $\mathcal{O}_0$-ideal L, returns an inert ideal L_0 and $\eta \in \mathbb{Z}[i] \setminus \{0\}$ satisfying $L = L_0\eta$. In particular, $L_0 \sim L$.*

Table 3: Deuring correspondence between endomorphism ring of supersingular elliptic curves and quaternion algebra $B_{p,\infty}$.

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
$\deg(\varphi)$	$\mathrm{nrd}(I_\varphi)$
$\varphi, \psi : E \to E'$	$I_\varphi \sim I_\psi$
$\widehat{\varphi} : E' \to E$	$I_{\widehat{\varphi}} = \overline{I_\varphi}$
$\psi \circ \varphi$	$I_\varphi \cdot I_\psi$

Proof. The proof of [LS26, Lemma 7] shows that $L_0 = L\eta^{-1}$ is inert when η has maximal norm among Gaussian integers satisfying $L \subseteq \mathcal{O}_0\eta$. Using the right-module decomposition $\mathcal{O}_0 = \mathbb{Z}[i] \oplus \frac{i+j}{2}\mathbb{Z}[i]$, this factor is computed as the Gaussian GCD of the coordinates of the input basis elements. Dividing the basis on the right by η and extracting the normalized inert representation by integer linear algebra yields the required output. These operations have polynomial bit complexity. □

For a rank-two $\mathbb{Z}[i]$-lattice $L \subseteq B_{p,\infty}$, we use the embedding $B_{p,\infty} = \mathbb{Q}(i) \oplus \mathbb{Q}(i)j \hookrightarrow \mathbb{C}^2$, $a + bj \mapsto (a, \sqrt{p}\, b)$, with Hermitian form $h(a + bj, c + dj) := a\bar{c} + p\, b\bar{d}$. For any $\mathbb{Z}[i]$-basis (β_1, β_2) of L, we define

$$\lambda_1(L) := \min_{0 \neq \alpha \in L} \mathrm{nrd}(\alpha), \qquad \det(L) := \sqrt{\det G_L},$$

where $G_L := (h(\beta_r, \beta_s))_{r,s=1}^2$ is the Hermitian Gram matrix of L.

Heuristic 1 ([AAA+26, Heuristic 9.1.1])**.** *Each of the reduced inert ideal behaves like a uniformly random integral lattice of the same rank and determinant: over $\mathbb{Z}[i]$ for the rank-2 lattices that are inert $\mathcal{O}_0$-ideals. In particular, in the two rank-2 cases, for every $t > 0$,*

$$\Pr\left[\lambda_1(L) \leq 2^{-t} \det L\right] \leq 2^{-t} \quad \textit{over } \mathbb{Z}, \qquad \Pr\left[\lambda_1(L) \leq 2^{-t} \det L\right] \leq 2^{-2t} \quad \textit{over } \mathbb{Z}[i].$$

Quaternion lattices. For the analysis of leaked ideals, we view them 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 its Gram matrix

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

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 $x^T G_\Lambda x$.

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

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

We also define the saturation of a quaternion lattice as follows.

Definition 1 (2-saturation)**.** Let $\Lambda \subseteq \mathcal{O}_0$ be a $\mathbb{Z}$-sublattice. The 2-saturation of Λ in $\mathcal{O}_0$ is

$$\mathrm{Sat}_2(\Lambda; \mathcal{O}_0) := \{x \in \mathcal{O}_0 : 2^r x \in \Lambda \text{ for some } r \in \mathbb{Z}_{\geq 0}\}.$$

Additionally, we recall the definition of primary abelian group.

Definition 2 (ℓ-primary finite abelian group)**.** Let ℓ be a prime. A finite abelian group G is called *ℓ-primary* if every element of G has order a power of ℓ; equivalently, if $|G| = \ell^e$ for some integer $e \geq 0$.

2.2 SQIsign

SQIsign [DFKL+20] is an isogeny-based digital signature scheme over supersingular elliptic curves. Conceptually, it is obtained by applying the Fiat-Shamir transform to a SQIsign identification protocol, instantiated by proving the knowledge of hidden endomorphism information associated with a public supersingular curve, as briefly shown in Figure 1.

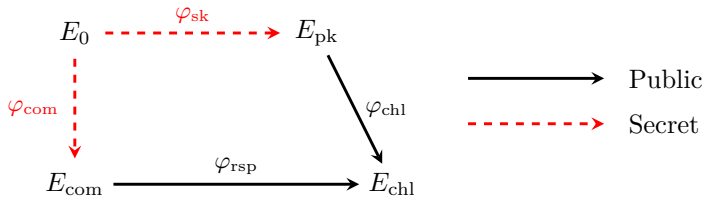


Figure 1: SQIsign identification protocol.

The signing procedure of SQIsign is specified as Algorithm 1. For more details, see [AAA+26].

2.3 Power analysis

The power consumption of a cryptographic device is influenced by the intermediate values processed during an execution. We denote a power trace by

$$\mathbf{t} = (t_1, \ldots, t_m) \in \mathbb{R}^m,$$

where t_j is the power measurement at sample index j. For a sensitive intermediate value z, an individual sample can be modeled as

$$t_j = f_j(z) + n_j,$$

Algorithm 1 SQIsign.Sign(sk, msg, $seed$)

Input: Secret signing key sk, message $msg \in \{0,1\}^*$,
48-byte random seed $seed$.
Output: Signature σ.

$prng_{\text{def}} \leftarrow \mathsf{PRNG.Init}(seed, \texttt{"def"})$
$prng_{\text{com}} \leftarrow \mathsf{PRNG.Init}(seed, \texttt{"com"})$
$prng_{\text{res}} \leftarrow \mathsf{PRNG.Init}(seed, \texttt{"res"})$
Parse sk as $(\mathsf{pk}, I_{\text{sk}}, M_{\text{sk}})$
$I_{\text{com}} \leftarrow \mathsf{UniformO_0ClassSampling}(2, prng_{\text{com}})$
$E_{\text{com}}, \mathcal{B}_{\text{com}} \leftarrow \mathsf{IdealToIsogeny}(I_{\text{com}}, prng_{\text{def}})$
$chl \leftarrow \mathsf{HASH}_\lambda(\mathsf{pk}, E_{\text{com}}, msg)$
$a_{\text{chl}} \leftarrow \mathsf{BytesToInt}(chl, \lambda/8)$
$(c_1, c_2) \leftarrow M_{\text{sk}} \cdot [1, a_{\text{chl}}]^{\mathsf{T}}$
$I_{\text{chl}} \leftarrow \mathsf{KernelDecomposedToIdeal}(\lambda, c_1, c_2)$
$I_{\text{sk,chl}} \leftarrow \mathsf{Intersect}(I_{\text{chl}}, I_{\text{sk}})$
$(I_{\text{sk,chl}}, \alpha_{\text{rsp}}, \alpha_{\text{sk,chl}}, q_{\text{rsp}})$
$\leftarrow \mathsf{SampleQuaternionResponse}(I_{\text{sk,chl}}, I_{\text{com}}, prng_{\text{res}}, prng_{\text{def}})$
$(E_{\text{aux}}, \mathcal{B}_{\text{aux}}, E_{\text{chl}}, \mathcal{B}_{\text{chl}})$
$\leftarrow \mathsf{HDResponse}(I_{\text{sk,chl}}, \alpha_{\text{rsp}}, \alpha_{\text{sk,chl}}, q_{\text{rsp}}, E_{\text{com}}, \mathcal{B}_{\text{com}}, prng_{\text{res}}, prng_{\text{def}})$
$M_{\text{chl}}, hint_{\text{aux}}, hint_{\text{chl}}$
$\leftarrow \mathsf{SetChangeOfBasisMatrix}(E_{\text{aux}}, E_{\text{chl}}, \mathcal{B}_{\text{aux}}, \mathcal{B}_{\text{chl}}, e_{\text{rsp}} + 2)$
$M_{\text{chl}} \leftarrow \mathsf{NormalizeMatrix}(M_{\text{chl}}, e_{\text{rsp}} + 2)$
return $\sigma \leftarrow (chl, E_{\text{aux}}, M_{\text{chl}}, hint_{\text{aux}}, hint_{\text{chl}})$

where f_j denotes an implementation-dependent leakage function and n_j represents measurement noise and background activity, modeled here as unrelated to z.

Common approximations of the leakage function include the Hamming-weight and Hamming-distance models [MOP07]. The Hamming-weight model relates the leakage to $\mathrm{HW}(z)$, the number of set bits in z. The Hamming-distance model instead relates the leakage to the number of bit transitions between two consecutively processed values, $\mathrm{HD}(z, z') = \mathrm{HW}(z \oplus z')$. Although these models are useful for interpreting power leakage, the actual leakage behavior of a device need not follow either model exactly.

In a profiled power analysis attack [CRR02], the adversary first collects a labeled profiling dataset

$$\mathcal{D}_{\text{prof}} = \{(\mathbf{t}_i, y_i)\}_{i=1}^{N},$$

where y_i denotes a sensitive intermediate value, or a selected function thereof, associated with the trace $\mathbf{t}_i$. A classifier trained on this dataset estimates a score or posterior probability $p_\theta(y \mid \mathbf{t})$ for each candidate value y.

During the attack phase, the adversary acquires one or more traces from the target implementation and applies the trained model to each trace. The resulting predictions may be used individually or combined in the subsequent recovery procedure.

Word-level leakage. Multi-precision integers are stored as sequences of fixed-width words, called *limbs*. Leakage may arise when these words are loaded, processed, or stored. In a fixed-width two's-complement representation, sign-extension bits repeat the sign rather than encode additional magnitude bits. Moreover, distinct words can have the same Hamming weight, so recovering a limb's Hamming weight need not determine its leading bits.

Correlation power analysis. Correlation power analysis (CPA) [BCO04] compares predicted leakage with measured samples. For a secret candidate k, let $z_i(k)$ be the intermediate predicted from k and the public input of trace i, and let $t_{i,j}$ be sample j of that trace. Under the Hamming-weight model,

$$r_j(k) = \mathrm{corr}_i\big(\mathrm{HW}(z_i(k)), t_{i,j}\big),$$

where Pearson correlation is the covariance divided by the product of the standard deviations across traces. Large absolute correlations support the candidate's predictions; they are not probabilities of correct recovery.

Leakage detection. First-order test vector leakage assessment (TVLA) [GJJR11] compares pointwise means of fixed-input and random-input trace groups. For group $g \in \{\mathrm{fix}, \mathrm{rnd}\}$, let n_g, $\bar{t}_{g,j}$, and $s^2_{g,j}$ denote its size, sample mean, and sample variance at index j. Welch's statistic is

$$\tau_j = \frac{\bar{t}_{\mathrm{fix},j} - \bar{t}_{\mathrm{rnd},j}}{\sqrt{s^2_{\mathrm{fix},j}/n_{\mathrm{fix}} + s^2_{\mathrm{rnd},j}/n_{\mathrm{rnd}}}}.$$

The conventional threshold $|\tau_j| > 4.5$ flags potential leakage; its statistical interpretation depends on the sample sizes and the number of tested points. Detecting a mean difference does not establish bit recovery, and failing to detect one does not establish absence of leakage.

Leakage strength and points of interest. Let T_j and Y denote the random variables corresponding to a trace sample and its sensitive label. The signal-to-noise ratio (SNR) [MOP07] is

$$\mathrm{SNR}_j(Y) = \frac{\mathrm{Var}(\mathbb{E}[T_j \mid Y])}{\mathbb{E}[\mathrm{Var}(T_j \mid Y)]},$$

comparing variation between label-conditioned means with variation within labels. The within-Hamming-weight value-SNR used below assesses variation among values of equal Hamming weight, beyond the Hamming-weight component. A label-permutation null is obtained by shuffling trace–label associations; the reported z-score expresses the observed statistic's excess over the permutation mean in units of the permutation standard deviation. *Points of interest* (POIs) are sample indices selected for their label-dependent leakage.

Templates and trace aggregation. Gaussian templates fit a mean vector and covariance matrix for the selected samples of each label [CRR02]. Bit-linear stochastic templates [SLP05] instead fit the mean leakage as an affine function of the label bits, together with a noise model. A multilayer perceptron provides a nonlinear classifier for the same prediction task. Multiple-trace attacks combine evidence for a common secret hypothesis, accounting for input-dependent intermediates. A single-trace attack uses one attack trace per unknown value, regardless of profiling-set size; freshly sampled values in other executions are not repeated observations of that value.

Recovery metrics and validation. Classification accuracy is the fraction of correctly predicted labels; the majority-class baseline always predicts the most frequent label. Guessing entropy (GE) is the average rank of the correct candidate after sorting by the attack scores [SMY09]. Throughout this paper, ranks are zero-based: the best rank is 0, and a uniformly random ordering of 256 candidates has expected rank 127.5. A small GE does not imply that the correct candidate is ranked first in every trial. Leave-one-value-out evaluation excludes the tested value from profiling and averages results over the tested values. Validation data guide model selection, whereas held-out test traces are reserved for

evaluation after models, sample locations, and candidates are fixed. Shifted-window and permuted-label controls test whether the observed agreement depends on the intended time locations and labels.

3 Attack scenarios

In this section, we present the vulnerabilities and algebraic key-recovery algorithms against SQIsign, when each specific side-channel leakage is obtained.

3.1 Key recovery from a direct secret ideal

In this subsection, we target the input secret ideal of ideal intersection. For the t-th signing execution, we write I_{chl} with $I_{\mathrm{chl}}^{(t)}$ from Algorithm 1. The signing algorithm invokes the ideal intersect algorithm $\mathsf{Intersect}$ on ideals I_{sk} and $I_{\mathrm{chl}}^{(t)}$, before passing its returned inert representative to $\mathsf{SampleQuaternionResponse}$.

3.1.1 Extracting the secret ideal from leakage

The computation of the lattice intersection $I_{\mathrm{chl}}^{(t)} \cap I_{\mathrm{sk}}$ takes the inert representation $I_{\mathrm{sk}} = (N_{\mathsf{sk}}, (x_{\mathsf{sk}}, y_{\mathsf{sk}}))$ of the secret ideal. Consequently, an implementation without sufficient protection of this representation may directly leak its coefficients.

The inert representation uniquely determines the secret ideal and its canonical $\mathbb{Z}$-basis:

$$B_{\mathsf{sk}} = \left(N_{\mathsf{sk}}, N_{\mathsf{sk}} i, x_{\mathsf{sk}} + y_{\mathsf{sk}} i + \frac{i+j}{2}, -y_{\mathsf{sk}} + x_{\mathsf{sk}} i + \frac{k-1}{2} \right).$$

Hence, from leaked values $\widehat{N}_{\mathsf{sk}}, \widehat{x}_{\mathsf{sk}}, \widehat{y}_{\mathsf{sk}}$, the adversary directly reconstructs

$$\widehat{I} := \left(\widehat{N}_{\mathsf{sk}}, (\widehat{x}_{\mathsf{sk}}, \widehat{y}_{\mathsf{sk}}) \right).$$

Under exact leakage,

$$\widehat{I} = (N_{\mathsf{sk}}, (x_{\mathsf{sk}}, y_{\mathsf{sk}})) = I_{\mathrm{sk}}.$$

Therefore, secret-ideal recovery is straightforward once the adversary can determine whether a leakage value indeed corresponds to the secret ideal. It remains only to verify the consistency of the reconstructed ideal with the public key, which we formalize in Algorithm 2.

For exact leakage, $\widehat{I} = I_{\mathrm{sk}}$, so $\mathsf{IdealToIsogeny}$ applied to $\widehat{I}$ returns the public-key curve E_{pk}.

More generally, let

$$(E_{\widehat{I}}, B_{\widehat{I}}) \leftarrow \mathsf{IdealToIsogeny}\left(\widehat{I}, \mathsf{prng}_{\mathrm{def}}\right),$$

for any random number $\mathsf{prng}_{\mathrm{def}}$. If $E_{\widehat{I}} = E_{\mathsf{pk}}$, then the right order of $\widehat{I}$ is isomorphic to $\mathrm{End}(E_{\mathsf{pk}})$. Hence $\widehat{I}$ can be used as a valid secret ideal for the public key, and Algorithm 3 reconstructs the remaining secret-key data required for signing. Therefore, the candidate is accepted exactly when

$$E_{\widehat{I}} = E_{\mathsf{pk}}.$$

Algorithm 2 FilterSecretIdealCandidate $(E_{\mathsf{pk}}, \widehat{I}, K)$

Input: The public-key curve E_{pk}, a candidate $\widehat{I} = \left(\widehat{N}_{\mathsf{sk}}, (\widehat{x}_{\mathsf{sk}}, \widehat{y}_{\mathsf{sk}})\right)$, and a repetition bound K.

Output: The candidate $\widehat{I}$ if it is a valid secret ideal for the public key; otherwise, $\perp$.

if $\widehat{I}$ is not a well-formed InertIdeal or $\widehat{N}_{\mathsf{sk}}$ is not an odd prime **then**
  return $\perp$
end if
for $u = 1, \ldots, K$ **do**
  Sample a fresh 48-byte seed s_u
  $\mathsf{prng}_u \leftarrow \mathsf{PRNG.Init}(s_u, \texttt{"def"})$
  try
    $(E_u, _) \leftarrow \mathsf{IdealToIsogeny}(\widehat{I}, \mathsf{prng}_u)$
  except
    continue
  if $E_u \neq E_{\mathsf{pk}}$ **then**
    return $\perp$
  end if
  return $\widehat{I}$
end for
return $\perp$

Algorithm 3 RecoverSecretKey$(\mathsf{pk}, \widehat{I})$

Input: A public key $\mathsf{pk} = (E_{\mathsf{pk}}, \mathsf{hint}_{\mathsf{pk}})$ and a recovered secret ideal $\widehat{I}$.

Output: A reconstructed secret key $\widehat{\mathsf{sk}} = \left(\mathsf{pk}, \widehat{I}, \widehat{M}_{\mathsf{sk}}\right)$, or $\perp$.

