# Qlapoti+ and More: Optimizing Isogeny-based Signatures

Yi-Fu Lai

Shanghai Jiao Tong University

**Abstract**

This paper presents several optimizations to Qlapoti (Asiacrypt'25), an ideal-finding procedure at the heart of modern isogeny-based signature schemes. We apply these optimizations to the Qlapoti-based NIST Round-2 SQIsign implementation from Asiacrypt'25. Together, they accelerate the Qlapoti procedure by approximately 1.6× to 5.3×, depending on the parameter set and implementation.

Under the Broadwell benchmark, compared with the baseline implementation in Asiacrypt'25, our optimizations achieve key-generation speedups of 1.29×, 2.23×, and 1.54×, and signing speedups of 1.23×, 1.79×, and 1.43×, at NIST security levels 1, 3, and 5, respectively.

Our techniques also apply to the Qlapoti-optimized PRISM implementation (PKC'25, Journal of Cryptology), for which we introduce an additional tailored optimizations. Under the Broadwell benchmark, compared with the baseline implementation in JoC using Qlapoti, our improvements translate into key-generation speedups of 1.22×, 1.87×, and 1.46×, and signing speedups of 1.47×, 1.90×, and 1.60×, at NIST security levels 1, 3, and 5, respectively.

## 1 Introduction

Isogeny-based signatures provide exceptionally compact public keys and signatures based on assumptions that are fundamentally different from those underlying lattice-based cryptography. SQIsign [DKL+20], which has advanced to the third round of the NIST additional digital-signature process [AAA+24], is a prominent example of this approach. Recent constructions based on higher-dimensional isogenies, including SQIsign2D-West [BDD+24] and PRISM [BBC+25], further improve the efficiency and simplicity of isogeny-based signatures. Nevertheless, their key generation and signing procedures remain considerably more expensive than those of many other post-quantum signature schemes.

A key operation in these schemes is the effective Deuring correspondence of the ideal-to-isogeny translation: given a quaternion ideal, one must compute the corresponding isogeny between supersingular elliptic curves. Although the Deuring correspondence gives a conceptual relation between quaternion ideals and isogenies, making this relation algorithmically efficient has long been one of the principal challenges in isogeny-based cryptography. Earlier approaches relied on constrained norm equations and the KLPT algorithm, resulting in complicated and relatively expensive translations.

Higher-dimensional isogenies provide a different approach. QFESTA showed how an endomorphism of suitable degree can be split through a two-dimensional isogeny [NO24]. The Clapoti framework of Page and Robert subsequently used this technique to translate ideals into isogenies [PR23]. In the quaternion setting, the resulting ideal-to-isogeny algorithm searches for two ideals $I_1, I_2$ equivalent to the input ideal and positive integers $u, v$ satisfying

$$u\mathsf{nrd}(I_1) + v\mathsf{nrd}(I_2) = 2^e.$$

The coefficients $u$ and $v$ require the construction of two auxiliary isogenies in addition to the main two-dimensional isogeny. Moreover, finding suitable ideals usually has a non-negligible failure probability, which must be mitigated by costly rerandomization and precomputation.

Qlapoti [BCE+25] substantially improves this translation by solving the more restrictive equation

$$\mathsf{nrd}(I_1) + \mathsf{nrd}(I_2) = 2^e$$

directly. Consequently, the auxiliary isogenies corresponding to $u$ and $v$ are no longer required, and the translation can be performed using a single chain of two-dimensional isogenies. Qlapoti also achieves a negligible failure probability, without the complex rerandomization procedure used by the previous implementation. The original work reports an acceleration of the ideal-to-isogeny translation by a factor between 2.2 and 2.6, together with substantial reductions in memory consumption. As a result, it gives a substantial improvement in SQIsign2D-West-type SQIsign family (e.g. the round-2 SQIsign) [BCE+25] and the salted version [BBC+26] of PRISM [BBC+25].

**Our contributions.** In this work, we revisit the Qlapoti procedure and introduce four optimizations, which we call Qlapoti+, unleashing its true power. Additionally, we give an additional optimizations in the ideal-to-isogeny procedure in salt-PRISM [BBC+26] out of the Qlapoti+ procedure. We summarize our contributions as follow.

1. In the Broadwell build, Qlapoti+ is 1.87×, 5.33×, and 4.00× faster than Qlapoti at NIST security levels 1, 3, and 5, respectively.

2. By applying Qlapoti+ to the Qlapoti-optimized NIST Round-2 SQIsign, under the Broadwell benchmark, these improvements translate into key-generation speedups of 1.29×, 2.23×, and 1.54×, and signing speedups of 1.23×, 1.79×, and 1.43×, at the three security levels, respectively.

3. We also apply Qlapoti+ to salt-PRISM [BBC+26] together with the additional tailored optimizations in Sec. B. These improvements translate into key-generation speedups of 1.22×, 1.87×, and 1.46×, and signing speedups of 1.47×, 1.90×, and 1.60×, at the three security levels, respectively.

**Remark 1.1.** We are aware of the recent attack [Wes26], which affects the parameter sets previously selected for both SQIsign and PRISM in the NIST security standard. Since the official SQIsign team has not yet proposed updated parameter sets, we continue to use the previously proposed ones. Nevertheless, our improvement is generic and applies independently of the choice and size of the parameters.

**Technical Overview.**

1. **Optimal-two-representative exploitation.** A single Cornacchia output gives several representations of the same sum-of-two-squares equation. In particular, when the input is composite with $n$ distinct prime divisors, there are at most $8^n$ valid representatives considering the sign-changing, swapping, and the Gaussian factorization. We choose two optimal representatives among them for the integral-$\theta$ branch inside Qlapoti. This substantially reduces the number of complete Qlapoti iterations caused by failures of the final output test, which occurs at the very end of the whole procedure.