Sample a fresh 48-byte seed $\widehat{\mathsf{seed}}$.
$\mathsf{prng}_{\mathrm{def}} \leftarrow \mathsf{PRNG.Init}\left(\widehat{\mathsf{seed}}, \texttt{"def"}\right)$
$(E_I, \widehat{B}_I) \leftarrow \mathsf{IdealToIsogeny}\left(\widehat{I}, \mathsf{prng}_{\mathrm{def}}\right)$
if $E_I \neq E_{\mathsf{pk}}$ **then**
  return $\perp$
end if
$\widehat{B}_{\mathsf{pk}} \leftarrow \mathsf{TorsionBasisFromHint}\left(E_{\mathsf{pk}}, \lambda + 2, \mathsf{hint}_{\mathsf{pk}}\right)$
$\widetilde{M}_{\mathsf{sk}} \leftarrow \mathsf{ChangeOfBasis}\left(E_{\mathsf{pk}}, \lambda + 2, \widehat{B}_I, \widehat{B}_{\mathsf{pk}}\right)$
$\widehat{M}_{\mathsf{sk}} \leftarrow \widetilde{M}_{\mathsf{sk}} \bmod 2^{\lambda}$
$\widehat{M}_{\mathsf{sk}} \leftarrow \mathsf{NormalizeMatrix}\left(\widehat{M}_{\mathsf{sk}}, \lambda\right)$
return $\left(\mathsf{pk}, \widehat{I}, \widehat{M}_{\mathsf{sk}}\right)$

3.1.2 Full secret key reconstruction from the secret ideal

We now provide Algorithm 3 which recovers the full secret key sk of SQIsign from its input secret ideal I_{sk} and the public key pk corresponding to sk.

The public key already provides E_{pk} and $\mathsf{hint}_{\mathsf{pk}}$, so only the secret change-of-basis matrix must be reconstructed to reconstruct the full secret key $\widehat{\mathsf{sk}}$. Algorithm 3 reruns the ideal-to-isogeny and change-of-basis steps used in the key generation of SQIsign to reconstruct the change-of-basis matrix in sk, by following the same procedure of the key generation step using the valid secret ideal $\widehat{I}$. So the reconstructed matrix satisfies the relation required by the key generation of SQIsign and therefore yields a valid signing-equivalent secret key.

Lemma 2 (Correctness of RecoverSecretKey)**.** *Suppose that $\widehat{I} = I_{sk}$ where I_{sk} is the secret ideal contained in a valid SQIsign secret key sk, associated with the public key $\mathsf{pk} = (E_{\mathsf{pk}}, \mathsf{hint}_{\mathsf{pk}})$. Then, Algorithm 3 returns a valid SQIsign secret key $\widehat{\mathsf{sk}}$ for pk.*

Proof. Let $(E_I, B_I) = \mathsf{IdealToIsogeny}\left(I_{\mathrm{sk}}, \mathsf{prng}_{\mathrm{def}}\right)$, and let $\varphi_I : E_0 \longrightarrow E_I$ be the isogeny corresponding to I_{sk}. Then $B_I = \varphi_I(B_0)$, where B_0 is the fixed basis of $E_0[2^f]$.
Since $\widehat{I} = I_{\mathrm{sk}}$, the isogeny computed by $\mathsf{IdealToIsogeny}$ corresponds to the same left $\mathcal{O}_0$-ideal used in key generation. Thus, their codomain curves are isomorphic. Since both curves are represented in normalized Montgomery form, we have $E_I = E_{\mathsf{pk}}$. Therefore, the consistency check in Algorithm 3 succeeds.
The ideal I_{sk} has odd reduced norm, so φ_I restricts to an isomorphism on $E_0[2^f]$. Consequently, B_I is a basis of $E_{\mathsf{pk}}[2^f]$. Hence, the subroutine $\mathsf{TorsionBasisFromHint}$ reconstructs the public basis $\widehat{B}_{\mathsf{pk}}$ of $E_{\mathsf{pk}}[2^{\lambda+2}]$ correctly.
By the specification of $\mathsf{ChangeOfBasis}$, the matrix $\widetilde{M}_{\mathsf{sk}}$ returned in Algorithm 3 satisfies

$$\widehat{B}_{\mathsf{pk}} = B_I \cdot 2^{f-\lambda-2}\widetilde{M}_{\mathsf{sk}}.$$

So the last subroutine $\mathsf{NormalizeMatrix}$ of Algorithm 3 is precisely the construction used to derive the secret matrix in the key generation of SQIsign. Therefore, $\left(\mathsf{pk}, I_{\mathrm{sk}}, \widehat{M}_{\mathsf{sk}}\right)$ is a valid SQIsign secret key for pk. □

3.2 Key recovery from sublattices

Even when the stored inert representation of the secret ideal I_{sk} is not directly leaked, a single exact sublattice of I_{sk} or its equivalent ideal suffices to recover I_{sk}. In this subsection, we provide two algebraic algorithms which compute the secret ideal or the signing-equivalent secret ideal from each sublattice, extracted in different vulnerabilities.

3.2.1 Key recovery from an intersection

We now target the leakage from the subroutine computing the intersection $I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}$.

Remark 1 (Intersection representation). The subroutine $\mathsf{Intersect}$ may return either $I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}$ or $(I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)})(1-i)^{-1}$. Since $\mathrm{nrd}(I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}) = 2^{\lambda}\mathrm{nrd}(I_{\mathrm{sk}})$ with odd prime $\mathrm{nrd}(I_{\mathrm{sk}})$ and $\mathrm{nrd}(1-i) = 2$, the two cases are distinguished by the 2-adic valuation of the leaked ideal's norm: λ in the former case and $\lambda - 1$ in the latter. In the latter case, right multiplication by $1-i$ recovers $I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}$. Thus, the input to Algorithm 4 is taken to be the reconstructed intersection $I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}$ in either case.

Algorithm 4 reconstructs the secret ideal from the leaked inert representation of $I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}$.

Algorithm 4 RecoverSecretIdealFromIntersection $(E_{\mathsf{pk}}, \widehat{I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}}, K)$

Input: The public-key curve E_{pk}, an inert ideal $\widehat{I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}}$, and a repetition bound K.
Output: A secret ideal $\widehat{I}$ for E_{pk}, or $\perp$.

$\widehat{I} \leftarrow \mathrm{Sat}_2\left(\widehat{I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}}; \mathcal{O}_0\right)$
if $\widehat{I}$ is not inert **then**
  return $\perp$
end if
$\widehat{I} \leftarrow$ FilterSecretIdealCandidate $\left(E_{\mathsf{pk}}, \widehat{I}, K\right)$
return $\widehat{I}$

Lemma 3 (Correctness of RecoverSecretIdealFromIntersection)**.** *Suppose that*

$$\widehat{I_{sk} \cap I_{chl}^{(t)}} = I_{sk} \cap I_{chl}^{(t)}$$

for an honest signing execution. Then, Algorithm 4 returns the secret ideal I_{sk}.

Proof. We first claim that

$$\mathrm{Sat}_2(I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}; \mathcal{O}_0) = I_{\mathrm{sk}}.$$

First, $I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)} = I_{\mathrm{chl}}^{(t)} \cap I_{\mathrm{sk}} \subseteq I_{\mathrm{sk}}$. Since KernelDecomposedToIdeal is invoked with parameter λ in Algorithm 1, we have $\mathrm{nrd}(I_{\mathrm{chl}}^{(t)}) = 2^\lambda$. Therefore, $[\mathcal{O}_0 : I_{\mathrm{chl}}^{(t)}] = \mathrm{nrd}\left(I_{\mathrm{chl}}^{(t)}\right)^2$ is a power of 2. By the second isomorphism theorem,

$$I_{\mathrm{sk}} / \left(I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}\right) \cong \left(I_{\mathrm{sk}} + I_{\mathrm{chl}}^{(t)}\right) / I_{\mathrm{chl}}^{(t)}$$

which is a subgroup of $\mathcal{O}_0 / I_{\mathrm{chl}}^{(t)}$. Thus, combining $\mathrm{nrd}\left(I_{\mathrm{chl}}^{(t)}\right)$ is a power of 2 and $I_{\mathrm{sk}} / \left(I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}\right) \cong \left(I_{\mathrm{sk}} + I_{\mathrm{chl}}^{(t)}\right) / I_{\mathrm{chl}}^{(t)}$, we have $I_{\mathrm{sk}} / \left(I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}\right)$ is 2-primary. Consequently, for every $\alpha \in I_{\mathrm{sk}}$, there exists $e \geq 0$ such that $2^e \alpha \in I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}$, and hence

$$I_{\mathrm{sk}} \subseteq \mathrm{Sat}_2(I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}; \mathcal{O}_0).$$

Conversely, the secret ideal I_{sk} has odd norm. Thus $[\mathcal{O}_0 : I_{\mathrm{sk}}] = \mathrm{nrd}(I_{\mathrm{sk}})^2$ is odd, which implies that I_{sk} is already 2-saturated: $\mathrm{Sat}_2(I_{\mathrm{sk}}) = I_{\mathrm{sk}}$. Since $I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)} \subseteq I_{\mathrm{sk}}$,

$$\mathrm{Sat}_2(I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}; \mathcal{O}_0) \subseteq \mathrm{Sat}_2(I_{\mathrm{sk}}; \mathcal{O}_0) = I_{\mathrm{sk}}.$$

Combining both inclusions gives

$$\mathrm{Sat}_2(I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}; \mathcal{O}_0) = I_{\mathrm{sk}}.$$

Therefore, we conclude that

$$\widehat{I} = \mathrm{Sat}_2\left(\widehat{I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}}; \mathcal{O}_0\right) = \mathrm{Sat}_2(I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}; \mathcal{O}_0) = I_{\mathrm{sk}}.$$

Hence, by the correctness of Algorithm 2, Algorithm 4 returns I_{sk}. □

Remark 2. The computation of 2-saturation can be performed by the integer coordinate matrix of the leaked ideal using Smith normal form, so it can be computed in the polynomial time.

3.2.2 Key recovery from an equivalent prime ideal

We now target the leakage from the subalgorithm EquivalentCoprimeIdeal, which computes an equivalent ideal with the intersection ideal $I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}$.

Let J_t be the odd-prime-norm inert ideal returned by EquivalentCoprimeIdeal in the t-th successful signing execution, so that $J_t \sim I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}$. We compose the isogeny associated with J_t with the dual of the public challenge isogeny, and construct the corresponding ideal from its kernel. This yields an ideal equivalent to I_{sk}, from which an odd-prime-norm inert representative can be obtained and verified against E_{pk}. Algorithm 5 formalizes these recovery steps.

Algorithm 5 RecoverSecretIdealFromPrimeIdeal $(\mathsf{pk}, c, \widehat{J}, K)$

Input: A public key $\mathsf{pk} = (E_{\mathsf{pk}}, \mathsf{hint}_{\mathsf{pk}})$, a challenge integer $0 \le c < 2^{\lambda}$, a recovered inert ideal $\widehat{J}$, and a verification repetition bound K.
Output: A signing-equivalent secret ideal $\widehat{I}$ for pk, or $\perp$.

1: **if** $\widehat{J}$ is not an inert ideal of odd prime norm coprime to p **then**
2: **return** $\perp$
3: **end if**
4: Initialize a seed for the pseudorandom generator prng
5: Compute the public challenge isogeny $\phi : E_{\mathsf{pk}} \to E_{\mathsf{chl}}$ and its dual from (pk, c)
6: $(E_J, B_J) \leftarrow \mathsf{IdealToIsogeny}(\widehat{J}, \mathsf{prng})$, where $B_J = \varphi_J(B_0)$ for the associated isogeny $\varphi_J : E_0 \to E_J$
7: **if** $E_J \neq E_{\mathsf{chl}}$ **then**
8: **return** $\perp$
9: **end if**
10: $\psi \leftarrow \widehat{\phi} \circ \varphi_J$
11: Compute a $\mathbb{Z}$-basis B_L of the ideal L corresponding to ψ from $\ker \psi = E_0[\widehat{J}] + \left(\varphi_J|_{E_0[2^{\lambda}]}\right)^{-1} (\ker \widehat{\phi})$
12: $(L_0, _) \leftarrow \mathsf{EquivalentInertFromBasis}(B_L)$
13: $B_{\mathrm{red}} \leftarrow \max\{\mathrm{bitlen}(2\mathrm{nrd}(L_0)), \mathrm{bitlen}(\lfloor\sqrt{p}\rfloor)\}$
14: $(\widehat{I}, _) \leftarrow \mathsf{EquivalentCoprimeIdeal}(L_0, 0, B_{\mathrm{red}}, \mathsf{prng})$
15: **if** $\gcd(\mathrm{nrd}(\widehat{I}), 2p) \neq 1$ **then**
16: **return** $\perp$
17: **end if**
18: **return** $\mathsf{FilterSecretIdealCandidate}(E_{\mathsf{pk}}, \widehat{I}, K)$

Lemma 4 (Correctness of Algorithm 5)**.** *Suppose that $\widehat{J}$ is an inert equivalent ideal with $I_{sk} \cap I_{chl}^{(t)}$ of odd prime norm, and that c is the public challenge of the same honest signing execution. Then, Algorithm 5 outputs a left $\mathcal{O}_0$-ideal $\widehat{I}$, satisfying $\widehat{I} \sim I_{sk}$.*

Proof. Since $\widehat{J} \sim I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}$, the associated isogeny φ_J has codomain E_{chl} in normalized form. Thus $\psi = \widehat{\phi} \circ \varphi_J : E_0 \to E_{\mathsf{pk}}$. As $\deg \varphi_J$ is odd, its restriction to $E_0[2^{\lambda}]$ is an isomorphism, which gives the kernel decomposition used in the algorithm. Hence the constructed ideal L corresponds to ψ, and the Deuring correspondence implies $L \sim I_{\mathrm{sk}}$.
By Lemma 1 and the specification of EquivalentCoprimeIdeal, we have

$$\widehat{I} \sim L_0 \sim L \sim I_{\mathrm{sk}}.$$

The prime-norm conversion and the coprimality test ensure that $\widehat{I}$ is an inert ideal of odd prime norm coprime to p. Its associated normalized curve is therefore E_{pk}, so, provided

that the invoked subroutines succeed and the coprimality test passes, the algorithm returns $\widehat{I} \sim I_{\mathrm{sk}}$. □

Remark 3. The input c must correspond to the signing execution that produced the leaked ideal, ensuring $E_J = E_{\mathsf{chl}}$ under exact leakage. The step computing ϕ and $\widehat{\phi}$ uses a chain of λ isogenies of degree 2.

In the step computing B_L, express a generator of $\ker \widehat{\phi}$ in the basis $[2^{f-\lambda}]B_J$ using 2-power discrete logarithms. The same coordinates in $[2^{f-\lambda}]B_0$ determine its preimage under φ_J. Apply KernelDecomposedToIdeal to this preimage to obtain the norm-2^λ ideal C, and compute a basis of $L = \widehat{J} \cap C$ by rank-four integer linear algebra.

3.3 Key recovery from a quaternion element

In this subsection, we provide two different vulnerabilities and algebraic algorithms to recover the secret ideal or a signing-equivalent secret ideal from a single quaternion element. For the t-th signing execution, let J_t be the prime-norm equivalent ideal with $I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}$, returned by the subroutine EquivalentCoprimeIdeal.

3.3.1 Key recovery from a conjugated quaternion element

We now target the conjugation of the subroutine SampleQuaternionResponse. Suppose that the side-channel leakage recovers the conjugated quaternion element $\alpha_t \in J_t$, which is the conjugate of β_t such that $J_t = I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)} \frac{\overline{\beta}_t}{\mathrm{nrd}(I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)})}$. Although $\alpha_t = \overline{\beta_t}$ belongs to the equivalent output ideal J_t, undoing the conjugation in SampleQuaternionResponse recovers the primitive equivalence witness used to construct J_t. The norm relation of J_t then allows the adversary to recover the odd prime norm of the secret ideal. Algorithm 6 formalizes this recovery.