2. **Quick solvability test.** In extended Cornacchia, trial division leaves a residual factor that is normally passed directly to a probable-prime test. We observe that, in the relevant sum-of-two-squares case, a residual factor congruent to 3 mod 4 cannot be represented. Such candidates can therefore be rejected using a very inexpensive modular test before invoking the expensive probable-prime test.

3. **Faster normalization.** The baseline implementation constructs successive multiples of a generator incrementally and applies a generic quaternion normalization routine to each multiple. We normalize the base generator $\alpha_0$ once and specialize the normalization of $\lambda\alpha_0$. If $d$ is the denominator of the normalized representation of $\alpha_0$, the common factor to be removed is simply $\mathsf{gcd}(\lambda, d)$. This replaces a general multiple-coordinate gcd computation with a single small gcd.

4. **Cache-and-Multiply Inverse Update and Fast Small Inversion.** The optimization has two stages. For every multiplier $\lambda$, the baseline recomputes $(2\lambda a_0)^{-1} \bmod n$. We instead first cache $u_0 = (2a_0)^{-1} \bmod n$ for each generator and obtain the required inverse as $u = u_0\lambda^{-1} \bmod n$. This replaces the repeated inversion of a full-size coordinate with the inversion of the typically small multiplier $\lambda$, followed by one modular multiplication. Second we further use and optimize the Arazi Lemma [JP03] to obtain the faster small inversion over a large modulo.

# 2 Preliminaries

## 2.1 Quaternion Algebra

We briefly recall the basic quaternion-ideal setting considered in this work. Let $p \equiv -1 \pmod 8$ be the characteristic prime, and let $B_{p,\infty} = \mathbb{Q} + \mathbb{Q}i + \mathbb{Q}j + \mathbb{Q}k$, where $i^2 = -1$, $j^2 = -p$, and $k = ij = -ji$. For $\gamma = a + bi + cj + dk$, its quaternion conjugate and reduced norm are $\overline{\gamma} = a - bi - cj - dk$ and $\mathsf{nrd}(\gamma) = \gamma\overline{\gamma} = a^2 + b^2 + p(c^2 + d^2)$, respectively.

Let $E_0$ be the base supersingular elliptic curve, and fix the identification $O_0 \simeq \mathrm{End}(E_0)$, where

$$O_0 = \mathbb{Z}\left\langle 1,\, i,\, \frac{i+j}{2},\, \frac{1+k}{2} \right\rangle.$$

We write $\iota \in \mathrm{End}(E_0)$ for the endomorphism corresponding to $i \in O_0$.

For a left $O_0$-ideal $I$, its reduced norm is denoted by $\mathsf{nrd}(I)$. Two left $O_0$-ideals $I$ and $J$ are equivalent if $I = J\gamma$ for some $\gamma \in B_{p,\infty}^{\times}$. Let $I$ have norm $n = \mathsf{nrd}(I)$. For non-zero $\beta \in I$, [BCE+25] considers the equivalent ideal $\chi_I(\beta) := I\overline{\beta}/n$, whose norm is $\mathsf{nrd}(\chi_I(\beta)) = \mathsf{nrd}(\beta)/n$.

For a target exponent $e$, suppose that $\beta_1, \beta_2 \in I$ satisfy $\mathsf{nrd}(\beta_1) + \mathsf{nrd}(\beta_2) = 2^e n$. Then

$$I_\ell = \chi_I(\beta_\ell) \quad (\ell \in \{1,2\}), \qquad \mathsf{nrd}(I_1)+\mathsf{nrd}(I_2) = 2^e, \qquad \theta = \frac{\beta_2\overline{\beta_1}}{n}, \qquad \mathsf{nrd}(\theta) = \mathsf{nrd}(I_1)\mathsf{nrd}(I_2).$$

## 2.2 Qlapoti Procedure

We recall the Qlapoti procedure of [BCE+25]. Algorithm 1 follows the reference C implementation and makes explicit several checks omitted from the high-level description in [BCE+25, Algorithm 2]. In particular, the checks specified in Algorithms 2 to 4 are relevant to our optimizations. The steps targeted by our optimizations are highlighted in red.

At a high level, Qlapoti takes a left $O_0$-ideal $J$ and returns two ideals $I_1$ and $I_2$, both equivalent to $J$, whose norms are compatible with a 2-power isogeny computation. It first replaces $J$ by a small equivalent ideal $I$ of norm $n$ and samples a small generator $\alpha \in I$. It then searches for $\beta_1, \beta_2 \in I$ such that $I_1 = \chi_I(\beta_1)$ and $I_2 = \chi_I(\beta_2)$ satisfy $\mathsf{nrd}(I_1) + \mathsf{nrd}(I_2) = 2^e$.

Restricting the unknown quaternion coefficients to $\mathbb{Z}[i]$ reduces the search first to finding a short solution $(s, t)$ to

$$2a_\alpha s + 2b_\alpha t \equiv 2^e - 2r \pmod n,$$

and then to solving

$$z_0^2 + z_1^2 = z.$$

The first problem is a two-dimensional approximate closest-vector problem and is handled using Babai rounding [Bab86]. The second is solved using extended Cornacchia, where $z$ need not be prime. The resulting values determine $\beta_1$ and $\beta_2$, which are then subjected to the final output test.

For a quaternion represented as $\eta = (q_0 + q_1 i + q_2 j + q_3 k)/d$, where $q_0, q_1, q_2, q_3, d \in \mathbb{Z}$ and $d > 0$, the routine Normalize divides the four numerator coordinates and $d$ by their common gcd. The remaining positive value of $d$ is called the normalized denominator of $\eta$. The predicate $\mathsf{OddNorm}(\eta)$ returns true exactly when $\mathsf{nrd}(\eta)$ is odd.

The final test is implemented by PrepareQlapotiOutput in Algorithm 3. It first normalizes $\theta = \beta_2\overline{\beta_1}/n$. If the normalized denominator is 2, the pair is accepted only when $\theta$ has odd reduced norm, as required by the subsequent two-dimensional isogeny routine. If the normalized denominator is 1, the routine checks whether the initial $(2, 2)$-step is one of the two supported product endomorphisms recognized by EndType. In that case, it removes the initial step using the corresponding half-sum/half-difference transformation. Otherwise, the candidate is rejected.

**Algorithm 1:** $\mathsf{QlapotiFull}(J)$: baseline implementation

**Input :** A left $O_0$-ideal $J$ and a target exponent $e$

**Output:** $\beta_1, \beta_2$, equivalent ideals $I_1 = \chi_I(\beta_1)$, $I_2 = \chi_I(\beta_2)$, $\theta = \beta_2\overline{\beta_1}/n$, and $\delta \in \{0, 1\}$, such that

$$\mathsf{nrd}(I_1) + \mathsf{nrd}(I_2) = 2^{e-\delta},$$

where $\delta = 1$ exactly when an initial product endomorphism has been removed

$I \leftarrow \mathsf{SmallestEquivIdeal}(J)$, $n \leftarrow \mathsf{nrd}(I)$
**while** true **do**
  $\alpha_0 \leftarrow \mathsf{SmallGenerator}(I)$, and write $\alpha_0 = a_0 + b_0 i + c_0 j + d_0 k$, where $a_0, b_0, c_0, d_0 \in \frac{1}{2}\mathbb{Z}$
  **if** $\gcd(2a_0, n) \neq 1$ **then continue**
  $r_0 \leftarrow \mathsf{nrd}(\alpha_0)/n$, and compute a reduced basis of $\mathcal{L}_{\alpha_0} = \{(x, y) \in \mathbb{Z}^2 : 2a_0x + 2b_0y \equiv 0 \pmod{n}\}$
  $\lambda \leftarrow 0$
  **while** true **do**
    $\lambda \leftarrow \min\{\ell > \lambda : \gcd(\ell, n) = 1\}$
    $\alpha \leftarrow \lambda\alpha_0$
    $a_\alpha \leftarrow \lambda a_0$, $b_\alpha \leftarrow \lambda b_0$, $r \leftarrow \lambda^2 r_0$
    **if** $2^e - 2r \leq 0$ **then continue**
    $u \leftarrow (2a_\alpha)^{-1} \bmod n$
    Using the reduced basis of $\mathcal{L}_{\alpha_0}$, apply Babai rounding to obtain a short pair $(s, t)$ satisfying $(2a_\alpha s + 2b_\alpha t)u \equiv (2^e - 2r)u \pmod{n}$
    $z \leftarrow 2(2^e - 2r - 2a_\alpha s - 2b_\alpha t)/n - s^2 - t^2$
    **if** $\neg\mathsf{ModCondition}(z, s, t)$ **then continue**
    $(z_0, z_1) \leftarrow \mathsf{Cornacchia}(z)$
    **if** $(z_0, z_1) = \perp$ **then continue**
    **if** $z_0 \not\equiv s \pmod 2$ **then**
      Swap $z_0, z_1$
    $(\widehat{z}_0, \widehat{z}_1) \leftarrow (z_0, z_1)$
    $a_1 \leftarrow (\widehat{z}_0 + s)/2$, $b_1 \leftarrow (\widehat{z}_1 + t)/2$, $a_2 \leftarrow s - a_1$, $b_2 \leftarrow t - b_1$
    $\beta_1 \leftarrow n(a_1 + b_1 i) + \alpha$, $\beta_2 \leftarrow n(a_2 + b_2 i) + \alpha$
    $R \leftarrow \mathsf{PrepareQlapotiOutput}(\beta_1, \beta_2, n)$
    **if** $R = \perp$ **then break**
    $(\beta_1', \beta_2', \theta, \delta) \leftarrow R$
    $I_1 \leftarrow \chi_I(\beta_1')$, $I_2 \leftarrow \chi_I(\beta_2')$
    **return** $\beta_1', \beta_2', I_1, I_2, \theta, \delta$

**Algorithm 2:** ModCondition$(z, s, t)$

**Input :** Integers $z, s, t$
**Output:** true if the parity constraints are compatible
`// The following line is an extra restriction in the current C implementation.`

**if** $z \equiv 0 \pmod 8$ **then**
  **return** false
**if** $z \equiv 0 \pmod 4$ **then**
  **return** $s \equiv 0 \pmod 2$ ***and*** $t \equiv 0 \pmod 2$
**else if** $z \equiv 1 \pmod 4$ **then**
  **return** $s \not\equiv t \pmod 2$
**else if** $z \equiv 2 \pmod 4$ **then**
  **return** $s \equiv 1 \pmod 2$ ***and*** $t \equiv 1 \pmod 2$
**else**
  **return** false

**Algorithm 3:** PrepareQlapotiOutput$(\beta_1, \beta_2, n)$

**Input :** Elements $\beta_1, \beta_2 \in I$ and $n = \mathsf{nrd}(I)$
**Output:** An admissible tuple $(\beta_1', \beta_2', \theta, \delta)$, where $\delta \in \{0, 1\}$, or $\perp$

$\theta \leftarrow \mathsf{Normalize}(\beta_2\overline{\beta_1}/n)$
**if** $\theta$ *has normalized denominator* 2 **then**
  **if** $\neg\mathsf{OddNorm}(\theta)$ **then**
    **return** $\perp$
  **return** $(\beta_1, \beta_2, \theta, 0)$
**if** $\theta$ *does not have normalized denominator* 1 **then**
  **return** $\perp$
$\tau \leftarrow \mathsf{EndType}(\theta)$
**if** $\tau = 1$ **then**
  $\beta_1' \leftarrow (\beta_1 - \beta_2)/2$
  $\beta_2' \leftarrow (\beta_1 + \beta_2)/2$
**else if** $\tau = 2$ **then**
  $\widetilde{\beta}_2 \leftarrow i\beta_2$
  $\beta_1' \leftarrow (\beta_1 - \widetilde{\beta}_2)/2$
  $\beta_2' \leftarrow (\beta_1 + \widetilde{\beta}_2)/2$
**else**
  **return** $\perp$
$\theta' \leftarrow \mathsf{Normalize}(\beta_2'\overline{\beta_1'}/n)$
**assert** that $\theta'$ has normalized denominator 2
**return** $(\beta_1', \beta_2', \theta', 1)$