Lemma 5 (Correctness of RecoverSecretIdealFromQuaternion)**.** *Suppose that $\widehat{\alpha} = \alpha_t$ and $\widehat{q} = \mathrm{nrd}(J_t)$ for an honest signing execution. Then the output of Algorithm 6 is exactly I_{sk}.*

Proof. Let $I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}$ denote the input inert ideal to EquivalentCoprimeIdeal in the corresponding signing execution. Since $\mathrm{nrd}(I_{\mathrm{chl}}) = 2^\lambda$ is coprime to $\mathrm{nrd}(I_{\mathrm{sk}})$, we have $\mathrm{nrd}(I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}) = 2^\lambda \mathrm{nrd}(I_{\mathrm{sk}})$, where $\mathrm{nrd}(I_{\mathrm{sk}})$ is odd by the specification of the key generation algorithm. Since the subroutine EquivalentCoprimeIdeal computes the output ideal $I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)} \frac{\overline{\beta_t}}{\mathrm{nrd}(I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)})}$ of norm $\frac{\mathrm{nrd}(\beta_t)}{\mathrm{nrd}(I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)})}$, and quaternion conjugation preserves norm,

$$\mathrm{nrd}(\alpha_t) = \mathrm{nrd}(J_t)\mathrm{nrd}(I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}) = \mathrm{nrd}(J_t)2^\lambda \mathrm{nrd}(I_{\mathrm{sk}}).$$

Hence we have

$$N = \mathrm{odd}\left(\frac{\mathrm{nrd}(\alpha_t)}{\mathrm{nrd}(J_t)}\right) = \mathrm{nrd}(I_{\mathrm{sk}}).$$

Moreover, $\overline{\alpha_t}$ is the primitive quaternion sampled from $I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}$ before the conjugation in SampleQuaternionResponse. Then, we have

$$\overline{\alpha_t} \in I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)} \subseteq I_{\mathrm{sk}}.$$

Moreover, $\overline{\alpha_t} \notin \mathrm{nrd}(I_{\mathrm{sk}})\mathcal{O}_0$, because the witness returned by EquivalentCoprimeIdeal is primitive.
Set $L := \mathcal{O}_0\overline{\alpha_t} + \mathrm{nrd}(I_{\mathrm{sk}})\mathcal{O}_0$. Since $\overline{\alpha_t} \in I_{\mathrm{sk}}$ and $\mathrm{nrd}(I_{\mathrm{sk}})\mathcal{O}_0 \subseteq I_{\mathrm{sk}}$, we have $L \subseteq I_{\mathrm{sk}}$. Since $B_{p,\infty}$ is ramified only at p and ∞, it splits at the prime $\mathrm{nrd}(I_{\mathrm{sk}}) \neq p$. As $\mathcal{O}_0$ is maximal,

Algorithm 6 RecoverSecretIdealFromQuaternion $(E_{\mathsf{pk}}, \widehat{\alpha}, \widehat{q}, K)$

Input: The public key curve E_{pk}, a recovered quaternion $\widehat{\alpha}$, a recovered output-ideal norm $\widehat{q}$, and a verification repetition bound K.
Output: A verified secret ideal $\widehat{I}$ for E_{pk}, or $\perp$.

if $\widehat{\alpha} \notin \mathcal{O}_0 \setminus \{0\}$ **then**
  return $\perp$
end if
$n \leftarrow \mathrm{nrd}(\widehat{\alpha})$
if $\widehat{q}$ is not prime or $\widehat{q} \nmid n$ **then**
  return $\perp$
end if
$N \leftarrow \mathrm{odd}\,(n/\widehat{q})$
if N is not prime or $\gcd(N, 2p) \neq 1$ **then**
  return $\perp$
end if
if $\overline{\widehat{\alpha}} \in N\mathcal{O}_0$ **then**
  return $\perp$
end if
$\widehat{I} \leftarrow \mathsf{InertGenToIdealOdd}(N, \overline{\widehat{\alpha}})$
if $\widehat{I} = \perp$ **then**
  return $\perp$
end if
$\widehat{I} \leftarrow \mathsf{FilterSecretIdealCandidate}(E_{\mathsf{pk}}, \widehat{I}, K)$
if $\widehat{I} \neq \perp$ **then**
  return $\widehat{I}$
end if
return $\perp$

reduction of its local matrix representation modulo $\mathrm{nrd}(I_{\mathrm{sk}})$ yields $\mathcal{O}_0/\mathrm{nrd}(I_{\mathrm{sk}})\mathcal{O}_0 \cong \mathrm{Mat}_2(\mathbb{F}_{\mathrm{nrd}(I_{\mathrm{sk}})})$, under which the reduced norm corresponds to the determinant.
Since image of $\overline{\alpha_t}$ modulo $\mathrm{nrd}(I_{\mathrm{sk}})\mathcal{O}_0$ is nonzero and has its determinant $\mathrm{nrd}(\alpha_t) \equiv 0 \pmod{\mathrm{nrd}(I_{\mathrm{sk}})}$, it has rank one. Hence the left ideal generated by its image has rank 2 over $\mathbb{F}_{\mathrm{nrd}(I_{\mathrm{sk}})}$, and therefore $[\mathcal{O}_0 : L] = \mathrm{nrd}(I_{\mathrm{sk}})^2$. Since $[\mathcal{O}_0 : I_{\mathrm{sk}}] = \mathrm{nrd}(I_{\mathrm{sk}})^2$, the inclusion $L \subseteq I_{\mathrm{sk}}$ implies $L = I_{\mathrm{sk}}$.
Thus, the subroutine InertGenToIdealOdd reconstructs I_{sk}. This candidate passes FilterSecretIdealCandidate, since its right order is isomorphic to $\mathrm{End}(E_{\mathsf{pk}})$. Hence Algorithm 6 returns a valid secret ideal for E_{pk}. □

Remark 4 (Quantum recovery without the ideal norm). If a quantum computer is available, a separate recovery of $\mathrm{nrd}(J_t)$ is unnecessary. Given an exact quaternion leakage α_t, the adversary computes $\mathrm{nrd}(\alpha_t)$ and factors $\mathrm{nrd}(\alpha_t)$ using Shor's algorithm. Since $\mathrm{nrd}(J_t)$ is prime and divides $\mathrm{nrd}(\alpha_t)$, it appears among the prime factors of $\mathrm{nrd}(\alpha_t)$. The adversary therefore invokes Algorithm 6 once for each prime factor $q \mid \mathrm{nrd}(\alpha_t)$, setting $\widehat{q} = q$. For $q = \mathrm{nrd}(J_t)$, the hypotheses of Lemma 5 hold, so Algorithm 6 outputs a valid recovered secret ideal.

3.3.2 Key recovery from a response quaternion

We now target the quaternion output of the subroutine RandomBoundedElement. For the t-th successful signing execution, let $\alpha_{\mathrm{rsp}}^{(t)}$ be an output random response quaternion element of RandomBoundedElement. As described in Section 3.2.2, $J_t \sim I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}$ is the inert ideal

produced by EquivalentCoprimeIdeal, of odd prime norm $\mathrm{nrd}(J_t) \neq p$. The specifications of IdealMultiplication and RandomBoundedElement give

$$\alpha_{\mathrm{rsp}}^{(t)} \in \overline{J_t} I_{\mathrm{com}}^{(t)}, \qquad \frac{\mathrm{nrd}(\alpha_{\mathrm{rsp}}^{(t)})}{\mathrm{nrd}(J_t)\mathrm{nrd}(I_{\mathrm{com}}^{(t)})} \in \mathbb{Z}, \qquad \frac{\mathrm{nrd}(\alpha_{\mathrm{rsp}}^{(t)})}{\mathrm{nrd}(J_t)\mathrm{nrd}(I_{\mathrm{com}}^{(t)})} \leq 2^{e_{\mathrm{rsp}}-1}, \tag{1}$$

and $\gcd(\mathrm{nrd}(J_t), \mathrm{nrd}(I_{\mathrm{com}}^{(t)})) = 1$. Consequently,

$$\overline{\alpha_{\mathrm{rsp}}^{(t)}} \in \overline{I_{\mathrm{com}}^{(t)}} J_t \subseteq J_t.$$

If $\alpha_{\mathrm{rsp}}^{(t)} \notin \mathrm{nrd}(J_t)\mathcal{O}_0$, then $\overline{\alpha_{\mathrm{rsp}}^{(t)}}$ generates J_t together with $\mathrm{nrd}(J_t)\mathcal{O}_0$ as in the proof of Lemma 5. When $\mathrm{nrd}(J_t)$ is unknown, we test the prime divisors of $\mathrm{nrd}(\alpha_{\mathrm{rsp}}^{(t)})$ and apply the recovery procedure of Section 3.2.2. Algorithm 7 formalizes this procedure.

Algorithm 7 RecoverSecretIdealFromResponse

Input: A public key pk, a challenge integer $0 \leq c < 2^{\lambda}$, a recovered quaternion $\widehat{\rho}$, the odd prime norm $\mathrm{nrd}(J_t)$, and a verification repetition bound K.
Output: A signing-equivalent secret ideal for pk, or $\perp$.

if $\widehat{\rho} \notin \mathcal{O}_0 \setminus \{0\}$ **then**
  return $\perp$
end if
if $\overline{\widehat{\rho}} \in \mathrm{nrd}(J_t)\mathcal{O}_0$ **then**
  return $\perp$
end if
$\widehat{J} \leftarrow \mathsf{InertGenToIdealOdd}(\mathrm{nrd}(J_t), \overline{\widehat{\rho}})$
if $\widehat{J} = \perp$ **then**
  return $\perp$
end if
$\widehat{I} \leftarrow \mathsf{RecoverSecretIdealFromPrimeIdeal}(\mathsf{pk}, c, \widehat{J}, K)$
if $\widehat{I} \neq \perp$ **then**
  return $\widehat{I}$
end if
return $\perp$

Remark 5 (Leakage requirements and factorization). If $\mathrm{nrd}(J_t)$ is separately recovered, it can be directly used for the input of Algorithm 7. Otherwise the preprocessing which performs prime factorization of $\mathrm{nrd}(\widehat{\rho})$ including a large prime divisor $\mathrm{nrd}(J_t)$, which can be performed in quantum polynomial time as in Remark 4, is necessary. After that preprocessing, one can apply Algorithm 7 for all prime factors of $\mathrm{nrd}(\widehat{\rho})$ until the valid signing-equivalent secret ideal is returned.

Lemma 6 (Recovery from a response quaternion)**.** *Suppose that* $\widehat{\rho} = \alpha_{\mathrm{rsp}}^{(t)}$*, and* c *is the public challenge of the same successful signing execution. Then, under Heuristic 1, Algorithm 7 returns a secret ideal* $\widehat{I_{sk}}$ *such that* $\mathsf{RecoverSecretKey}(\mathsf{pk}, \widehat{I_{sk}})$ *is a valid secret key for* pk*.*

Proof. First, we claim that $\alpha_{\mathrm{rsp}}^{(t)} \notin \mathrm{nrd}(J_t)\mathcal{O}_0$ with an overwhelming probability. For an arbitrary left $\mathcal{O}_0$-ideal A, define

$$\mu(A) := \min_{0 \neq a \in A} \frac{\mathrm{nrd}(a)}{\mathrm{nrd}(A)}.$$

For any inert ideal A with its norm N, as a rank-two $\mathbb{Z}[i]$-lattice, it satisfies $A = \mathbb{Z}[i]N + \frac{1}{2}\mathbb{Z}[i](2x + (2y+1)i + j)$. Then, the Hermitian Gram matrix of A is

$$G_A = \begin{pmatrix} N^2 & N(2x-(2y+1)i)/2 \\ N(2x+(2y+1)i)/2 & ((2x)^2+(2y+1)^2+p)/4 \end{pmatrix},$$

and hence we have $\det(A) := \sqrt{\det G_A} = N\sqrt{p}/2$ and $\lambda_1(A) = \min_{\alpha \in A\setminus\{0\}} \mathrm{nrd}(\alpha) = N\mu(A)$. Hence Heuristic 1 gives

$$\Pr[\mu(A) \le \sqrt{2}^{e_{\mathrm{rsp}}-1}] \le \frac{4 \cdot 2^{e_{\mathrm{rsp}}-1}}{p}, \qquad A \in \{I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}, I_{\mathrm{com}}^{(t)}\},$$

since $2^{e_{\mathrm{rsp}}-1} < p/4$ for the specified parameters. Indeed, if $A' = A\theta$ for $\theta \in B_{p,\infty}^{\times}$, then $\lambda_1(A') = \mathrm{nrd}(\theta)\lambda_1(A)$ and $\mathrm{nrd}(A') = \mathrm{nrd}(\theta)\mathrm{nrd}(A)$, so $\mu(A') = \mu(A)$. Since $I_{\mathrm{com}}^{(t)}$ is equivalent to the ideal reduced within UniformO0ClassSampling, the same bound applies to $I_{\mathrm{com}}^{(t)}$. Since $\mathrm{nrd}(J_t) \in J_t$ and $J_t \sim I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}$, we have $\mathrm{nrd}(J_t) \ge \mu(J_t) = \mu(I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)})$. If $\alpha_{\mathrm{rsp}}^{(t)} = \mathrm{nrd}(J_t)\eta$ for some $\eta \in \mathcal{O}_0$, then $\mathrm{nrd}(J_t)\eta \in \overline{J_t} I_{\mathrm{com}}^{(t)} \subseteq I_{\mathrm{com}}^{(t)}$ since $\alpha_{\mathrm{rsp}}^{(t)} \in \overline{J_t}\mathcal{O}_0$. As $\mathrm{nrd}(I_{\mathrm{com}}^{(t)})$ is coprime to $\mathrm{nrd}(J_t)$, this implies $\eta \in I_{\mathrm{com}}^{(t)}$. Equation (1) and the response-degree bound therefore yield

$$\mu(I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)})\mu(I_{\mathrm{com}}^{(t)}) \le \mathrm{nrd}(J_t)\mu(I_{\mathrm{com}}^{(t)}) \le \frac{\mathrm{nrd}(\alpha_{\mathrm{rsp}}^{(t)})}{\mathrm{nrd}(J_t)\mathrm{nrd}(I_{\mathrm{com}}^{(t)})} \le 2^{e_{\mathrm{rsp}}-1}.$$

Thus, to satisfy $\alpha_{\mathrm{rsp}}^{(t)} \in \mathrm{nrd}(J_t)\mathcal{O}_0$, it should satisfy an event $\mu(I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}) \le \sqrt{2}^{e_{\mathrm{rsp}}-1}$ or $\mu(I_{\mathrm{com}}^{(t)}) \le \sqrt{2}^{e_{\mathrm{rsp}}-1}$, of probability at most $8 \cdot 2^{e_{\mathrm{rsp}}-1}/p = 2^{-\Omega(\lambda)}$ by the union bound. This probability is negligible, hence we have

$$\Pr\left[\alpha_{\mathrm{rsp}}^{(t)} \notin \mathrm{nrd}(J_t)\mathcal{O}_0\right] \text{ is overwhelming.}$$

Now, we prove that Algorithm 7 returns a signing-equivalent secret key for pk under the condition $\alpha_{\mathrm{rsp}}^{(t)} \notin \mathrm{nrd}(J_t)\mathcal{O}_0$. Set $L := \mathcal{O}_0\overline{\alpha_{\mathrm{rsp}}^{(t)}} + \mathrm{nrd}(J_t)\mathcal{O}_0$. Since $\overline{\alpha_{\mathrm{rsp}}^{(t)}} \in J_t$ and $\mathrm{nrd}(J_t)\mathcal{O}_0 \subseteq J_t$, we have $L \subseteq J_t$. Under $\mathcal{O}_0/\mathrm{nrd}(J_t)\mathcal{O}_0 \cong \mathrm{Mat}_2(\mathbb{F}_{\mathrm{nrd}(J_t)})$, the representation of $\overline{\alpha_{\mathrm{rsp}}^{(t)}}$ in $\mathrm{Mat}_2(\mathbb{F}_{\mathrm{nrd}(J_t)})$ is nonzero and has determinant zero by (1), so it has rank one. Its generated left ideal has dimension two over $\mathbb{F}_{\mathrm{nrd}(J_t)}$, so

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

Together with $L \subseteq J_t$, this proves $L = J_t$.
Equation (1) ensures that $\mathrm{nrd}(J_t) \mid \mathrm{nrd}(\alpha_{\mathrm{rsp}}^{(t)})$, and InertGenToIdealOdd therefore reconstructs J_t on this candidate. Then, by Lemma 4, Algorithm 7 outputs a valid secret ideal $\widehat{I}$ corresponding to pk. □

Remark 6 (Applicability of the lattice heuristic). For each $\Lambda \in \{I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}, I_{\mathrm{com}}^{(t)}\}$ in Lemma 6, we may additionally apply inert ideal reduction algorithm before invoking Heuristic 1 to satisfy its condition that each inert ideal is reduced. This computes a Minkowski-reduced basis of the same inert ideal Λ, without changing the underlying $\mathbb{Z}[i]$-lattice. Consequently, $\lambda_1(A)$, $\det(A)$, and $\mu(A) = \lambda_1(A)/\mathrm{nrd}(A)$ remain unchanged. Thus, this reduction can be inserted without breaking the correctness the recovery procedure.

4 Completing algebraic objects from partial MSB leakage

The preceding attacks assume that a leakage source returns an exact inert ideal or an exact quaternion element. In this section, we consider the more realistic case in which a trace reveals the correct most significant bits (MSBs) of each integer coefficient.

4.1 Extracting an inert ideal from partial MSB leakage

In this subsection, we propose an algorithm which completes a full inert ideal from its partial MSB of each integer component. This algorithm can be applied to recover the necessary leakage values for the algebraic secret key recovery algorithms proposed in Section 3.1 and Section 3.2.

Let Λ denote I_{sk} or the intersection $I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}$, or the equivalent prime-norm ideal $J_t \sim I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}$. Round-3 SQIsign represents Λ by a tuple $(N, (x, y))$ which represents an inert ideal, where $0 \le x, y < N$. Suppose that the leaked MSBs determine

$$z = A_z + \delta_z, \quad 0 \le \delta_z < X_z := 2^{u_z}, \quad z \in \{N, x, y\},$$

where A_z is the value obtained by appending u_z zero bits to the leaked MSB prefix of z, δ_z is its unknown u_z-bit suffix, and $X_z := 2^{u_z}$ is the width of the resulting candidate interval, so that $z = A_z + \delta_z$ with $0 \le \delta_z < X_z$, for each $z \in \{N, x, y\}$.

Write $\mathcal{I}_z := [A_z, A_z + X_z) \cap \mathbb{Z}$ for each $z \in \{N, x, y\}$. The exact tuple therefore belongs to the finite set $\mathcal{B}_{\mathbb{Z}} := \{(n, a, b) \in \mathcal{I}_N \times \mathcal{I}_x \times \mathcal{I}_y : n > 0,\ 0 \le a, b < n\}$.

Integral action constraints. Set

$$u := \frac{i+j}{2}, \qquad \kappa := \frac{p+1}{4}, \qquad g := x + yi + u.$$

The canonical basis returned by the algorithm InertIdealBasis is (N, Ni, g, ig). Since the reduced norm of every element of a norm-N ideal is divisible by N, the stored coefficients satisfy

$$F(x, y) := x^2 + y^2 + y + \kappa = \mathrm{nrd}(g) \equiv 0 \pmod{N}. \tag{2}$$

Conversely, this congruence suffices to validate a normalized tuple. Indeed, by letting $L := \mathbb{Z}N + \mathbb{Z}Ni + \mathbb{Z}g + \mathbb{Z}ig$, we have $L \subseteq \mathcal{O}_0$ and $iL = L$. Using $u^2 = -\kappa$ and $ui = -1 - iu$, we obtain

$$\begin{aligned} uN &= Ng - xN - yNi, \\ u(Ni) &= -N(ig) - (y+1)N + xNi, \\ ug &= xg - y(ig) - F(x, y), \\ u(ig) &= -(y+1)g - x(ig) + F(x, y)i. \end{aligned}$$

Thus $N \mid F(x, y)$ implies $uL \subseteq L$. Since $(1, i, u, iu)$ generate $\mathcal{O}_0$, L is an integral left $\mathcal{O}_0$-ideal. The displayed basis gives $\mathrm{nrd}(L) = N$, since $N \mid \mathrm{nrd}(g)$. Hence a normalized tuple defines an inert ideal of norm N exactly when (2) holds.

Algebraic pruning. The norm restriction depends on the leakage target. Set $e := 0$ for $\Lambda \in \{I_{\mathrm{sk}}, J_t\}$, and $e := \lambda$ for $\Lambda = I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}$. Since I_{sk} has odd prime norm coprime to p, whereas $\mathrm{nrd}(I_{\mathrm{chl}}^{(t)}) = 2^{\lambda}$, the genuine norm belongs to

$$\mathcal{N}_e := \{n \in \mathcal{I}_N : n = 2^e q, \quad q \text{ is an odd prime}, \quad q \neq p\}.$$

For each $n \in \mathcal{N}_e$, restrict the coefficient intervals to

$$\mathcal{X}_n := \mathcal{I}_x \cap [0, n), \quad \mathcal{Y}_n := \mathcal{I}_y \cap [0, n).$$

Discard n if either set is empty. Every remaining candidate must satisfy

$$a^2 \equiv -b^2 - b - \kappa \pmod{n}, \qquad (a, b) \in \mathcal{X}_n \times \mathcal{Y}_n. \tag{3}$$

Extracting the inert ideal. For each admissible norm n, we solve (3) by a meet-in-the-middle search. Store every $a \in \mathcal{X}_n$ in a table indexed by $a^2 \bmod n$. For each $b \in \mathcal{Y}_n$, retrieve all entries indexed by $(-b^2 - b - \kappa) \bmod n$. Each match directly determines an inert candidate $\widehat{I} := \big(n, (a, b)\big)$.

Every table entry retains all coefficients with that residue, so repeated residues do not cause candidates to be lost. Algorithm 8 returns the resulting candidate set.

Algorithm 8 CompleteInertIdeal

Require: The parameter p, the known exponent $e \in \{0, \lambda\}$, and, for each $z \in \{N, x, y\}$, the integer m_z representing leaked partial MSBs, and the number u_z of unknown LSBs.
Ensure: The set $\mathcal{S}$ of inert candidates for the leaked ideal Λ, satisfying the intervals and the target norm restriction.

For each $z \in \{N, x, y\}$, set $A_z \leftarrow 2^{u_z} m_z$, $X_z \leftarrow 2^{u_z}$, and $\mathcal{I}_z \leftarrow [A_z, A_z + X_z) \cap \mathbb{Z}$
$\kappa \leftarrow (p+1)/4$, $\mathcal{S} \leftarrow \emptyset$
for all $n \in \mathcal{N}_e := \{n' \in \mathcal{I}_N : n' = 2^e q,\ q \text{ is an odd prime}, q \neq p\}$ **do**
    Form $\mathcal{X}_n \leftarrow \mathcal{I}_x \cap [0, n)$, $\mathcal{Y}_n \leftarrow \mathcal{I}_y \cap [0, n)$
    if $\mathcal{X}_n = \emptyset$ or $\mathcal{Y}_n = \emptyset$ **then**
        continue
    end if
    Initialize a table H with empty lists as default entries.
    for all $a \in \mathcal{X}_n$ **do**
        Append a to $H[a^2 \bmod n]$
    end for
    for all $b \in \mathcal{Y}_n$ **do**
        $r \leftarrow (-b^2 - b - \kappa) \bmod n$
        for all $a \in H[r]$ **do**
            $\mathcal{S} \leftarrow \mathcal{S} \cup \{(n, (a, b))\}$
        end for
    end for
end for
return $\mathcal{S}$

Remark 7 (Intersection representation). Depending on the implementation, the intersection algorithm Intersect may return $\Lambda' = (I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)})(1-i)^{-1}$, rather than the intersection itself. Since $\mathrm{nrd}(1-i) = 2$, we have $\mathrm{nrd}(\Lambda') = 2^{\lambda-1}\mathrm{nrd}(I_{\mathrm{sk}})$. In this case, Algorithm 8 is applied with the modification $e = \lambda - 1$, and each completed candidate is right-multiplied by $1 - i$ before applying Algorithm 4. Otherwise, $e = \lambda$ is used as the current algorithm.

Lemma 7 (Correctness of CompleteInertIdeal). *With the notation above, Algorithm 8 terminates and returns exactly the set*

$$\mathcal{S} = \left\{ (n, (a, b)) : \begin{array}{ll} (n, a, b) \in \mathcal{B}_{\mathbb{Z}}, & n \in \mathcal{N}_e, \\ F(a, b) \equiv 0 & \pmod{n} \end{array} \right\}.$$

If all leaked MSBs of Λ are correct, then $\Lambda \in \mathcal{S}$.

Proof. All loops range over finite sets, so the algorithm terminates. Fix $n \in \mathcal{N}_e$. For each $a \in \mathcal{X}_n$, the algorithm stores a in the list indexed by $a^2 \bmod n$. When processing $b \in \mathcal{Y}_n$, it retrieves a exactly when $a^2 \equiv -b^2 - b - \kappa \pmod{n}$, which is equivalent to $F(a, b) \equiv 0 \pmod{n}$. Since each list retains every coefficient having its index, all pairs satisfying (3) are returned, and no other pair is returned. Moreover, for $n \in \mathcal{N}_e$, the condition $(a, b) \in \mathcal{X}_n \times \mathcal{Y}_n$ is equivalent to $(n, a, b) \in \mathcal{B}_{\mathbb{Z}}$. This proves the asserted description of $\mathcal{S}$.

By the integral-action argument above, a normalized tuple $(n,(a,b))$ represents an inert ideal of norm n if and only if $F(a,b) \equiv 0 \pmod{n}$. Hence the displayed set is precisely the claimed set of inert ideals.
Finally, if all leaked MSBs are correct, the tuple satisfies $(N,x,y) \in \mathcal{B}_{\mathbb{Z}}$. The target norm restriction gives $N \in \mathcal{N}_e$, and (2) gives $F(x,y) \equiv 0 \pmod{N}$. Therefore, $\Lambda = (N,(x,y)) \in \mathcal{S}$, which also proves the claim. □

To reconstruct a signing key, the adversary can apply Algorithm 2 to each candidate when $\Lambda = I_{\mathrm{sk}}$, Algorithm 4 when $\Lambda = I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}$, or Algorithm 5, and pass each accepted secret ideal to Algorithm 3. By Lemma 7, the correct inert ideal is included in $\mathcal{S}$.

Lemma 8 (Complexity of CompleteInertIdeal)**.** *Algorithm 8 has expected time and space complexities*

$$O\left(\left\lceil \frac{2^{u_N}}{2^e} \right\rceil 2^{u_x+u_y} \left(1+\log(2+p)+e+\sum_{z\in\{N,x,y\}} \left(u_z+\log(2+|m_z|)\right)\right)^{O(1)}\right).$$

Proof. See Appendix A.1. □

4.2 Extracting a quaternion element from partial MSB leakage

In this subsection, we provide two completion algorithms which reconstruct single quaternion elements for the inputs of algorithms in Section 3.3 from partial MSBs of their integer components.

4.2.1 Completing a quaternion element for Section 3.3.1

We first present the algorithm that recovers an exact value for Section 3.3.1 from partial MSBs of its integer components. Recall that $\alpha_t \in J_t$ is the value stored after conjugation in SampleQuaternionResponse, such that $J_t = I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)} \frac{\overline{\beta}_t}{\mathrm{nrd}(I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)})}$ whereas $\beta_t := \overline{\alpha_t} \in I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}$. Write the reduced standard-basis representation of α_t as

$$\alpha_t = \frac{a_{0,t} + a_{1,t}i + a_{2,t}j + a_{3,t}k}{r_t}, \quad r_t \in \{1,2\},$$

where $\gcd(a_{0,t}, a_{1,t}, a_{2,t}, a_{3,t}, r_t) = 1$. For each $0 \le s < 4$, let $m_{s,t}$ be the integer representing partial MSBs, and let $u_{s,t}$ be the number of unknown LSBs, so that

$$a_{s,t} = A_{s,t} + x_{s,t}, \qquad A_{s,t} := 2^{u_{s,t}} m_{s,t}, \qquad 0 \le x_{s,t} < 2^{u_{s,t}}.$$

Then, $\mathcal{I}_{s,t} := [A_{s,t}, A_{s,t} + 2^{u_{s,t}}) \cap \mathbb{Z}$ is the corresponding candidate interval. For negative coefficients, the MSB includes the sign bit and is interpreted in two's-complement form. If r_t is unknown, we run the procedure for both $r_t = 1$ and $r_t = 2$ and take the union of the returned sets.

Norm congruence. The norm relation established in Section 3.3 gives

$$\mathrm{nrd}(\alpha_t) = \mathrm{nrd}(\beta_t) = \mathrm{nrd}(J_t)\mathrm{nrd}(I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}) = \mathrm{nrd}(J_t)2^{\lambda}\mathrm{nrd}(I_{\mathrm{sk}}).$$

Set $M_t := r_t^2 2^{\lambda}$ and $(w_0, w_1, w_2, w_3) := (1, 1, p, p)$. The numerator coefficients therefore satisfy

$$\sum_{s=0}^{3} w_s a_{s,t}^2 \equiv 0 \pmod{M_t}. \tag{4}$$

This congruence does not require knowledge of $\mathrm{nrd}(J_t)$.

Exact meet-in-the-middle recovery. Substituting $a_{s,t} = A_{s,t} + x_{s,t}$ into (4) gives a bounded quadratic congruence in the four unknown suffixes. We split the coordinates as $\{0,3\}$ and $\{1,2\}$, and define

$$F_{03}(x_{0,t}, x_{3,t}) := (A_{0,t} + x_{0,t})^2 + p(A_{3,t} + x_{3,t})^2 \bmod M_t,$$
$$F_{12}(x_{1,t}, x_{2,t}) := (A_{1,t} + x_{1,t})^2 + p(A_{2,t} + x_{2,t})^2 \bmod M_t.$$

The correct leakage values satisfy

$$F_{03}(x_{0,t}, x_{3,t}) \equiv -F_{12}(x_{1,t}, x_{2,t}) \pmod{M_t}.$$

When $r_t = 2$, membership in $\mathcal{O}_0$ requires $a_{0,t} \equiv a_{3,t} \pmod 2$, and $a_{1,t} \equiv a_{2,t} \pmod 2$. This split allows the two parity conditions to be checked independently before matching. For each match, we retain the reconstructed quaternion only if its representation is reduced and it is a nonzero primitive element of $\mathcal{O}_0$. Every table entry stores all pairs with the same residue. This procedure is formalized in Algorithm 9.

Lemma 9 (Correctness of Algorithm 9)**.** *Algorithm 9 terminates and returns $\mathcal{B}_t$, a set of nonzero primitive quaternions in $\mathcal{O}_0$. If the leaked MSBs are correct for an honest signing execution, then $\alpha_t \in \mathcal{B}_t$.*

Proof. All loops range over finite sets, so the algorithm terminates. A quadruple is retrieved by the table exactly when its two pairs pass the parity tests and satisfy

$$F_{03}(x_{0,t}, x_{3,t}) + F_{12}(x_{1,t}, x_{2,t}) \equiv 0 \pmod{M_t}.$$

For a reconstructed $\widehat{\alpha} \in \mathcal{O}_0$, this is equivalent to $2^\lambda \mid \mathrm{nrd}(\widehat{\alpha})$, because $M_t = r_t^2 2^\lambda$. The remaining tests enforce a reduced representation, nonzero order membership, and primitivity. Thus every returned quaternion satisfies the stated conditions.
If the leaked MSBs are correct, then $x_{s,t} = a_{s,t} - A_{s,t} \in \{0, \ldots, 2^{u_{s,t}} - 1\}$ for every $0 \le s < 4$. Thus, the first and second pairs of loops enumerate $(x_{0,t}, x_{3,t})$ and $(x_{1,t}, x_{2,t})$, respectively, corresponding to the numerator coefficients of α_t.
Finally, the correct witness β_t is primitive in $\mathcal{O}_0$. Conjugation preserves $\mathcal{O}_0$ and primitivity, so $\alpha_t = \overline{\beta_t}$ is also primitive in $\mathcal{O}_0$. The norm relation above gives $2^\lambda \mid \mathrm{nrd}(\alpha_t)$, and correct leakage places its numerator coefficients in the prescribed intervals. With the correct reduced denominator 1 or 2, α_t therefore satisfies all conditions and belongs to $\mathcal{B}_t$. □

For secret-ideal recovery, use the output-ideal norm $\mathrm{nrd}(J_t)$ recovered from the same execution and run Algorithm 6, for each $\alpha \in \mathcal{B}_t$. These candidates are passed directly, since that algorithm performs conjugation internally. By Lemma 9, the correct candidate is included; by Lemma 5, its invocation reconstructs I_{sk} whenever verification succeeds.

If $\mathrm{nrd}(J_t)$ is unavailable, a quantum adversary may instead apply the factorization procedure of Remark 4 to each completed candidate.

Lemma 10 (Complexity of CompleteEquivalentQuaternion)**.** *Algorithm 9 has expected time and space complexities*

$$O\left(2^{\sum_{s=0}^{3} u_{s,t}} \left(1 + \log(2+p) + \lambda + \sum_{s=0}^{3} \left(u_{s,t} + \log(2 + |m_{s,t}|)\right)\right)^{O(1)}\right).$$

Proof. See Appendix A.2. □

Algorithm 9 CompleteEquivalentQuaternion

Require: The parameters p, λ, a denominator $r_t \in \{1, 2\}$, and, for each $0 \le s < 4$, the leaked signed MSB prefix $m_{s,t}$ and the number $u_{s,t}$ of unknown LSBs.
Ensure: The set $\mathcal{B}_t$ of candidates for the stored quaternion α_t, with the prescribed reduced representation, primitive in $\mathcal{O}_0$, and of norm divisible by 2^λ.

For each $0 \le s < 4$, set $A_{s,t} \leftarrow 2^{u_{s,t}} m_{s,t}$ and $\mathcal{I}_{s,t} \leftarrow [A_{s,t}, A_{s,t} + 2^{u_{s,t}}) \cap \mathbb{Z}$
$M_t \leftarrow r_t^2 2^\lambda$, $\mathcal{B}_t \leftarrow \emptyset$
Initialize a table $\mathcal{H}$ with empty lists as default entries.
for $x_{0,t} = 0, \ldots, 2^{u_{0,t}} - 1$ **do**
  for $x_{3,t} = 0, \ldots, 2^{u_{3,t}} - 1$ **do**
    if $r_t = 2$ and $A_{0,t} + x_{0,t} \not\equiv A_{3,t} + x_{3,t} \pmod 2$ **then**
      continue
    end if
    Append $(x_{0,t}, x_{3,t})$ to $\mathcal{H}[(A_{0,t} + x_{0,t})^2 + p(A_{3,t} + x_{3,t})^2 \bmod M_t]$
  end for
end for
for $x_{1,t} = 0, \ldots, 2^{u_{1,t}} - 1$ **do**
  for $x_{2,t} = 0, \ldots, 2^{u_{2,t}} - 1$ **do**
    if $r_t = 2$ and $A_{1,t} + x_{1,t} \not\equiv A_{2,t} + x_{2,t} \pmod 2$ **then**
      continue
    end if
    $z \leftarrow -\left((A_{1,t} + x_{1,t})^2 + p(A_{2,t} + x_{2,t})^2\right) \bmod M_t$
    for all $(x_{0,t}, x_{3,t}) \in \mathcal{H}[z]$ **do**
      $a_s \leftarrow A_{s,t} + x_{s,t}$ for $0 \le s < 4$
      $\widehat{\alpha} \leftarrow (a_0 + a_1 i + a_2 j + a_3 k)/r_t$
      if $\gcd(a_0, a_1, a_2, a_3, r_t) \neq 1$ or $\widehat{\alpha} \notin \mathcal{O}_0 \setminus \{0\}$ **then**
        continue
      end if
      if $\widehat{\alpha}$ is not primitive in $\mathcal{O}_0$ **then**
        continue
      end if
      $\mathcal{B}_t \leftarrow \mathcal{B}_t \cup \{\widehat{\alpha}\}$
    end for
  end for
end for
return $\mathcal{B}_t$

4.2.2 Completing a quaternion element for Section 3.3.2

We now consider partial MSB leakage of the response quaternion $\alpha_{\text{rsp}}^{(t)}$ of Section 3.3.2. The corresponding signature allows us to recover the response degree. We use it to match the unknown coefficient bits of the response quaternion element. The response degree $q_{\text{rsp}}^{(t)}$ satisfies

$$\text{nrd}(\alpha_{\text{rsp}}^{(t)}) = q_{\text{rsp}}^{(t)} \text{nrd}(J_t) \text{nrd}(I_{\text{com}}^{(t)}), \qquad 1 \le q_{\text{rsp}}^{(t)} \le 2^{e_{\text{rsp}}-1},$$

where all three factors are odd.

The isogeny used in verification can be reconstructed from the public key and signature by the same procedure with the verification algorithm. Let $\Phi_t : E_{\text{chl}}^{(t)} \times E_{\text{aux}}^{(t)} \longrightarrow F_1 \times F_2$ be the isogeny reconstructed during the verification, where F_1 and F_2 are elliptic curves. For the reconstructed basis (P, Q) of $E_{\text{chl}}^{(t)}[2^{e_{\text{rsp}}+2}]$, define points P_1, P_2, Q_1, Q_2 satisfying $\Phi_t\left(P, 0_{E_{\text{aux}}^{(t)}}\right) = (P_1, P_2)$, and $\Phi_t\left(Q, 0_{E_{\text{aux}}^{(t)}}\right) = (Q_1, Q_2)$. Then, restricting Φ_t to $E_{\text{chl}}^{(t)} \times$

$\{0_{E_{\mathrm{aux}}^{(t)}}\}$ and projecting onto F_1 or F_2 gives isogenies of degrees $q_{\mathrm{rsp}}^{(t)}$ and $2^{e_{\mathrm{rsp}}} - q_{\mathrm{rsp}}^{(t)}$, as in the same procedure with the verification algorithm of SQIsign.

Let $\psi_{\nu,t} : E_{\mathrm{chl}}^{(t)} \longrightarrow F_\nu,\ \nu \in \{1,2\}$ denote the two component isogenies obtained by restricting Φ_t to $E_{\mathrm{chl}}^{(t)} \times \{0_{E_{\mathrm{aux}}^{(t)}}\}$, so that $P_\nu = \psi_{\nu,t}(P)$, and $Q_\nu = \psi_{\nu,t}(Q)$. Their degrees satisfy $\{\deg \psi_{1,t}, \deg \psi_{2,t}\} = \{q_{\mathrm{rsp}}^{(t)},\ 2^{e_{\mathrm{rsp}}} - q_{\mathrm{rsp}}^{(t)}\}$.