**Algorithm 4:** $\mathsf{EndType}(\theta)$

**Input :** A normalized integral quaternion element $\theta$
**Output:** A type $\tau \in \{0, 1, 2\}$, where 1 denotes the identity-graph type endomorphism, 2 denotes the $i$-graph type endomorphism, and 0 denotes that neither supported type is recognized

$(c_0, c_1, c_2, c_3) \leftarrow$ the coordinates of $\theta$ in the basis $(1, i, j, k)$
$(t_1, t_2, t_3, t_4) \leftarrow (c_0, c_1, c_3, c_2) \bmod 4$
$\mathcal{T}_1 \leftarrow \{(2,0,1,2), (2,0,3,2), (2,2,1,0), (2,2,3,0)\}$
`// An error in the Qlapoti code that` $(2,2,3,0)$ `was replaced by` $(2,2,3,2)$
$\mathcal{T}_2 \leftarrow \{(0,2,2,1), (0,2,2,3), (2,2,0,1), (2,2,0,3)\}$
**if** $(t_1, t_2, t_3, t_4) \in \mathcal{T}_1$ **then**
    **return** 1
**if** $(t_1, t_2, t_3, t_4) \in \mathcal{T}_2$ **then**
    **return** 2
**return** 0

### 2.3 Qlapoti and Its Application to Isogeny-Based Signatures

Qlapoti is used within the ideal-to-isogeny translation of both SQIsign and PRISM. In these schemes, the secret key is generated and stored as a left $O_0$-ideal $I_{\mathsf{sk}}$. During key generation, $I_{\mathsf{sk}}$ is translated into an isogeny

$$\varphi_{\mathsf{sk}} \colon E_0 \longrightarrow E_{\mathsf{vk}},$$

and the codomain curve $E_{\mathsf{vk}}$ forms the main component of the public key. Each invocation of the ideal-to-isogeny translation uses Qlapoti to find equivalent ideals whose norms satisfy the conditions required by the subsequent two-dimensional isogeny computation.

Regarding the signing algorithms, SQIsign is obtained by applying the Fiat–Shamir transform to a $\Sigma$-protocol. During signing, the signer first samples a commitment ideal $I_{\mathsf{com}}$ and translates it into an isogeny $\psi_{\mathsf{com}} \colon E_0 \to E_{\mathsf{com}}$. The message and the commitment determine a challenge isogeny originating from the public-key curve. The signer combines this challenge with the secret ideal and the commitment isogeny to obtain an ideal corresponding to the response isogeny. After replacing this ideal by an equivalent ideal of suitable norm, the signer computes an efficient representation of the response. This representation also requires an auxiliary isogeny obtained by translating another ideal of a prescribed norm. Thus, in addition to its use during key generation, the SQIsign signing procedure invokes the ideal-to-isogeny translation for both the commitment and the auxiliary isogeny.

PRISM [BBC+25] follows a hash-and-sign construction. Given a message, the signing algorithm hashes it to a prime $q$ and samples a left ideal $I_\sigma$ of norm $q(2^a - q)$ in the right order of $I_{\mathsf{sk}}$. It then translates the product $I_\sigma I_{\mathsf{sk}}$ into an isogeny

$$\sigma \colon E_{\mathsf{vk}} \longrightarrow E_{\mathsf{sig}}$$

of degree $q(2^a - q)$, and returns an efficient representation of $\sigma$ as the signature. The salted implementation considered in this work [BBC+26] uses the same ideal-to-isogeny translation and therefore also invokes Qlapoti during key generation and signing. It also naturally applies to [Ler25].

In both schemes, verification will not use Qlapoti. For a more detailed description of SQIsign, PRISM, and the role of ideal-to-isogeny translation in their signing procedures, we refer the reader to [BCE+25, Sec. 2.4].

# 3 Optimization

We now present main four optimizations to the Qlapoti procedure. Algorithm 5 summarizes the optimized procedure, with the modified steps highlighted in blue. Primarily,

1. Testing two representatives derived from the same Cornacchia output (Sec. 3.1).
2. Introducing a quick solvability test before the probable-prime test (Sec. 3.2).
3. Accelerating normalization by specializing it to elements of the form $\lambda\alpha_0$ (Sec. 3.3).
4. Trading-off a large modular inversion by caching a partial element and a multiplication with a smaller inversion (Sec. 3.4).

## 3.1 Optimal Two-Representaive Exploitation in (Extended) Cornacchia for Endtype

The optimal multi-representative optimization is shown in Lines 19 to 22 of Algorithm 5. To simplify the analysis later, we use four representaives here first, and then show two $(v_0, v_3)$ is optimal with respect to the rejection condition later.

**Original Implementation.** Given $z$, (extended) Cornacchia returns one representation

$$z = z_0^2 + z_1^2.$$

The coordinates are first exchanged when necessary so that $z_0 \equiv s \pmod 2$ and $z_1 \equiv t \pmod 2$ for integrality. The implementation then constructs

$$a_1 = \frac{z_0 + s}{2}, \qquad a_2 = \frac{s - z_0}{2}, \qquad b_1 = \frac{z_1 + t}{2}, \qquad b_2 = \frac{t - z_1}{2}. \tag{1}$$

and obtains

$$\beta_1 = n(a_1 + b_1 i) + \alpha, \qquad \beta_2 = n(a_2 + b_2 i) + \alpha. \tag{2}$$

This single pair is passed to $\mathsf{PrepareQlapotiOutput}(\beta_1, \beta_2, n)$. If the procedure returns $\perp$, the current generator is discarded and Qlapoti restarts with a new one.