Set $n := 2^{e_{\mathrm{rsp}}+2}$. Since (P,Q) is a basis of $E_{\mathrm{chl}}^{(t)}[n]$, $\zeta_n^{E_{\mathrm{chl}}^{(t)}}(P,Q)$ has order n. Compatibility of the Weil pairing with isogenies gives

$$\zeta_n^{F_\nu}(P_\nu, Q_\nu) = \zeta_n^{E_{\mathrm{chl}}^{(t)}}(P,Q)^{\deg \psi_{\nu,t}}.$$

Hence, since both component degrees are smaller than n, we have

$$\deg \psi_{\nu,t} = \log_{\zeta_n^{E_{\mathrm{chl}}^{(t)}}(P,Q)} \zeta_n^{F_\nu}(P_\nu, Q_\nu).$$

Finally, $q_{\mathrm{rsp}}^{(t)} \le 2^{e_{\mathrm{rsp}}-1}$ implies that it is the smaller component degree, and therefore

$$q_{\mathrm{rsp}}^{(t)} = \min_{\nu\in\{1,2\}} \left\{ \log_{\zeta_n^{E_{\mathrm{chl}}^{(t)}}(P,Q)} \zeta_n^{F_\nu}(P_\nu, Q_\nu) \right\}. \tag{5}$$

The logarithms are computable in polynomial time since n is a power of two.

Write the normalized representation of the response quaternion element as

$$\alpha_{\mathrm{rsp}}^{(t)} = \frac{a_{0,t} + a_{1,t}i + a_{2,t}j + a_{3,t}k}{r_t}, \qquad r_t \in \{1,2\},$$

with $\gcd(a_{0,t}, a_{1,t}, a_{2,t}, a_{3,t}, r_t) = 1$. For each $0 \le s < 4$, let $m_{s,t}$ be the integer representing partial MSBs, and let $u_{s,t}$ be the number of unknown LSBs, so that

$$a_{s,t} = A_{s,t} + x_{s,t}, \qquad A_{s,t} := 2^{u_{s,t}} m_{s,t}, \qquad 0 \le x_{s,t} < 2^{u_{s,t}}.$$

Then, $\mathcal{I}_{s,t} := [A_{s,t}, A_{s,t} + 2^{u_{s,t}}) \cap \mathbb{Z}$ is the corresponding candidate interval. Since $\mathrm{nrd}(\alpha_{\mathrm{rsp}}^{(t)})/q_{\mathrm{rsp}}^{(t)}$ is odd, we have

$$a_{0,t}^2 + a_{1,t}^2 + p(a_{2,t}^2 + a_{3,t}^2) \equiv r_t^2 q_{\mathrm{rsp}}^{(t)} \pmod{2r_t^2 q_{\mathrm{rsp}}^{(t)}}. \tag{6}$$

Algorithm 10 enumerates solutions within these intervals by matching the coordinate pairs $\{0,3\}$ and $\{1,2\}$. It retains only nonzero elements of $\mathcal{O}_0$ with normalized representations.

Lemma 11 (Correctness of CompleteResponseQuaternion)**.** *Algorithm 10 terminates and returns a set $\mathcal{B}_t \subseteq \mathcal{O}_0 \setminus \{0\}$. If the leaked MSBs and the input denominator are correct for an honest signing execution, then $\alpha_{\mathrm{rsp}}^{(t)} \in \mathcal{B}_t$.*

Proof. By (5), the first step of Algorithm 10 recovers the response degree $q_{\mathrm{rsp}}^{(t)}$ corresponding to the honest signature. For every $0 \le s < 4$, the set $\mathcal{I}_{s,t} = [2^{u_{s,t}} m_{s,t}, 2^{u_{s,t}} m_{s,t} + 2^{u_{s,t}}) \cap \mathbb{Z}$ is finite. Hence both enumeration loops and every list traversal in the algorithm are finite, so the algorithm terminates. Moreover, an element is inserted into $\mathcal{B}_t$ only if it belongs to $\mathcal{O}_0 \setminus \{0\}$. Therefore, we have

$$\mathcal{B}_t \subseteq \mathcal{O}_0 \setminus \{0\}.$$

Assume now that the leaked MSBs and the denominator r_t are correct. Write the normalized representation of the response quaternion element of this execution as $\alpha_{\mathrm{rsp}}^{(t)} = \frac{a_{0,t}+a_{1,t}i+a_{2,t}j+a_{3,t}k}{r_t}$. Correctness of the leaked MSBs means that, for each $0 \le s < 4$, there

Algorithm 10 CompleteResponseQuaternion

Require: The public key pk, an honest signature σ_t, the parameters p, e_{rsp}, a denominator $r_t \in \{1,2\}$, and, for each $0 \le s < 4$, the leaked signed MSB prefix $m_{s,t}$ and the number $u_{s,t}$ of unknown LSBs.

Ensure: A set $\mathcal{B}_t$ of candidates for $\alpha_{\mathrm{rsp}}^{(t)}$ with the prescribed reduced representation and coefficient intervals.

Recover $q_{\mathrm{rsp}}^{(t)}$ from (pk, σ_t) using (5)
For each $0 \le s < 4$, set $A_{s,t} \leftarrow 2^{u_{s,t}} m_{s,t}$ and $\mathcal{I}_{s,t} \leftarrow [A_{s,t}, A_{s,t} + 2^{u_{s,t}}) \cap \mathbb{Z}$
$M_t \leftarrow 2r_t^2 q_{\mathrm{rsp}}^{(t)}$, $\mathcal{B}_t \leftarrow \emptyset$
Initialize a table H with empty lists as default entries
for all $(b_{0,t}, b_{3,t}) \in \mathcal{I}_{0,t} \times \mathcal{I}_{3,t}$ **do**
    if $r_t = 1$ or $b_{0,t} \equiv b_{3,t} \pmod 2$ **then**
        Append $(b_{0,t}, b_{3,t})$ to $H[(b_{0,t}^2 + p b_{3,t}^2) \bmod M_t]$
    end if
end for
for all $(b_{1,t}, b_{2,t}) \in \mathcal{I}_{1,t} \times \mathcal{I}_{2,t}$ **do**
    if $r_t = 1$ or $b_{1,t} \equiv b_{2,t} \pmod 2$ **then**
        for all $(b_{0,t}, b_{3,t}) \in H[(M_t/2 - b_{1,t}^2 - p b_{2,t}^2) \bmod M_t]$ **do**
            $\widehat{\alpha} \leftarrow (b_{0,t} + b_{1,t} i + b_{2,t} j + b_{3,t} k)/r_t$
            if $\gcd(b_{0,t}, b_{1,t}, b_{2,t}, b_{3,t}, r_t) = 1$ and $\widehat{\alpha} \in \mathcal{O}_0 \setminus \{0\}$ **then**
                $\mathcal{B}_t \leftarrow \mathcal{B}_t \cup \{\widehat{\alpha}\}$
            end if
        end for
    end if
end for
return $\mathcal{B}_t$

exists an integer $x_{s,t} \in [0, 2^{u_{s,t}})$ such that $a_{s,t} = 2^{u_{s,t}} m_{s,t} + x_{s,t}$. Hence $a_{s,t} \in \mathcal{I}_{s,t}$ for every $0 \le s < 4$.

If $r_t = 2$, the condition $\alpha_{\mathrm{rsp}}^{(t)} \in \mathcal{O}_0$ implies $a_{0,t} \equiv a_{3,t} \pmod 2$ and $a_{1,t} \equiv a_{2,t} \pmod 2$. Thus the first loop stores $(a_{0,t}, a_{3,t})$, and the second loop processes $(a_{1,t}, a_{2,t})$.

Set $M_t = 2r_t^2 q_{\mathrm{rsp}}^{(t)}$ as in the algorithm. For the honest response quaternion, $q_{\mathrm{rsp}}^{(t)} \mid \mathrm{nrd}(\alpha_{\mathrm{rsp}}^{(t)})$ and $\mathrm{nrd}(\alpha_{\mathrm{rsp}}^{(t)})/q_{\mathrm{rsp}}^{(t)}$ is odd. Therefore, we have

$$a_{0,t}^2 + a_{1,t}^2 + p(a_{2,t}^2 + a_{3,t}^2) = r_t^2 \mathrm{nrd}(\alpha_{\mathrm{rsp}}^{(t)}) \equiv r_t^2 q_{\mathrm{rsp}}^{(t)} = \frac{M_t}{2} \pmod{M_t}.$$

Equivalently,

$$a_{0,t}^2 + p a_{3,t}^2 \equiv \frac{M_t}{2} - a_{1,t}^2 - p a_{2,t}^2 \pmod{M_t}.$$

Hence, when the second loop processes $(a_{1,t}, a_{2,t})$, it retrieves $(a_{0,t}, a_{3,t})$ from the table. Therefore, since a coefficient $a_{s,t}$ of $\alpha_{\mathrm{rsp}}^{(t)}$ is contained in the interval $\mathcal{I}_{s,t}$ for each $0 \le s < 4$, we conclude that $\alpha_{\mathrm{rsp}}^{(t)} \in \mathcal{B}_t$. □

Lemma 12 (Complexity of CompleteResponseQuaternion)**.** *Algorithm 10 has expected time and space complexities*

$$O\left(2^{\sum_{s=0}^{3} u_{s,t}} \left(1 + \log(2+p) + e_{\mathrm{rsp}} + \sum_{s=0}^{3} \left(u_{s,t} + \log(2 + |m_{s,t}|)\right)\right)^{O(1)}\right).$$

Proof. See Appendix A.3. □

5 Experiments

In this section, we experimentally extract leakages to recover each value which is necessary for the secret key recovery algorithm.

5.1 Experiment of leakage extraction for Section 3.1

We evaluate whether power leakage from the secret-matrix and CRT computations allows an adversary to recover the stored inert representation $I_{\mathsf{sk}} = (N_{\mathsf{sk}}, (x_{\mathsf{sk}}, y_{\mathsf{sk}}))$ targeted in Section 3.1. The source code of this section is available at:

https://github.com/suseong3e/cw-sqisign-response

Experimental setup. We run the Round-3 SQIsign `p500_27` arithmetic on a CW308 STM32F4 clocked at 7.3636 MHz. A ChipWhisperer captures TIO4-triggered power traces at 29.4545 MS/s with 25.09 dB gain; the matrix and CRT measurements use decimation factors 1 and 4, respectively. An honest host supplies operands generated during signing to the original arithmetic routines running on the target. The adversary observes only the traces, public challenges, and implementation, not the secret operands. We locate operation windows within each trace set without temporal re-alignment. Thus, the measurements concern isolated signing arithmetic, not complete on-device signing executions.

Target. We target the secret matrix multiplication $(c_1, c_2) = M_{\mathsf{sk}}(1, chl)$ in Sign (Algorithm 1) and the subsequent `ibz_crt` arithmetic. We consider the CRT branch in which the secret residue is $r_{\mathsf{sk}} \in \{x_{\mathsf{sk}}, y_{\mathsf{sk}}\}$; the other residue r_{chl} is determined by the challenge and M_{sk}. Writing $q = 2^{192}$ and $u = q^{-1} \bmod N_{\mathsf{sk}}$, the CRT recombination satisfies

$$r = r_{chl} + q\big((r_{\mathsf{sk}} - r_{chl})u \bmod N_{\mathsf{sk}}\big).$$

Listing 1 identifies the corresponding secret-dependent arithmetic: matrix word operations (W1, lines 1–3), CRT subtraction (W2, lines 5–6), and multiplication followed by recombination (W3, lines 7–9).

Leakage model. Unlike the profiled attacks of Section 5.2, we use challenge-dependent Hamming-weight predictions for 32-bit intermediates of a fixed target key. From 1,024 matrix traces, we extend candidate words of M_{sk} through carry dependencies and verify their predictions at subsequent stores. The resulting $\widehat{M}_{\mathsf{sk}}$ determines the challenge-dependent CRT operands. We then correlate predicted CRT subtraction and multiplication intermediates to constrain the low and remaining words of x_{sk}, y_{sk}, and u, and derive N_{sk} using the inert-ideal norm relation. For this stage we collect 2,112 traces, one per coordinate for each of 1,024 challenges and 32 disjoint held-out challenges.

Results. The searches retain one complete matrix candidate $\widehat{M}_{\mathsf{sk}}$ and one inert-ideal candidate with a 257-bit norm and two 256-bit coordinates. We freeze these candidates before the held-out evaluation. Figure 2 illustrates a representative CRT store and the correlation for the frozen candidate across 17 held-out challenges. Over 136 predicted leakage events from 17 held-out challenges per coordinate, the sum of absolute correlations is 116.806, compared with 25.061 at incorrect time windows and a maximum of 80.895 over 99 block-preserving challenge permutations. Only after this evaluation do we compare the candidates against the known answer, confirming every bit of M_{sk}, N_{sk}, x_{sk}, y_{sk}, and u.

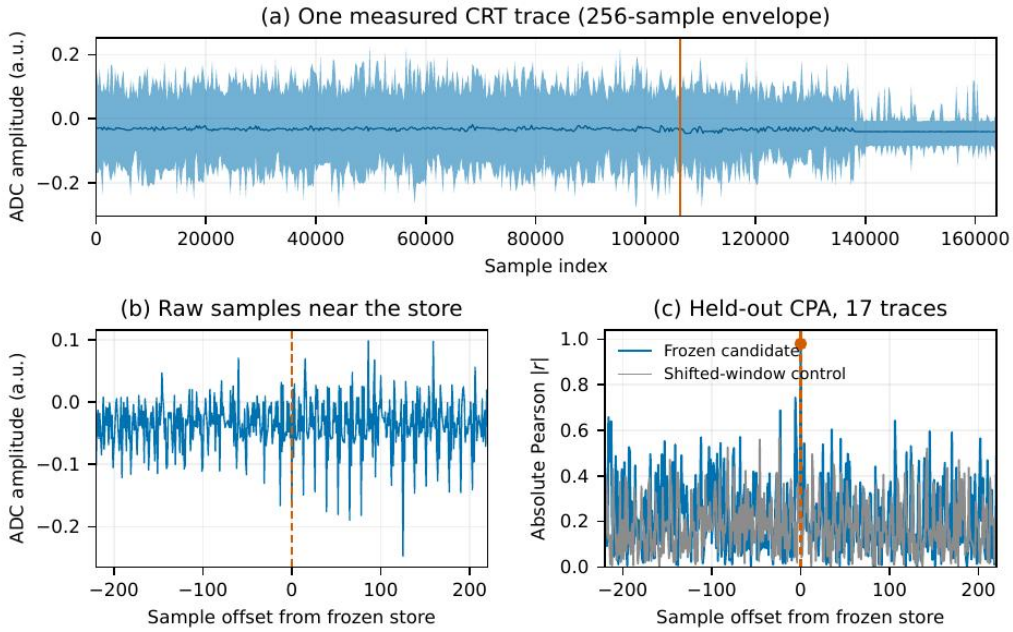


Figure 2: Measured CRT power leakage and held-out validation. (a) A 256-sample envelope of one CRT trace, with the selected store marked. (b) Raw samples around that store. (c) Absolute Pearson correlation for the frozen candidate across 17 held-out challenges, compared with a shifted-window control. The peak at the selected store supports the candidate's prediction on previously unused traces.

Implications for key recovery. For the evaluated key, the measured traces yield the exact inert representation $\widehat{I} = (\widehat{N}_{\mathsf{sk}}, (\widehat{x}_{\mathsf{sk}}, \widehat{y}_{\mathsf{sk}})) = I_{\mathrm{sk}}$ successfully. Consequently, Algorithms 2 and 3 can use this value, the public key, and fresh randomness to reconstruct a signing-equivalent secret key in polynomial time, by Lemma 2.

5.2 Experiments of leakage extraction for Section 3.2

In this subsection, we experimentally validate the two sublattice-based recovery strategies of Section 3.2 on a concrete microcontroller target, assessing whether each targeted subroutine leaks enough information about the secret ideal to drive the corresponding recovery algorithm. The source code of this section is available at:

https://github.com/OdysseyBeyond/sqisign-sca-sec52

Experimental setup. We run the reference implementation of Round-3 SQIsign [AAA+26] on an STM32F303 (ARM Cortex-M4) clocked at 7.37 MHz and acquire power traces with a ChipWhisperer-Lite capture board, sampling synchronously at four samples per clock (≈ 29.5 MHz) with 24.8 dB of gain. A GPIO trigger brackets the targeted computation, and traces are captured at a fixed offset without re-alignment. For each configuration we acquire 21,000 traces in a fixed-versus-random design: 18,000 traces use a freshly sampled secret ideal (the profiling set) and 3,000 reuse one fixed secret ideal (the attack set). Leakage detection and profiled template attacks follow standard methods [CRR02, MOP07].

5.2.1 Leakage from the intersection.

Target. We target the subroutine Intersect invoked on I_{sk} and $I_{\mathrm{chl}}^{(t)}$, whose output is the inert representation $I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)} = (N, (x, y))$ of the intersection sublattice with $N = N_1 N_2$ and $N_2 = n(I_{\mathrm{chl}}^{(t)})$. The secret-dependent coordinate is formed as $X_2 = x_{\mathrm{pre}} \cdot t \bmod N_2$ (and y analogously), after which x is obtained by CRT recombination with the a-side coordinate and the inert reduction. The acquisition window brackets this tail of Intersect, covering both the modular multiplications that form the coordinates and the subsequent materialization of (x, y). Listing 2 reproduces the traced tail of the reference code and marks the three regions analysed below: the Gaussian multiply (W1, lines 13–15), the coordinate-forming multiplies (W2, lines 23–31), and the CRT recombination with materialization of (x, y) (W3, lines 33–39).

Leakage model. We model the low limbs of the secret-dependent intermediates by their value and Hamming weight, and take the least-significant byte as the recovery target. Value leakage is isolated with the within-Hamming-weight value-SNR, which removes the Hamming-weight component so that only value-dependent variation remains, and its significance is assessed against a label-permutation null (z-score over 20 permutations). Profiled recovery uses bit-linear (stochastic) templates on the 12 points of interest of highest value-SNR; guessing entropy is computed leave-one-value-out and averaged over all 256 byte values, so that it is not biased by the particular fixed secret used for the attack set.