This rejection is a major source of failure in practice. In our experiments, PrepareQlapotiOutput rejects approximately 87.5% of the pairs obtained from the single Cornacchia representation. The rejection is expensive as it leads to whole procedure of Algorithm 1 has to be restarted.

**Optimization.** It is not hard to observe that to reuse the sign-and-swap representatives of the Cornacchia output computed at Line 16 at negligible additional cost. For a representation $(z_0, z_1)$ from Line 16 satisfying $z = z_0^2 + z_1^2$, these representatives preserve the sum-of-two-squares equation and the parity conditions used in Eq. (1); hence they produce valid pairs through Eq. (2) without repeating the preceding lattice, CVP, or Cornacchia computations.

The four independent sign for choices for $z_0, z_1$ form the first four valid Cornacchia solutions. In fact, the four solutions form two conjugate pairs. As proved in Lemma 3.1, the two members of each pair have the same acceptance outcome under PrepareQlapotiOutput. It therefore suffices to test

$$(z_0, z_1) \qquad \text{and} \qquad (z_0, -z_1).$$

The other two solutions come from swapping the representatives. In that case, two further acceptance-distinct natural representatives are tested:

$$(z_1, z_0) \qquad \text{and} \qquad (z_1, -z_0).$$

**Algorithm 5:** $\mathsf{QlapotiPlusFull}(J)$: optimized implementation

**Input :** A left $O_0$-ideal $J$ and a target exponent $e$

**Output:** $\beta_1, \beta_2$, equivalent ideals $I_1 = \chi_I(\beta_1)$, $I_2 = \chi_I(\beta_2)$, $\theta = \beta_2\overline{\beta_1}/n$, and $\delta \in \{0, 1\}$, such that $\mathsf{nrd}(I_1) + \mathsf{nrd}(I_2) = 2^{e-\delta}$, where $\delta = 1$ exactly when the recognized initial product endomorphism has been removed

$I \leftarrow \mathsf{SmallestEquivIdeal}(J)$, $n \leftarrow \mathsf{nrd}(I)$
**while** true **do**
  $\alpha_0 \leftarrow \mathsf{SmallGenerator}(I)$, and write $\alpha_0 = a_0 + b_0 i + c_0 j + d_0 k$, where $a_0, b_0, c_0, d_0 \in \frac{1}{2}\mathbb{Z}$
  **if** $\gcd(2a_0, n) \neq 1$ **then continue**
  $r_0 \leftarrow \mathsf{nrd}(\alpha_0)/n$, and compute a reduced basis of $\mathcal{L}_{\alpha_0} = \{(x, y) \in \mathbb{Z}^2 : 2a_0 x + 2b_0 y \equiv 0 \pmod{n}\}$
  $u_0 \leftarrow (2a_0)^{-1} \bmod n$
  $\lambda \leftarrow 0$
  **while** true **do**
    $\lambda \leftarrow \min\{\ell > \lambda : \gcd(\ell, n) = 1\}$
    $\alpha \leftarrow \lambda\alpha_0$, $a_\alpha \leftarrow \lambda a_0$, $b_\alpha \leftarrow \lambda b_0$, $r \leftarrow \lambda^2 r_0$
    **if** $2^e - 2r \leq 0$ **then continue**
    $u \leftarrow u_0\lambda^{-1} \bmod n$
    Using the reduced basis of $\mathcal{L}_{\alpha_0}$, apply the simplified Babai rounding to obtain a short pair $(s, t)$ satisfying $(2a_\alpha s + 2b_\alpha t)u \equiv (2^e - 2r)u \pmod{n}$
    $z \leftarrow 2(2^e - 2r - 2a_\alpha s - 2b_\alpha t)/n - s^2 - t^2$
    **if** $z \leq 0$ ***or*** $\neg\mathsf{ModCondition}(z, s, t)$ **then continue**
    $(z_0, z_1) \leftarrow \mathsf{Cornacchia}(z)$
    **if** $(z_0, z_1) = \perp$ **then continue**
    **if** $z_0 \not\equiv s \pmod 2$ **then** Swap $z_0, z_1$
    $\mathcal{R}_{\text{eff}} \leftarrow \{v_0 = (z_0, z_1), v_1 = (z_0, -z_1)\}$
    **if** $s \equiv t \pmod 2$ **then**
      $\mathcal{R}_{\text{eff}} \leftarrow \mathcal{R}_{\text{eff}} \cup \{v_2 = (z_1, z_0), v_3 = (z_1, -z_0)\}$
      `// We will show it suffices to consider` $R_{\text{eff}} = \{v_0, v_3\}$ `eventually.`
    **foreach** $(\widehat{z}_0, \widehat{z}_1) \in \mathcal{R}_{\text{eff}}$ **do**
      $a_1 \leftarrow (\widehat{z}_0 + s)/2$, $b_1 \leftarrow (\widehat{z}_1 + t)/2$, $a_2 \leftarrow s - a_1$, $b_2 \leftarrow t - b_1$
      $\beta_1 \leftarrow n(a_1 + b_1 i) + \alpha$, $\beta_2 \leftarrow n(a_2 + b_2 i) + \alpha$
      $R \leftarrow \mathsf{PrepareQlapotiOutput}(\beta_1, \beta_2, n)$
      **if** $R \neq \perp$ **then**
        $(\beta_1', \beta_2', \theta, \delta) \leftarrow R$, $I_1 \leftarrow \chi_I(\beta_1')$, $I_2 \leftarrow \chi_I(\beta_2')$
        **return** $\beta_1', \beta_2', I_1, I_2, \theta, \delta$
      **break**

We remark that the swapping preserves the integrality of Eq. (1) only when $s \equiv t \pmod 2$ due to the parity condition between $z_0, z_1, z, t$.[1]

Moreover, we note that for the integral-$\theta$ branch, the extended Cornacchia will usually take a composite as input. If there are $\ell$ distinct prime divisors in the input, there are actually at most $8^n$ valid representatives as the output through the Gaussian factorization (i.e. if $p_i = (x_i + y_i i)$ for $i \in \{1, 2\}$, then $(x_1 + y_1 i)(x_2 + y_2 i) = (x + yi)$ for some $x, y$). The following sequence of Lemmas 3.1 and 3.5 to 3.7 will show that the four representatives (eventually two) considered above is the optimal choice.