Results. The materialized output representation does not leak its value: at the CRT/inert-reduction window (W3) the low byte of each coordinate shows no value-SNR above the permutation null ($z < 1.5$) and its guessing entropy stays at chance (≈ 128 out of 256). The leakage that drives recovery is instead at the modular multiplications that *form* the secret-dependent quantities, where a secret limb enters as a multiplicand. Over the same traces, the two operands of the coordinate-forming multiply $X_2 = x_{\mathrm{pre}} \cdot t$ (W2) leak their value strongly (value-SNR 1.68 and 1.19; $z = 46$ and 34), whereas their reduced product X_2 does not ($z = 1.4$): the reduction mod N_2 diffuses the low limbs, and the final (x, y) is only copied into the output, never multiplied. An attacker therefore profiles the forming operands and recomputes (x, y) rather than templating the output itself. Fig. 3 and Table 4 report recovery of the low byte of the secret coordinate at the Gaussian multiply inside Intersect (W1), the earliest point at which a secret coordinate is consumed as a multiplicand.

Effect of limb granularity and optimization level. Both the big-integer limb width and the compiler optimization level change the exposed value leakage markedly. At `-O0`, where each limb is loaded and multiplied explicitly, the 16-bit build leaks more per trace than the 32-bit build (value-SNR 1.01 vs. 0.70) and reaches full single-byte recovery first (GE ≤ 1 at $N^\star = 19$ attack traces, against GE $= 1.3$ still at $N = 40$ for 32-bit): narrower limbs spread the same secret byte over more separately observable load/multiply events. Size optimization suppresses this. The 16-bit `-Os` build keeps the 16-bit limbs packed in 32-bit registers, its value-SNR drops to 0.10, and single-byte guessing entropy only falls to 6.3 at $N = 40$. In every build the leakage is per-limb, so recovery of a full coordinate proceeds limb by limb.

Remark 8 (Reduction of inert-ideal completion by low-byte leakage). Table 4 shows that the low byte of a secret-dependent inert-ideal coordinate can be ranked with guessing entropy 0.5 using 40 traces for the 16-bit `-O0` build. Correctly identified low bytes x_0, y_0 of two inert-ideal coordinates can be incorporated into Algorithm 8 by writing $x = x_0 + 2^8 X$ and $y = y_0 + 2^8 Y$. For B-bit coordinate ranges ($B \geq 8$), this reduces the number of

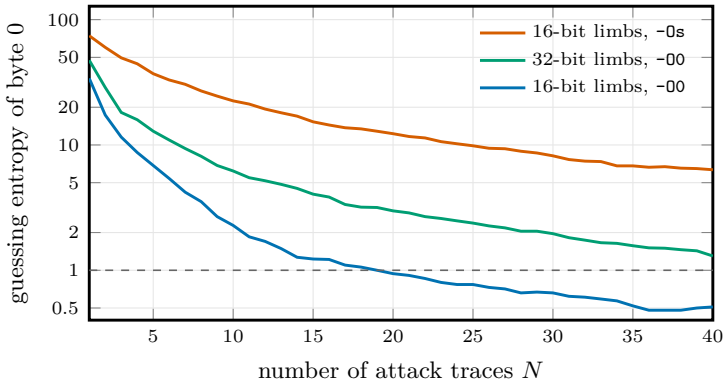


Figure 3: Leakage from Intersect: guessing entropy of the low byte of the secret coordinate at the Gaussian multiply, averaged over all byte values (leave-one-value-out), versus the number of attack traces, for three builds. Narrower limbs and `-O0` both lower the entropy; the materialized output coordinate (not shown) stays at chance.

Table 4: Leakage and recovery for the intersection experiment: within-Hamming-weight value-SNR of the low byte (matched profiling size) and leave-one-value-out guessing entropy at $N = 1, 10, 40$ attack traces, averaged over all byte values. $N^\star$ is the smallest N with GE ≤ 1.

Configuration	value-SNR	$\mathrm{GE}_{N=1}$	$\mathrm{GE}_{N=10}$	$\mathrm{GE}_{N=40}$	$N^\star$
16-bit, `-Os`	0.10	74.4	22.5	6.3	—
32-bit, `-O0`	0.70	47.4	6.2	1.3	—
16-bit, `-O0`	1.01	34.0	2.3	0.5	19

candidates on each side of its meet-in-the-middle search from at most 2^B to 2^{B-8}, and the raw pair space by a factor of 2^{16}. A low byte recovered for only one coordinate reduces only the corresponding side. These are conditional savings in the completion stage: the experiment does not demonstrate error-free recovery of both bytes, recovery of all remaining limbs or the ideal norm, or complete signing-key recovery.

5.2.2 Leakage from the equivalent prime ideal.

Target. We target the subroutine EquivalentCoprimeIdeal, which returns the odd-prime-norm inert ideal $J_t \sim I_{\mathrm{sk}} \cap I_{\mathrm{chl}}^{(t)}$ consumed by Algorithm 5. Once the enumeration has accepted a candidate of prime norm n, the implementation forms the inert representation $(n, (x, y))$ of J_t from the reduced generator $\gamma' = a + bi + cj + dk$, whose coordinates are taken modulo n: it computes $s = c^2 + d^2 \bmod n$, $t = (2s)^{-1} \bmod n$, the Gaussian product $(a + bi)(c - di) = (ac + bd) + (bc - ad)\, i$, and finally $x = (ac + bd)\, t \bmod n$ and $y = (bc - ad - s)\, t \bmod n$. The acquisition window covers lines 10–25 of Listing 3: it opens immediately before the Gaussian product, formed by the sign flip at line 11 and the multiply at lines 12–13, after the variable-time inversion at line 8, and closes once (x, y) have been formed at line 25. The enumeration and the primality test are executed on the host, which streams the exact input (γ', n) of this routine to the device; the device then runs the unmodified library code, so the traced arithmetic is that of the reference implementation. In the fixed-versus-random design the host re-seeds its sampler from the record, so the fixed class corresponds to one fixed J_t.

Leakage model and results. We use the metrics of Section 5.2.1. Two observations shape the template. First, the coordinates satisfy $a \equiv d$ and $b \equiv c \pmod 2$ (a consequence of the $\mathcal{O}_0$ basis), and the least-significant bit of each pair shifts the whole waveform: its correlation with the trace reaches $|r| = 0.82$ and 0.83 and persists over more than a thousand samples, consistent with a data-dependent timing difference of one cycle. This bit is recovered from a single trace. Second, the remaining seven bits leak through their Hamming weight at coordinate-specific points ($|r| = 0.35$, 0.35, 0.39, 0.25 for a, b, c, d), so points of interest must be selected per feature: the parity samples otherwise crowd out the rest and the attack stalls at one recovered bit. With 20 points chosen by the Hamming weight of bits 1–7 and 4 by bit 0, the leave-one-value-out guessing entropy of the low byte at `-Os` falls from 44–57 at $N = 1$ to 4.6, 11.7, 1.2 and 7.4 at $N = 40$ for a, b, c, d (Fig. 4, Table 5). At `-O0` the Hamming-weight correlations are higher ($|r| = 0.44$, 0.43, 0.47, 0.38) and the guessing entropy at $N = 40$ reaches 4.8, 4.5, 2.3 and 7.0; the unoptimized product is longer, so its window is 24,400 samples instead of 8,000 (the Hamming-weight leakage point of c moves from sample 4,634 to 21,014). The outputs x, y, the multiplier t and the modulus n show no value leakage anywhere in the 170k-sample window ($z \leq 1.5$ over 2,143 traces): all of them are produced or consumed behind the data-dependent reductions. The prime norm n is not recovered in either window: in the formation window its low limbs show no value leakage ($z \leq 1.5$), and at the start of its primality test (small-prime sieve and Montgomery set-up; 4,275 traces) the five low limbs show no Hamming-weight correlation above the noise floor ($|r| \leq 0.11$), whereas the top limb, essentially the bit length of n, produces a strong and persistent correlation ($|r| = 0.60$) through data-dependent control flow. Recovering n, which the attack needs, therefore remains open. Recovering the four coordinates limb by limb thus reconstructs $\gamma' \bmod n$, from which (x, y), and hence J_t given n, follow by the deterministic formation above.

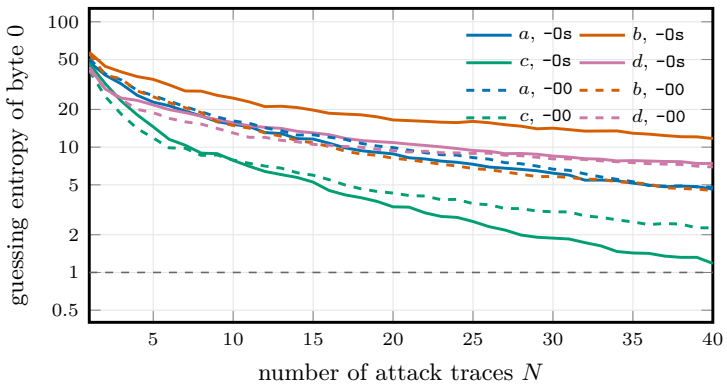


Figure 4: Leakage from EquivalentCoprimeIdeal: leave-one-value-out guessing entropy of the low byte of each coordinate of the reduced generator γ' of J_t at the Gaussian product, versus the number of attack traces, for the 32-bit `-Os` (solid; 8,000-sample window) and `-O0` (dashed; 24,400-sample window) builds.

Remark 9 (Ranked candidates for the equivalent prime ideal). The measured leakage supplies ranked hypotheses for the low bytes of the four generator coefficients a, b, c, d, but does not recover the prime norm n or the remaining limbs. To quantify the local search reduction with an explicit coverage criterion, let g_i be the zero-based mean guessing entropy of coefficient i at $N = 40$ in Table 5, and retain the top $L_i = \lceil 8g_i \rceil$ candidates per byte. Markov's inequality and the union bound give an average simultaneous inclusion probability of at least $1 - \sum_i g_i/L_i \geq 1/2$ under the reported evaluation distribution. The

Table 5: Leakage and recovery for the equivalent-prime-ideal experiment: correlation of the trace with bit 0 and with the Hamming weight of bits 1–7 of each coordinate, and leave-one-value-out guessing entropy at $N = 1, 10, 40$ attack traces.

Build	Coord.	$\lvert r\rvert_{\text{bit }0}$	$\lvert r\rvert_{\text{HW}}$	$\text{GE}_{N=1}$	$\text{GE}_{N=10}$	$\text{GE}_{N=40}$	$N^\star$
`-Os`	a	0.82	0.35	47.8	15.2	4.6	—
	b	0.83	0.35	57.2	24.6	11.7	—
	c	0.83	0.39	46.3	7.8	1.2	—
	d	0.82	0.25	43.6	15.7	7.4	—
`-O0`	a	0.82	0.44	50.7	16.2	4.8	—
	b	0.83	0.43	55.6	14.9	4.5	—
	c	0.83	0.47	40.8	7.9	2.3	—
	d	0.82	0.38	41.2	13.0	7.0	—

four-byte candidate volumes then become

$$\texttt{-Os} : 37 \cdot 94 \cdot 10 \cdot 60 \approx 2^{21.0},$$
$$\texttt{-O0} : 39 \cdot 36 \cdot 19 \cdot 56 \approx 2^{20.5},$$

compared with 2^{32} unconstrained four-byte tuples, a reduction of about 11.0 and 11.5 bits, respectively, in this local search. The coverage bound concerns the profiled repeated-input experiment, not fresh equivalent ideals in ordinary signing executions. As n and the higher generator limbs remain unknown, these figures neither establish reconstruction of J_t via Algorithm 5 nor imply a reduction in SQIsign's overall security level.

5.3 Experiments of leakage extraction for Section 3.3

We examine whether power traces reveal the partial MSBs of the two quaternion values targeted in Section 3.3: the conjugated equivalence witness and the random response quaternion. The source code of this section is available at:

https://github.com/my-rooftop/SQI-sign-analysis

Experimental setup. We use the `p324_3` reference implementation of SQIsign (commit `f417ebd`), with 32-bit limbs, on an STM32F415 (ARM Cortex-M4) mounted on a CW308 board and clocked at 7.3846 MHz. A PicoScope 3403D samples power traces at 250 MS/s (4 ns/sample), with AC coupling and a ±100 mV input range; GPIO triggers isolate the target operation. For the completed conjugation experiment, inputs are generated by executing the protocol on a host and replayed on the target with their original values and declared bit bounds. The targeted routine is unchanged, although its surrounding instructions differ from a complete signing execution. The response-quaternion experiment of Section 5.3.2 uses the same device, acquisition settings, and profiling methodology.

5.3.1 Leakage from quaternion conjugation.

Target. We target Conjugate at line 2 of SampleQuaternionResponse (Algorithm 3.18 in [AAA+26]), which replaces β_t by $\alpha_t = \overline{\beta_t}$ as in Section 3.3.1. The implementation copies the denominator and real coordinate and negates its three imaginary coordinates in place. As shown in Listing 4, the acquisition window brackets lines 5–8, the whole body of the wrapper, and the negations invoke `ibz_neg`, which processes the two's-complement representation limb by limb at lines 15–16. Each coordinate has a declared width of 401 bits and is therefore processed over 13 32-bit limbs even when its magnitude is smaller;

unused high limbs contain sign extension. The loop count at line 14 is independent of the secret value, but the words read and written at lines 15–16 are not.

Leakage model. Since the quaternion changes between signatures, recovery is evaluated from a single trace per test quaternion. Profiled templates [CRR02, MOP07] use 16,000 profiling traces and 4,000 held-out traces acquired on the same device. The labels comprise coordinate signs, limb Hamming weights (HWs), magnitude bit-lengths, and magnitude prefixes. For a coordinate v, the tested T-bit prefix is $\lfloor |v|/2^{L-T} \rfloor$, where L is the fixed magnitude-bit-length bound used for classification and $T = 1, \ldots, 8$; leading zeros are retained. We also train a multilayer perceptron on 14,000/2,000/4,000 training/validation/test traces, choosing the epoch on the validation set and including a shuffled-label control.

Results. Figure 5 shows the results of a fixed-versus-random Welch t-test with 60 traces per group, collected alternately. The maximum absolute t-statistic is 9.92, exceeding the threshold 4.5 at 2,647 sample points, whereas a fixed-versus-fixed control yields a maximum of 4.23. These results indicate data-dependent power leakage during quaternion conjugation.

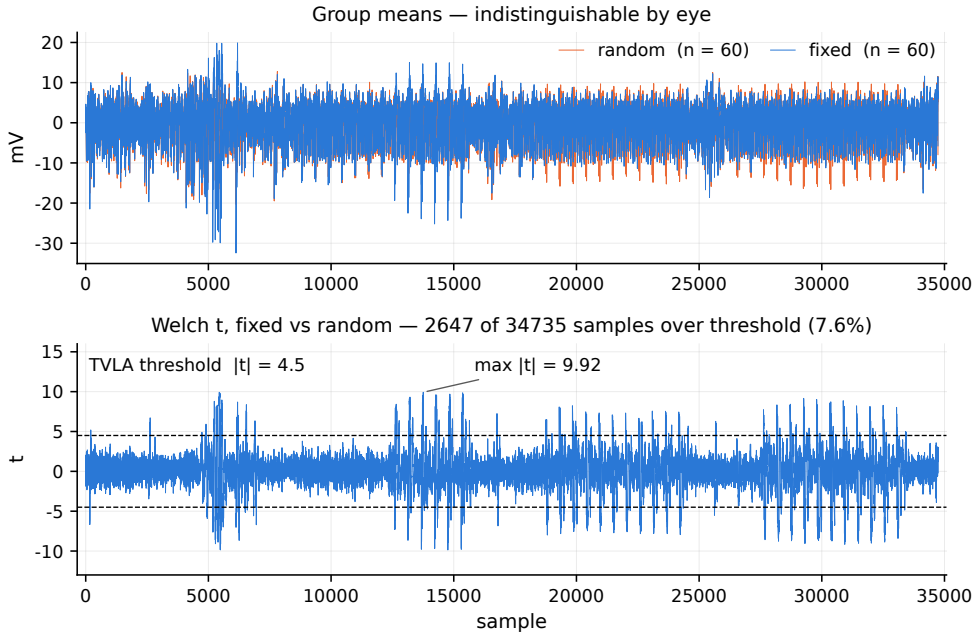


Figure 5: Power leakage from Conjugate in Algorithm 3.18, line 2. The upper plot shows the mean power traces for fixed and random inputs (60 traces each). The lower plot shows the corresponding Welch's t-statistics, where 2,647 of 34,735 samples exceed the threshold $|t| = 4.5$, with $\max |t| = 9.92$.

The t-statistic exhibits four successive regions of pronounced leakage, consistent with the real-coordinate copy and the three negations. In a separate correlation analysis of 200 random-input traces, representative non-sign-extension limbs exhibit Hamming-weight correlations $|r| = 0.40$–0.80. This localizes the observed leakage to the processed limb values, while high sign-extension limbs primarily reveal coefficient signs rather than independent magnitude bits.

Table 6 summarizes template classification. All four coordinate signs are recovered without error on the held-out set, and upper-limb HWs are predicted above baseline.

However, neither the templates nor the neural network improves upon the majority-class baseline for any tested prefix ($T = 1, \ldots, 8$) of any coordinate.

Table 6: Single-trace template classification on 4,000 held-out traces. HW denotes Hamming weight; the baseline predicts the most frequent class.

Target	Accuracy	Baseline
Coordinate sign	100.0%	50.0%
Upper-limb HW	26.6–28.6%	11–14%
Magnitude bit-length	27.4%	27.5%
Magnitude prefix ($T = 1, \ldots, 8$)	No improvement over baseline	

Implications for key recovery. Although the coefficient signs and limb Hamming weights are exposed, these observations do not determine the leading magnitude bits: distinct limb values can have the same Hamming weight, and the tested classifiers do not recover coefficient bit-lengths or magnitude prefixes above baseline. Consequently, the measured leakage does not provide the sufficiently narrow coefficient intervals required by Section 4.2.1, and key recovery from this operation remains undemonstrated under the evaluated setup.

5.3.2 Leakage from a response quaternion.

Target. We target RandomBoundedElement at line 5 of SampleQuaternionResponse (Algorithm 3.18 in [AAA+26]), which produces the response quaternion $\alpha_{\mathrm{rsp}}^{(t)}$ of Section 3.3.2. The implementation is `quat_lattice_sample_from_ball`, which rejection-samples a vector from a parallelogram (Listing 5, lines 2–6) and then forms the element from the accepted vector: the matrix evaluation at line 8, the copy of the lattice denominator at line 9, and the normalization at line 10. The acquisition window is line 21, the write-out of the four accumulated coordinates inside `ibz_mat_4x4_eval`, bracketed once per coordinate. We trace the write-out rather than the accumulation at lines 16–20 because the latter costs 39,085 cycles, some 1.3 million samples at 4 ns, which no single capture window of the instrument holds; the write-out is a few hundred cycles and carries the complete formed value.

Two properties of the enclosing routine preclude tracing it as a whole. A single call costs roughly 5.6 s on the target, and the rejection loop executes a data-dependent number of iterations, so traces of complete calls are not mutually aligned. The host therefore executes the protocol up to line 5, retains the lattice and the sample vector that the rejection loop actually accepted, and streams them to the device, which executes the unmodified library routine on that input; the device output was verified equal to the host's $\alpha_{\mathrm{rsp}}^{(t)}$ before acquisition. The measurement thus concerns the formation arithmetic in isolation, not a complete signing execution.

Leakage model. We reuse the methodology of Section 5.3.1. As $\alpha_{\mathrm{rsp}}^{(t)}$ is redrawn for every signature, recovery is evaluated from a single trace, with 16,000 profiling and 4,000 held-out traces acquired on the same device; the neural-network classifier uses a 14,000/2,000/4,000 training/validation/test split with the epoch chosen on the validation set. Labels are the values written at line 21 of Listing 5, the output of `ibz_mat_4x4_eval` rather than the normalized $\alpha_{\mathrm{rsp}}^{(t)}$; the two coincide on 66.3% of traces and differ by a factor of two on the remainder, where the gcd is 2. Since normalization by two shifts fixed-width magnitude prefixes, the reported classification results concern the measured pre-normalization coefficients; transferring them to the normalized quaternion requires accounting for the normalization factor. Each coordinate is labelled by its sign, its magnitude

bit-length, the Hamming weights of its 32-bit limbs, and its T-bit magnitude prefix for $T = 1, \ldots, 8$. Information estimates are reported against a shuffled-label control run through the identical pipeline, since selecting points of interest by correlation and then fitting templates at those same points extracts apparent information from unrelated traces.

Results. The quantity to be inferred is 746 bits: four coordinates of median magnitude 267, 267, 104 and 104 bits, together with their four signs. Of the 36 limbs processed, 22 lie entirely below the magnitude and carry value; the remainder are sign extension or boundary limbs whose Hamming weight is determined by the sign.

Figure 6 reports a fixed-versus-random Welch t-test over 1,000 traces. The maximum absolute t-statistic is 25.62, exceeding the threshold 4.5 at 4,553 of 25,492 sample points (17.9%), while a fixed-versus-fixed control reaches only 4.11 and crosses the threshold nowhere. The t-trace separates into two broad regions and two narrower ones, matching the per-limb correlation peaks of the two 267-bit and two 104-bit coordinates respectively.

Resampled to the $60 + 60$ split used there and averaged over 40 repetitions, the same data give $\max |t| = 10.49 \pm 0.84$ with $7.61 \pm 0.92\%$ of samples over threshold, against 9.92 and 7.62% for Conjugate. The two operations are therefore indistinguishable in leakage strength per trace. Over 20,000 traces, all 22 value-carrying limbs correlate with their Hamming weight above the multiple-comparison noise floor of 0.032, with $|r|$ ranging to 0.725 (median 0.533) and SNR at the selected point ranging to 5.27 (median 2.16). The limbs appear in time order about four clock cycles apart, which localizes the leakage to the individual limb write-outs.

Table 7 summarizes single-trace classification. All four coordinate signs are recovered essentially without error, and limb Hamming weights are predicted well above baseline, yielding 9.0 bits of information once the 0.5-bit control is subtracted, against a ceiling of 78 bits were every limb's Hamming weight known exactly. Neither the magnitude bit-length nor any magnitude prefix is recovered: bit-length classification is at baseline for all four coordinates ($z = -0.3, -0.2, 1.2, -0.5$; 0.09 bits in total), and for every $T = 1, \ldots, 8$ the prefix accuracy equals the majority-class baseline.

Table 7: Single-trace classification of response-quaternion coefficients on 4,000 held-out traces. HW denotes Hamming weight; the baseline predicts the most frequent class.

Target	Accuracy	Baseline
Coefficient sign	99.75–99.95%	50.0%
Limb HW (templates)	16–27%	$\approx 5\%$
Limb HW (neural network)	50–58%	13.6–14.0%
Limb byte (neural network)	0.88–1.27%	0.70–0.85%
Magnitude bit-length	24.2–25.9%	24.5–25.1%
Magnitude prefix ($T = 1, \ldots, 8$)	No improvement over baseline	

The neural network changes the magnitude of the Hamming-weight result but not its character. It classifies limb Hamming weight at 50–58% against a 13.6–14.0% baseline, far above the Gaussian templates. Classification of a full byte of the same limb is a much weaker effect: it stays at baseline on two of the three limbs examined (0.93% and 0.88% against 0.85% and 0.80%) and exceeds it on the third (1.27% against 0.70%, $z = +4.4$). Shuffled-label controls stay within $z \approx \pm 2$ throughout, so the latter effect is real, but it is an increase of 0.57 percentage points on a 256-class problem, against the 44-point margin the same network achieves on the 33-class weight. Naming a byte requires separating values within one Hamming-weight class; the measurements indicate that the leakage is concentrated in the Hamming weight, without being strictly confined to it. The decisive test remains the prefix classification, which is at baseline for every T and every coordinate.

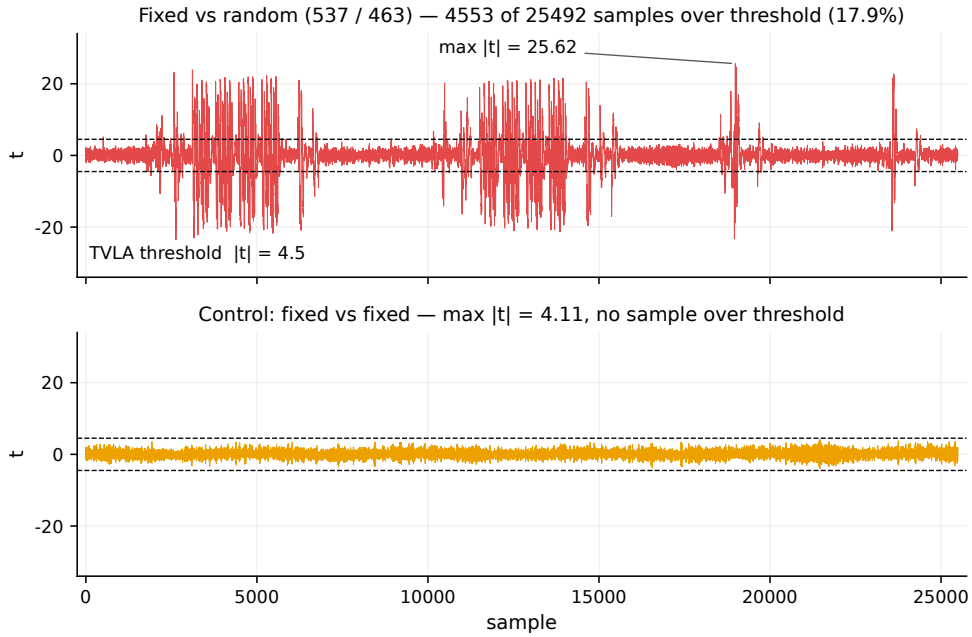


Figure 6: Power leakage from RandomBoundedElement in Algorithm 3.18, line 5. The upper plot shows Welch's t-statistic between 537 fixed-input and 463 random-input traces, with 4,553 of 25,492 samples exceeding $|t| = 4.5$ and $\max |t| = 25.62$. The lower plot shows the fixed-versus-fixed control, which stays within the threshold and confirms that the upper result is data-dependent leakage rather than acquisition drift.

Implications for key recovery. The measurement reproduces, on a structurally different operation, the outcome of Section 5.3.1. The signs of all four coefficients are recovered from a single trace, and Hamming-weight leakage is unambiguous, but the leading magnitude bits are not obtained: the bit-length is not recoverable, so the position of the most significant bit is unknown, and Hamming weight is symmetric in the bits, so it cannot discriminate within a weight class. The estimated 9.0 bits of Hamming-weight information quantify statistical dependence between the traces and the Hamming-weight labels, but do not by themselves establish a corresponding reduction in exhaustive-search complexity. Consequently the measured leakage does not supply the signed coefficient prefixes required by Section 4.2.2, and CompleteResponseQuaternion cannot be driven from this operation under the evaluated setup.

That two operations of different arithmetic structure—an in-place negation and a multiply-accumulate formation—of different operand size and different sign-extension fraction yield the same outcome suggests that the obstacle is not specific to either implementation, but follows from the predominantly Hamming-weight character of the available leakage.

5.3.3 Discussion and future works for experiments on quaternion elements

Both quaternion experiments reveal coefficient signs and significant limb Hamming-weight information, but neither recovers the magnitude prefixes required by the completion algorithms of Section 4.2. A possible direction is to use the measured Hamming-weight distributions as soft algebraic constraints, rather than attempting to infer MSBs directly: candidate quaternions could be ranked jointly by these distributions and by the norm congruences derived therein, including the publicly recoverable response-degree relation in

the second experiment. Such a method would also need to account for classification errors and for the fact that response quaternions change across signing executions. Whether these constraints enable a practical recovery algorithm, or one whose total cost falls below the claimed security level of SQIsign, remains an open question.

Acknowledgments

Use of generative AI tools. We have used OpenAI's ChatGPT 6 Pro as a discussion aid during our research, such as specification of research ideas, and to check the grammar and spelling of author-written manuscript text. We have also used Claude Opus 5.5 to assist in developing source code for this work. The authors have reviewed all of the AI-assisted materials, and take full responsibility for the technical claims, proofs, experimental results, and source code presented in this work.

References

[AAA+26] Marius A. Aardal, Gora Adj, Diego F. Aranha, Andrea Basso, Giacomo Borin, Isaac Andrés Canales Martínez, Jorge Chávez-Saab, Maria Corte-Real Santos, Pierrick Dartois, Luca De Feo, Max Duparc, Thomas Espitau, Jonathan Komada Eriksen, Tako Boris Fouotsa, Décio Luiz Gazzoni Filho, Basil Hess, Riccardo Invernizzi, David Kohel, Antonin Leroux, Patrick Longa, Luciano Maino, Michael Meyer, Marzio Mula, Kohei Nakagawa, Hiroshi Onuki, Lorenz Panny, Sikhar Patranabis, Christophe Petit, Giacomo Pope, Krijn Reijnders, Damien Robert, Francisco Rodríguez-Henríquez, Sina Schaeffler, Frederik Vercauteren, Alexandre Wallet, Benjamin Wesolowski, and Wessel van Woerden. SQIsign. Technical report, National Institute of Standards and Technology, 2026.

[ABC+26] Gorjan Alagic, Maxime Bros, Pierre Ciadoux, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, Hamilton Silberg, Daniel Smith-Tone, and Noah Waller. Status report on the second round of the additional digital signature schemes for the NIST post-quantum cryptography standardization process. Technical report, National Institute of Standards and Technology, 2026.

[AMBB+25] Carlos Aguilar Melchor, Slim Bettaieb, Loïc Bidoux, Thibauld Feneuil, Philippe Gaborit, Nicolas Gama, Shay Gueron, James Howe, Andreas Hülsing, David Joseph, Antoine Joux, Mukul Kulkarni, Edoardo Persichetti, Tovohery H. Randrianarisoa, Matthieu Rivain, and Dongze Yue. The syndrome decoding in the head (SD-in-the-head) signature scheme: Algorithm specifications and supporting documentation, version 2.0, 2025.

[BBB+25] Carsten Baum, Ward Beullens, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Christian Majenz, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, et al. FAEST v2: Algorithm Specifications, 2025.

[BBFR25] Ryad Benadjila, Charles Bouillaguet, Thibauld Feneuil, and Matthieu Rivain. MQOM: MQ on my mind: Algorithm specifications and supporting documentation, version 2.0, 2025.

[BCC+25a] Ward Beullens, Fabio Campos, Sofía Celi, Basil Hess, and Matthias J. Kannwischer. MAYO update, 2025. Presentation at the Sixth PQC Standardization Conference, National Institute of Standards and Technology (NIST).

[BCC+25b] Ward Beullens, Fabio Campos, Sofía Celi, Basil Hess, and Matthias J. Kannwischer. MAYO: Round 2 version, 2025.

[BCD+25] Ward Beullens, Ming-Shing Chen, Jintai Ding, Boru Gong, Matthias J. Kannwischer, Jacques Patarin, Bo-Yuan Peng, Dieter Schmidt, Cheng-Jhih Shih, Chengdong Tao, and Bo-Yin Yang. UOV: Unbalanced oil and vinegar: Algorithm specifications and supporting documentation, version 2.0, 2025.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation Power Analysis with a Leakage Model. In *Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.

[CRR02] Suresh Chari, Josyula R Rao, and Pankaj Rohatgi. Template attacks. In *International workshop on cryptographic hardware and embedded systems*, pages 13–28. Springer, 2002.

[Deu41] 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.

[DF17] Luca De Feo. Mathematics of isogeny based cryptography. *arXiv preprint arXiv:1711.04062*, 2017.

[DFKL+20] 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.

[FIH+25] Hiroki Furue, Yasuhiko Ikematsu, Fumitaka Hoshino, Tsuyoshi Takagi, Haruhisa Kosuge, Kimihiro Yamakoshi, Rika Akiyama, Satoshi Nakamura, Shingo Orihara, and Koha Kinjo. QR-UOV, 2025.

[GJJR11] Gilbert Goodwill, Benjamin Jun, Josh Jaffe, and Pankaj Rohatgi. A Testing Methodology for Side-Channel Resistance Validation. In *NIST Non-Invasive Attack Testing Workshop*, 2011.

[Ibu82] 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.

[JMKR23] David Jacquemin, Anisha Mukherjee, Péter Kutas, and Sujoy Sinha Roy. Ready to SQI? Safety First! Towards a constant-time implementation of isogeny-based signature, SQIsign. *Cryptology ePrint Archive*, 2023.

[Ler22] Antonin Leroux. Quaternion Algebra and isogeny-based cryptography. *Theses, Ecole doctorale de l'Institut Polytechnique de Paris*, 2022.

[LHK+24] Jeonghwan Lee, Donghoe Heo, Hyeonhak Kim, Gyusang Kim, Suhri Kim, Heeseok Kim, and Seokhie Hong. Fault attack on SQIsign. In *International Conference on Post-Quantum Cryptography*, pages 54–76. Springer, 2024.

[LS26] Antonin Leroux and Sina Schaeffler. New algorithms for quaternion ideals in SQIsign. Cryptology ePrint Archive, Paper 2026/2153, 2026.

[MCJ+25] Anisha Mukherjee, Maciej Czuprynko, David Jacquemin, Péter Kutas, and Sujoy Sinha Roy. Simple Power Analysis Attack on SQIsign. In *International Conference on Cryptology in Africa*, pages 245–269. Springer, 2025.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks: Revealing the secrets of smart cards*. Springer, 2007.

[Nat24] National Institute of Standards and Technology (US). Stateless hash-based digital signature standard. Technical report, 2024.

[NLLM26] Ben Nelson, Joshua Limbrey, Cong Ling, and Andrew Mendelsohn. Cryptanalysis of HAWK: a guessing game. Cryptology ePrint Archive, Paper 2026/1318, 2026.

[oST24] National Institute of Standards and Technology. Module-Lattice-Based digital signature standard. Technical report, 2024.

[PFH+20] 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.

[Ran26] Lars Ran. Wedges, Oil, and Vinegar: An Analysis of UOV in the Exterior Algebra. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 363–388. Springer, 2026.

[Sil09] Joseph H Silverman. *The arithmetic of elliptic curves*, volume 106. Springer, 2009.

[SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. A Stochastic Model for Differential Side Channel Cryptanalysis. In *Cryptographic Hardware and Embedded Systems – CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 30–46. Springer, 2005.

[SMY09] François-Xavier Standaert, Tal G. Malkin, and Moti Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2009.

[SW26] Zygimantas Straznickas and Stephen A. Weis. HAWK-n key recovery reduces to SVP in dimension $n/2 + 1$. Cryptology ePrint Archive, Paper 2026/1593, 2026.

[Vél71] Jacques Vélu. Isogénies entre courbes elliptiques. *Comptes-Rendus de l'Académie des Sciences*, 273:238–241, 1971.

[Voi21] John Voight. *Quaternion algebras*. Springer, 2021.

[WCD+25] Lih-Chung Wang, Chun-Yen Chou, Jintai Ding, Yen-Liang Kuan, Jan Adriaan Leegwater, Ming-Siou Li, Bo-Shu Tseng, Po-En Tseng, and Chia-Chun Wang. SNOVA: Proposal for NISTPQC: Additional digital signature schemes, version 2.0, 2025.

A Additional proofs

A.1 Proof of Lemma 8

The interval

$$\mathcal{I}_N = [A_N, A_N + 2^{u_N}) \cap \mathbb{Z}$$

contains exactly 2^{u_N} consecutive integers. The distance between two consecutive integers divisible by 2^e is 2^e. Consequently, the number of integers in $\mathcal{I}_N$ divisible by 2^e is at most

$$\left\lceil \frac{2^{u_N}}{2^e} \right\rceil.$$

Since every element of $\mathcal{N}_e$ is divisible by 2^e, we have

$$|\mathcal{N}_e| \le \left\lceil \frac{2^{u_N}}{2^e} \right\rceil.$$

Such integers can be enumerated by starting from the first multiple of 2^e in $\mathcal{I}_N$ and increasing by 2^e. For each enumerated integer, testing whether its quotient by 2^e is an odd prime distinct from p takes time polynomial in its bit length.