Overall, the optimized implementation consequently invokes PrepareQlapotiOutput on two effective representatives in general and on at most four when swapping is parity-compatible. All representatives reuse the same $z$, $\alpha$, $s$, and $t$; only the inexpensive reconstruction of $(a_1, a_2, b_1, b_2, \beta_1, \beta_2)$ in Eqs. (1) and (2). We give a rigorous analysis in Sec. 3.1.1 to quantify the improved failure rate.

### 3.1.1 Multi-representative Choices Analysis

We analyze only the post-Cornacchia rejection caused by EndType in the integral branch of PrepareQlapotiOutput.

**Relation to non-gluing steps.** The EndType test has a geometric role in the two-dimensional 2-power walk represented by the pair $(\beta_1, \beta_2)$. In the integral-$\theta$ branch, PrepareQlapotiOutput attempts to recognize whether the initial $(2, 2)$-step is one of the two supported product-to-product endomorphisms of $E_0 \times E_0$:

$$\varepsilon_1(P, Q) = (P + Q, P - Q), \qquad \varepsilon_\iota(P, Q) = \big(\iota(P) + Q, \iota(P) - Q\big).$$

Indeed, if $(P, Q)$ lies in the kernel of $\varepsilon_1$, then $P + Q = P - Q = 0$, whence $Q = P$ and $2P = 0$; thus the kernel is $\{(P, P) : P \in E_0[2]\}$. Likewise, $\varepsilon_\iota(P, Q) = 0$ gives $Q = \iota(P)$ and $2\iota(P) = 0$, so its kernel is $\{(P, \iota(P)) : P \in E_0[2]\}$. On $E_0[2] \times E_0[2]$, their kernels are respectively the graphs of the identity and of the distinguished automorphism $\iota$. Their codomains therefore remain elliptic products, rather than becoming non-product principally polarized abelian surfaces. In this sense, both maps are initial *non-gluing* steps.

The (corrected) Algorithm 4 distinguishes these two possibilities. When EndType returns 1 or 2, PrepareQlapotiOutput will apply the corresponding transformation. This transformation algebraically removes the recognized product endomorphism from the beginning of the walk and reduces the remaining 2-power exponent by one.

On the other side, EndType$(\theta) = 0$ collects the rest non-gluing case that cannot be proceeded. The implementation therefore rejects this representation and resumes the search loop. Accordingly, the probabilities studied below measure the failure to recognize and remove this initial non-gluing step.

We first show that for a Cornacchia pair $(z_0, z_1)$ among four solutions via sign-changing only will result in at most two different classes of EndType.

**Lemma 3.1** (Conjugate-pair redundancy)**.** For a valid pair $(x, y)$ computed from $\alpha, s, t$ in Algorithm 1, let $\beta_1^{(x,y)}, \beta_2^{(x,y)}$ be the elements obtained by substituting $(x, y)$ for $(z_0, z_1)$ in Eqs. (1) and (2) respectively, and define $\theta^{(x,y)} := \beta_2^{(x,y)}\overline{\beta_1^{(x,y)}}/n$.

For every valid pair $(x, y)$, we have

$$\theta^{(-x,-y)} = \overline{\theta^{(x,y)}}, \qquad \theta^{(-x,y)} = \overline{\theta^{(x,-y)}}.$$

[1] This condition will always hold in the current branch where the denominator is 1. For clarity of the presentation we keep condition.

Hence, each of the two pairs $\{(x,y),(-x,-y)\},\{(x,-y),(-x,y)\}$ has a common acceptance outcome under PrepareQIapotiOutput.

*Proof.* By Eqs. (1) and (2), for the prefix $\alpha, s, t$, we know

$$a_1 = \frac{x+s}{2}, \qquad a_2 = \frac{s-x}{2}, \qquad b_1 = \frac{y+t}{2}, \qquad b_2 = \frac{t-y}{2},$$

and $\beta_1 = n(a_1 + b_1 i) + \alpha, \beta_2 = n(a_2 + b_2 i) + \alpha$.

Hence, replacing both $x$ and $y$ by their negatives exchanges $a_1$ with $a_2$ and $b_1$ with $b_2$, i.e.

$$\beta_1^{(-x,-y)} = \beta_2^{(x,y)}, \qquad \beta_2^{(-x,-y)} = \beta_1^{(x,y)}.$$

Then we have the first equation

$$\theta^{(-x,-y)} = \beta_2^{(-x,-y)}\overline{\beta_1^{(-x,-y)}}/n = \beta_1^{(x,y)}\overline{\beta_2^{(x,y)}}/n = \overline{\theta^{(x,y)}}.$$

Similar computation will give the second equation.

Since the conjugation preserves the normalized denominator and reduced norm, the conjugate representatives are accepted or rejected simultaneously in both the OddNorm and EndType branches. □

Similarly when the swap condition in Line 20 holds, each of the two pairs $\{(y,x),(-y,-x)\}$, $\{(y,-x),(-y,x)\}$ has a common acceptance outcome under PrepareQIapotiOutput. Hence, following Lines 19 and 21, define the four effective representatives [2]

$$v_0 = (z_0, z_1), \qquad v_1 = (z_0, -z_1), \qquad v_2 = (z_1, z_0), \qquad v_3 = (z_1, -z_0).$$

### 3.1.2 Multi-representative Probability Analysis

We next quantify the success rates of the original and optimized tests. All probabilities in this subsection are conditioned on the effective representative orbit reaching the integral-$\theta$ branch of Algorithm 3. The second part of Lemma 3.4 shows that this is an orbit-level condition: once one effective representative has normalized denominator 1, all four do. We also use $e \geq 3$, as in all parameter sets considered here.

To motivate the analysis, our experimental results show that the single-representative failure rate is close to 87.5%, whereas the four-representative branch fails close to 75%, rather than $(0.875)^4$. This is because the four acceptance events are highly correlated.

1. **Heuristic parity model.** For each effective representative, let

$$d_1 = \frac{\mathsf{nrd}(\beta_1)}{n}.$$

   Conditioned on reaching the integral branch, we assume that $d_1$ is odd with probability approximately 1/2. This assumption is supported by our experiments.

2. **Norm-7 and profile classification.** We prove that $d_1$ is odd if and only if $\mathsf{nrd}(\theta) \equiv 7 \pmod 8$ (Lemma 3.2), and that exactly 32 modulo-4 profiles have this norm, of which the corrected EndType table accepts exactly 8 among 32 (Lemma 3.3).

3. **Heuristic profile distribution.** Conditioned on the corresponding $d_1$ being odd, we assume that the number of occurrences of the 8 accepting modulo-4 profiles is close to 1/4 among all 32 candidates profiles (i.e. 8/32=1/4), which is supported by the experiments.

[2] By Lemmas 3.5 and 3.6, $\{v_0, v_3\}$ is a hitting set for every nonempty EndType-acceptance set in the integral branch. We nevertheless retain all four effective representatives in the presentation because PrepareQIapotiOutput also contains the separate denominator-two OddNorm branch.

4. **Single-representative success probability.** Under the preceding models, a single representative is accepted with probability approximately
$$\frac{1}{2} \cdot \frac{1}{4} = \frac{1}{8},$$
and is rejected with probability approximately 7/8.

5. **Four-representative (two-representatives eventually) success probability.** Let $X$ be the number of accepted effective representatives. By the heuristic above, $\mathbb{E}[X] \approx 4 \cdot (1/8) = 1/2$. By Lemmas 3.5 and 3.6, $X \in \{0, 2\}$. Hence
$$\Pr[X = 2] = \frac{\mathbb{E}[X]}{2} \approx \frac{1}{4}.$$
Thus the four-representative branch succeeds with probability approximately 1/4 and rejects with probability approximately 3/4.

6. **Expected number of repetitions.** The corresponding geometric means for these conditional EndType trials are approximately 8 and 4, respectively, yielding a factor-2 reduction.

We consider two cases with respect to $n$, the norm of ideal $I$ in algorithm 3.

**Lemma 3.2** (Parity of $d_1$ and the norm of $\theta$)**.** Let $d_i = \frac{\mathsf{nrd}(\beta_i)}{n} \in \mathbb{Z}$ $(i \in \{1, 2\})$, and suppose that the Qlapoti norm equation gives $d_1 + d_2 = 2^e$ with $e \geq 3$. For $\theta = \beta_2\overline{\beta_1}/n$, one has
$$\mathsf{nrd}(\theta) = d_1 d_2 \quad \text{and} \quad d_1 \text{ is odd} \iff \mathsf{nrd}(\theta) \equiv 7 \pmod 8.$$

*Proof.* Multiplicativity of the reduced norm gives
$$\mathsf{nrd}(\theta) = \frac{\mathsf{nrd}(\beta_2)\mathsf{nrd}(\beta_1)}{n^2} = d_1 d_2.$$
Since $d_2 = 2^e - d_1$ and $e \geq 3$, reduction modulo 8 yields $\mathsf{nrd}(\theta) \equiv -d_1^2 \pmod 8$. If $d_1$ is odd, then $d_1^2 \equiv 1 \pmod 8$, so $\mathsf{nrd}(\theta) \equiv 7 \pmod 8$. Conversely, if $d_1$ is even, then $d_1^2 \equiv 0$ or $4 \pmod 8$, and therefore $-d_1^2 \not\equiv 7 \pmod 8$. □

**Lemma 3.3** (Norm-7 profiles accepted by EndType)**.** Assume that $p \equiv -1 \pmod 8$. Among the $4^4$ profiles
$$(t_1, t_2, t_3, t_4) \in (\mathbb{Z}/4\mathbb{Z})^4,$$
exactly 32 satisfy
$$t_1^2 + t_2^2 + p(t_3^2 + t_4^2) \equiv 7 \pmod 8.$$
Moreover, the two lists $\mathcal{T}_1$ and $\mathcal{T}_2$ in Algorithm 4 are disjoint, contain four profiles each, and all eight satisfy this congruence. Hence EndType accepts exactly 8 of the 32 profiles.

*Proof.* Since $p \equiv -1 \pmod 8$, the required condition is
$$t_1^2 + t_2^2 - t_3^2 - t_4^2 \equiv 7 \pmod 8.$$
For $t \in \mathbb{Z}/4\mathbb{Z}$, the square $t^2 \bmod 8$ maps $0 \mapsto 0$, $1, 3 \mapsto 1$, and $2 \mapsto 4$. Consequently, for an ordered pair $(t, t')$ mod 4, the number of solutions to $t^2 + t'^2 \equiv r \pmod 8$ is

| $r$ | 0 | 1 | 2 | 4 | 5 |
|---|---|---|---|---|---|
| $N_r$ | 2 | 4 | 4 | 2 | 4. |

Thus the number of profiles satisfying $t_1^2 + t_2^2 - (t_3^2 + t_4^2) \equiv 7 \pmod 8$ is
$$N_0 N_1 + N_1 N_2 + N_4 N_5 = 2 \cdot 4 + 4 \cdot 4 + 2 \cdot 4 = 32.$$

□

To simplify the presentation we need to show that using the EndType branch implies that $s$ has the same parity as $t$.