Fix $n \in \mathcal{N}_e$. Since

$$\mathcal{X}_n \subseteq \mathcal{I}_x, \qquad \mathcal{Y}_n \subseteq \mathcal{I}_y,$$

and the two input intervals have cardinalities

$$|\mathcal{I}_x| = 2^{u_x}, \qquad |\mathcal{I}_y| = 2^{u_y},$$

we have

$$|\mathcal{X}_n| \le 2^{u_x}, \qquad |\mathcal{Y}_n| \le 2^{u_y}.$$

The first inner loop performs at most 2^{u_x} insertions into H, and the second inner loop performs at most 2^{u_y} lookups. For each $b \in \mathcal{Y}_n$, every element retrieved from H corresponds to a distinct $a \in \mathcal{X}_n$. Therefore, over the complete execution for this fixed n, the innermost loop is executed at most

$$|\mathcal{X}_n|\,|\mathcal{Y}_n| \le 2^{u_x+u_y}$$

times.

All modular squarings, reductions, primality tests, table operations, and candidate insertions have bit complexity polynomial in

$$1 + \log(2+p) + e + \sum_{z \in \{N,x,y\}} \big(u_z + \log(2 + |m_z|)\big),$$

which bounds the bit lengths of the input-derived integers and candidate representations. Since

$$2^{u_x} + 2^{u_y} \le 2 \cdot 2^{u_x+u_y},$$

the expected time for a fixed n is

$$O\left(2^{u_x+u_y}\left(1 + \log(2+p) + e + \sum_{z \in \{N,x,y\}} \big(u_z + \log(2 + |m_z|)\big)\right)^{O(1)}\right).$$

Multiplying by the number of possible elements of $\mathcal{N}_e$ gives the claimed expected time bound.

For a fixed n, the table H contains at most 2^{u_x} stored coefficients. Moreover, at most $2^{u_x+u_y}$ candidates can be generated for this n. Hence, over all $n \in \mathcal{N}_e$, the returned set contains at most

$$\left\lceil \frac{2^{u_N}}{2^e} \right\rceil 2^{u_x+u_y}$$

candidates. Each table entry and each returned candidate has representation length polynomial in the input bit length. Since

$$2^{u_x} \leq \left\lceil \frac{2^{u_N}}{2^e} \right\rceil 2^{u_x+u_y},$$

the table, the returned candidate set, and all auxiliary values together require

$$O\left(\left\lceil \frac{2^{u_N}}{2^e} \right\rceil 2^{u_x+u_y} \left(1 + \log(2+p) + e + \sum_{z\in\{N,x,y\}} \big(u_z + \log(2+|m_z|)\big) \right)^{O(1)} \right)$$

space.

A.2 Proof of Lemma 10

The first two nested loops enumerate all pairs

$$(x_{0,t}, x_{3,t}) \in \{0, \ldots, 2^{u_{0,t}} - 1\} \times \{0, \ldots, 2^{u_{3,t}} - 1\}.$$

Thus, before the parity test, they enumerate exactly

$$2^{u_{0,t}+u_{3,t}}$$

pairs. The parity test can only discard pairs, so the number of entries inserted into $\mathcal{H}$ is at most $2^{u_{0,t}+u_{3,t}}$.

Likewise, the second two nested loops enumerate exactly

$$2^{u_{1,t}+u_{2,t}}$$

pairs $(x_{1,t}, x_{2,t})$ before their parity test. Each surviving pair performs one lookup in $\mathcal{H}$.

Every execution of the innermost loop is determined by one pair $(x_{0,t}, x_{3,t})$ from the first pair space and one pair $(x_{1,t}, x_{2,t})$ from the second pair space. Since each pair on each side is enumerated at most once, the innermost loop is executed at most

$$2^{u_{0,t}+u_{3,t}} \cdot 2^{u_{1,t}+u_{2,t}} = 2^{\sum_{s=0}^{3} u_{s,t}}$$

times.

Each execution performs only polynomial-time integer operations: forming the four coefficients, constructing $\widehat{\alpha}$, computing a greatest common divisor, testing membership in $\mathcal{O}_0$, testing primitivity, and inserting an accepted candidate into $\mathcal{B}_t$. All integers involved have bit lengths bounded by

$$O\left(1 + \log(2+p) + \lambda + \sum_{s=0}^{3} \big(u_{s,t} + \log(2+|m_{s,t}|)\big)\right).$$

Furthermore,

$$2^{u_{0,t}+u_{3,t}} \leq 2^{\sum_{s=0}^{3} u_{s,t}},$$
$$2^{u_{1,t}+u_{2,t}} \leq 2^{\sum_{s=0}^{3} u_{s,t}}.$$

Therefore, the pair enumerations, table operations, and all candidate checks together require expected time

$$O\left(2^{\sum_{s=0}^{3} u_{s,t}} \left(1 + \log(2+p) + \lambda + \sum_{s=0}^{3} \bigl(u_{s,t} + \log(2 + |m_{s,t}|)\bigr)\right)^{O(1)}\right).$$

For space, $\mathcal{H}$ stores at most

$$2^{u_{0,t}+u_{3,t}}$$

pairs. The returned set $\mathcal{B}_t$ contains at most one candidate for each quadruple enumerated by the Cartesian product of the two pair spaces. Hence

$$|\mathcal{B}_t| \le 2^{\sum_{s=0}^{3} u_{s,t}}.$$

Every stored pair, quaternion candidate, and auxiliary integer has representation length polynomial in the input bit length. The returned set therefore dominates the table and the remaining workspace, giving space complexity

$$O\left(2^{\sum_{s=0}^{3} u_{s,t}} \left(1 + \log(2+p) + \lambda + \sum_{s=0}^{3} \bigl(u_{s,t} + \log(2 + |m_{s,t}|)\bigr)\right)^{O(1)}\right).$$

A.3 Proof of Lemma 12

The first step recovers $q_{\mathrm{rsp}}^{(t)}$ from the public key and the signature. As established by Section 4.2.2 of the main body, this recovery uses the verification procedure, Weil-pairing evaluations, and discrete logarithms in a group of power-of-two order. Its time and space complexities are polynomial in $\log p$ and e_{rsp}.

For every $0 \le s < 4$, the interval

$$\mathcal{I}_{s,t} = [A_{s,t}, A_{s,t} + 2^{u_{s,t}}) \cap \mathbb{Z}$$

contains exactly $2^{u_{s,t}}$ integers. Consequently, the first enumeration loop processes exactly

$$|\mathcal{I}_{0,t} \times \mathcal{I}_{3,t}| = 2^{u_{0,t}+u_{3,t}}$$

pairs before its parity test. The parity test can only discard pairs, so at most this many entries are inserted into H.

Similarly, the second enumeration loop processes exactly

$$|\mathcal{I}_{1,t} \times \mathcal{I}_{2,t}| = 2^{u_{1,t}+u_{2,t}}$$

pairs before its parity test, and each surviving pair performs one lookup in H.

Each execution of the innermost loop is uniquely determined by one pair from $\mathcal{I}_{0,t} \times \mathcal{I}_{3,t}$ and one pair from $\mathcal{I}_{1,t} \times \mathcal{I}_{2,t}$. Therefore, the total number of executions of the innermost loop is at most

$$\begin{aligned} |\mathcal{I}_{0,t} \times \mathcal{I}_{3,t}| \cdot |\mathcal{I}_{1,t} \times \mathcal{I}_{2,t}| &= 2^{u_{0,t}+u_{3,t}} 2^{u_{1,t}+u_{2,t}} \\ &= 2^{\sum_{s=0}^{3} u_{s,t}}. \end{aligned}$$

Each such execution constructs one $\widehat{\alpha}$, computes one greatest common divisor, tests membership in $\mathcal{O}_0 \setminus \{0\}$, and possibly inserts the candidate into $\mathcal{B}_t$. All these operations have bit complexity polynomial in

$$1 + \log(2+p) + e_{\mathrm{rsp}} + \sum_{s=0}^{3} \bigl(u_{s,t} + \log(2 + |m_{s,t}|)\bigr),$$

which bounds the bit lengths of all input-derived integers and candidate representations.

Since

$$2^{u_{0,t}+u_{3,t}} \le 2^{\sum_{s=0}^{3} u_{s,t}},$$
$$2^{u_{1,t}+u_{2,t}} \le 2^{\sum_{s=0}^{3} u_{s,t}},$$

the response-degree computation, the two pair enumerations, all table operations, and all candidate checks together require expected time

$$O\left(2^{\sum_{s=0}^{3} u_{s,t}} \left(1 + \log(2+p) + e_{\mathrm{rsp}} + \sum_{s=0}^{3} \left(u_{s,t} + \log(2 + |m_{s,t}|)\right)\right)^{O(1)}\right).$$

For space, the table H stores at most

$$2^{u_{0,t}+u_{3,t}}$$

pairs. The returned set $\mathcal{B}_t$ contains at most one candidate for each quadruple in the Cartesian product of the two input pair spaces. Hence

$$|\mathcal{B}_t| \le 2^{\sum_{s=0}^{3} u_{s,t}}.$$

The response-degree recovery requires only polynomial space, and each stored pair and quaternion candidate has polynomial representation length. Thus the returned set dominates the table and all remaining workspace, yielding space complexity

$$O\left(2^{\sum_{s=0}^{3} u_{s,t}} \left(1 + \log(2+p) + e_{\mathrm{rsp}} + \sum_{s=0}^{3} \left(u_{s,t} + \log(2 + |m_{s,t}|)\right)\right)^{O(1)}\right).$$

B Listings in Section 5

```
/* W1: challenge-dependent matrix arithmetic */
(c1, c2) = M_sk * (1, chl)
/* r_chl is derived from (c1, c2) */

/* W2: secret-dependent CRT subtraction */
delta = r_sk - r_chl
/* W3: CRT multiplication and recombination */
k = (delta * u) mod N_sk
r = r_chl + 2^192 * k
```

Listing 1: Schematic data flow of the measured matrix and CRT arithmetic (not a verbatim source-code excerpt). The CRT branch is shown for one coordinate; r_{sk} is x_{sk} or y_{sk}.

```
// setter = 2 b->x + (2 b->y + 1) i
ibz_add(&setter.v[0], &b->x, &b->x);
ibz_add(&setter.v[1], &b->y, &b->y);
ibz_add(&setter.v[1], &setter.v[1], &ibz_const_one);
// tmp = (u - v i)^2 mod 2 N2
ibz_copy(&tmp.v[0], &(split_a->coord).v[0]);
ibz_neg(&tmp.v[1], &(split_a->coord).v[1]);
ibz_add(&t, &b->norm, &b->norm);
ibz_vec_2_gaussian_mul(&tmp, &tmp, &tmp);
ibz_mod(&tmp.v[0], &tmp.v[0], &t);
ibz_mod(&tmp.v[1], &tmp.v[1], &t);
/* W1: Gaussian multiply, setter is a multiplicand */
ibz_vec_2_gaussian_mul(&tmp, &setter, &tmp);
ibz_mod(&tmp.v[0], &tmp.v[0], &t);
```

```
15 ibz_mod(&tmp.v[1], &tmp.v[1], &t);
16 ibz_mul(&u2, &split_a->coord.v[0], &split_a->coord.v[0]);
17 ibz_mul(&v2, &split_a->coord.v[1], &split_a->coord.v[1]);
18 ibz_add(&t, &u2, &v2);
19 // t = (u^2 + v^2)^{-1} mod N2
20 ibz_invmod(&t, &t, &b->norm);
21 ibz_div_2exp(&x, &tmp.v[0], 1); // x_pre
22 /* W2: coordinate-forming multiplies */
23 // X2 = x_pre * t mod N2
24 ibz_mul(&x, &x, &t);
25 ibz_mod(&x, &x, &b->norm);
26 // Y2 = ((Y2 - u^2 - v^2) / 2) * t mod N2
27 ibz_sub(&y, &tmp.v[1], &u2);
28 ibz_sub(&y, &y, &v2);
29 ibz_div_2exp(&y, &y, 1);
30 ibz_mul(&y, &y, &t);
31 ibz_mod(&y, &y, &b->norm);
32 /* W3: CRT recombination, materialization of (x, y) */
33 ibz_copy(&x2, &a->x);
34 ibz_copy(&y2, &a->y);
35 // x = CRT([x1, X2], [N1, N2]), y = CRT([y1, Y2], [N1, N2])
36 ibz_crt(&x, &x2, &x, &a->norm, &b->norm);
37 ibz_crt(&y, &y2, &y, &a->norm, &b->norm);
38 ibz_mul(&r, &a->norm, &b->norm);
39 quat_xyn_to_inert_ideal(intersection, &x, &y, &r);
```

Listing 2: Traced tail of `quat_ideal_intersect_00` (`src/ideal.c` of the reference implementation; initializations and assertions omitted). `a` is the operand with split generator `split_a` $= u + vi$ and `b` the operand in inert representation (`b->x`, `b->y`, `b->norm` $= N_2$) whose coordinates are secret-dependent. The comments W1–W3 mark the regions analysed in the text: the Gaussian multiply (lines 13–15), the coordinate-forming multiplies (lines 23–31), and the CRT recombination with materialization of (x, y) (lines 33–39).

```
1  // s = c^2 + d^2 mod n
2  ibz_mul(&tmp, &tmp_gen.coord.v[2], &tmp_gen.coord.v[2]);
3  ibz_mul(&r, &tmp_gen.coord.v[3], &tmp_gen.coord.v[3]);
4  ibz_add(&tmp, &tmp, &r);
5  ibz_mod(&s, &tmp, norm);
6  // t = (2s)^{-1} mod n
7  ibz_add(&tmp, &s, &s);
8  ibz_invmod(&t, &tmp, norm);
9  /* acquisition window opens */
10 // (a + b i)(c - d i) = (ac + bd) + (bc - ad) i
11 ibz_neg(&tmp_gen.coord.v[3], &tmp_gen.coord.v[3]);
12 ibz_vec_2_gaussian_mul(&tmp_g, (ibz_vec_2_t *)&tmp_gen.coord.v[0],
13                            (ibz_vec_2_t *)&tmp_gen.coord.v[2]);
14 ibz_neg(&tmp_gen.coord.v[3], &tmp_gen.coord.v[3]);
15 ibz_mod(&tmp_g.v[0], &tmp_g.v[0], norm);
16 ibz_mod(&tmp_g.v[1], &tmp_g.v[1], norm);
17 // x = (ac + bd) t mod n
18 ibz_mul(&tmp, &tmp_g.v[0], &t);
19 ibz_mod(&x, &tmp, norm);
20 // y = (bc - ad - s) t mod n
21 ibz_sub(&tmp, &tmp_g.v[1], &s);
22 ibz_mod(&tmp, &tmp, norm);
23 ibz_mul(&tmp, &tmp, &t);
24 ibz_mod(&y, &tmp, norm);
25 quat_xyn_to_inert_ideal(ideal, &x, &y, norm);
26 /* acquisition window closes */
```

Listing 3: Traced tail of `quat_ideal_create_00_inert_odd` (`src/ideal.c`), the routine that turns the accepted generator into the inert representation $(n, (x, y))$ of J_t inside EquivalentCoprimeIdeal. `tmp_gen.coord.v[0..3]` hold the coordinates a, b, c, d of γ' reduced modulo `norm` $= n$. The acquisition window is lines 10–25, the region between the

two comments. The routine stores $x' = (ac + bd)\, t \bmod n$ and $y' = (bc - ad - s)\, t \bmod n$; the coordinates $x = 2x'$ and $y = 1 + 2y'$ quoted in the text are formed when the generator is reconstituted (`quat_ideal_odd_inert_gen`).

```
1  /* Algorithm 3.18, line 2 (protocol.c) */
2  quat_alg_conj(sk_chall_quat, sk_chall_quat);
3
4  /* quat_alg_conj (algebra.c), wrapper body */
5  ibz_copy(&conj->denom, &x->denom);
6  ibz_copy(&conj->coord.v[0], &x->coord.v[0]);
7  for (int i = 1; i < 4; i++)
8      ibz_neg(&conj->coord.v[i], &x->coord.v[i]);
9
10 /* ibz_neg (mp.c), the limb-wise negation */
11 void ibz_neg(ibz_t *neg, const ibz_t *a) {
12     int n = NUM_LIMBS(a->bitlen);
13     digit_t carry = 1;
14     for (int i = 0; i < n; i++) {
15         ddigit_t t = (ddigit_t)~a->limbs[i] + carry;
16         neg->limbs[i] = t;
17         carry = t >> NUM_BITS_LIMB;
18     }
19     neg->bitlen = a->bitlen;
20 }
```

Listing 4: Traced conjugation call (`protocol.c`, line 2), the `quat_alg_conj` wrapper (`algebra.c`, lines 5–8) and the limb-wise negation in `ibz_neg` (`mp.c`, lines 11–20). The GPIO trigger is raised before line 5 and lowered after line 8, so the acquisition window is the whole wrapper body: one denominator copy, one real-coordinate copy and three negations. Lines 15–16 are the load and store that carry the secret limb; they execute 13 times per coordinate.

```
1  /* quat_lattice_sample_from_ball (src/lattice.c) */
2  do {
3      /* sample x and map it into the parallelogram */
4      ibz_mat_4x4_eval_t(&x, &x, &U);
5      quat_qf_eval(&tmp, &G, &x);
6  } while ((ibz_get(&tmp) % 4 == 0) || (ibz_cmp(&tmp, &rad) > 0));
7
8  ibz_mat_4x4_eval(&(res->coord), &(lat.basis), &x);
9  ibz_copy(&(res->denom), &(lat.denom));
10 quat_alg_normalize(res);
11
12 /* ibz_mat_4x4_eval (src/dim4.c), the callee invoked above */
13 {
14     for (int i = 0; i < 4; i++)
15         ibz_set(&sum.v[i], 0, 0);
16     for (int i = 0; i < 4; i++)
17         for (int j = 0; j < 4; j++) {
18             ibz_mul(&prod, &mat->m[i][j], &vec->v[j]);
19             ibz_add(&(sum.v[i]), &(sum.v[i]), &prod);
20         }
21     ibz_vec_4_copy(res, &sum);
22 }
```

Listing 5: Traced tail of `quat_lattice_sample_from_ball` (`src/lattice.c`, lines 1–10) together with the callee `ibz_mat_4x4_eval` (`src/dim4.c`, lines 12–22) invoked at line 8. The GPIO trigger brackets line 21 and is raised once per coordinate, giving four pulses per trace. The accumulation at lines 16–20 costs 39,085 cycles, which exceeds the capture window at 4 ns/sample and is therefore not traced; the normalization at line 10 follows the window, so the traced values are the evaluation output before the gcd is divided out.