We prove Line 20 of Algorithm 5 always holds when PrepareQlapotiOutput runs the denominator 1 branch (i.e. runing the EndType branch).

**Lemma 3.4.** Let $(x, y)$ be a parity-compatible representative computed in Qlapoti and Qlapoti+ such that $x \equiv s \pmod 2, y \equiv t \pmod 2$. If $\theta^{(x,y)}$ has normalized denominator 1, then

$$s \equiv t \pmod 2.$$

Moreover, once one parity-compatible representative reaches the denominator-one branch, all four effective sign-and-swap representatives do.

*Proof.* Write $\alpha = a + bi + cj + dk$ and let $r = \mathsf{nrd}(\alpha)/n$. The CVP step produces $s, t$ satisfying

$$(2as + 2bt)u \equiv (2^e - 2r)u \pmod n,$$

where $u \in (\mathbb{Z}/n\mathbb{Z})^\times$. Cancelling $u$ shows that

$$\ell := \frac{2^e - 2r - 2as - 2bt}{n} \in \mathbb{Z}.$$

The Cornacchia target is then defined by

$$z_{\text{res}} = 2\ell - s^2 - t^2,$$

and the returned representative satisfies $x^2 + y^2 = z_{\text{res}}$. Consequently,

$$n(s^2 + t^2 + x^2 + y^2) + 4as + 4bt + 4r = 2^{e+1}. \tag{1}$$

Set

$$A = ns + 2a, \qquad B = nt + 2b, \qquad C = 2c, \qquad D = 2d.$$

Writing $\theta^{(x,y)} = q_0 + q_1 i + q_2 j + q_3 k$, we have

$$q_0 = \frac{A^2 + B^2 - n^2(x^2 + y^2) + p(C^2 + D^2)}{4n}, \quad q_2 = \frac{Cx - Dy}{2}, \quad q_3 = \frac{Dx + Cy}{2}.$$

Since $\theta^{(x,y)}$ has normalized denominator 1, all of $q_0, q_1, q_2, q_3$ are integers.

Using $\mathsf{nrd}(\alpha) = nr$, we obtain

$$4nq_0 = n^2(s^2 + t^2 - x^2 - y^2) + 4n(as + bt) + 4nr = n\big(2^{e+1} - 2n(x^2 + y^2)\big),$$

where the second equality follows from (1). Hence

$$q_0 = 2^{e-1} - \frac{n(x^2 + y^2)}{2}. \tag{2}$$

Suppose first that $n$ is odd. Since $q_0 \in \mathbb{Z}$, equation (2) implies that $x^2 + y^2$ is even. Thus $x \equiv y \pmod 2$, and therefore $s \equiv t \pmod 2$.

Suppose now that $n$ is even. By the generator condition, $\mathsf{gcd}(2a, n) = 1$, so $2a$ is odd. Since the coordinates of $\alpha \in O_0$ satisfy $2a \equiv D \pmod 2$, it follows that $D$ is odd.

If $x$ is even and $y$ is odd, then $Cx - Dy$ is odd, contradicting $q_2 \in \mathbb{Z}$. If $x$ is odd and $y$ is even, then $Dx + Cy$ is odd, contradicting $q_3 \in \mathbb{Z}$. Hence $x \equiv y \pmod 2$, and again $s \equiv t \pmod 2$.

It remains to prove that the denominator-one condition is preserved by the four effective representatives. Clearly, sign changes preserve the parity of the numerators of $q_1, q_2, q_3$, while $q_0$ depends on $(x, y)$ only through $x^2 + y^2$. Moreover, since $x \equiv y \pmod 2$, swapping $x$ and $y$ also preserves the parity of each numerator in

$$q_1 = \frac{Bx - Ay}{2}, \qquad q_2 = \frac{Cx - Dy}{2}, \qquad q_3 = \frac{Dx + Cy}{2}.$$

Hence, if one effective representative has normalized denominator 1, then all four do. □

We next show the central proposition that, on the four effective representatives, EndType returns a nonzero value either zero times or exactly twice. For a candidate in the four-representative branch, define

$$\mathcal{S} = \{j \in \{0,1,2,3\} : \mathsf{EndType}(\theta^{v_j}) \in \{1,2\}\}.$$

**Lemma 3.5** (Acceptance classification for odd $n$)**.** Assume that $p \equiv -1 \pmod 8$ and that $n$ is odd. Let $\mathcal{S}$ denote the index set of the four effective representatives generated by Qlapoti+ that pass the EndType test. Then

$$\mathcal{S} \in \big\{\varnothing, \{0,1\}, \{2,3\}\big\}.$$

*Proof.* Assume that $\mathcal{S} \neq \varnothing$. Whenever the proof below studies the parity of an accepted representative, we may temporarily denote that representative by $(x, y)$. Note that signs and swapping do not change whether its two coordinates are both odd or both even. After the parity step, $(x, y)$ again denotes the original Cornacchia pair used to define $v_0, \ldots, v_3$. Write $\alpha = a + bi + cj + dk \in O_0$ as in Algorithm 1 and define the integers

$$A = ns + 2a, \qquad B = nt + 2b, \qquad C = 2c, \qquad D = 2d.$$

Since the standard basis is $O_0 = \left\langle 1,\ i,\ \frac{i+j}{2},\ \frac{1+k}{2} \right\rangle$, for $\alpha$ we naturally have

$$2a \equiv 2d \equiv D \pmod 2, \qquad 2b \equiv 2c \equiv C \pmod 2.$$

(This is because by writing $\alpha = u + vi + w(i+j)/2 + r(1+k)/2$ in the displayed order basis gives $2a = 2u + r$, $2b = 2v + w$, $C = w$, and $D = r$.)

Since $n$ is odd by assumption, and Algorithm 2 and the parity adjustment in the algorithm gives $x \equiv s \pmod 2$ and $y \equiv t \pmod 2$. Hence,

$$A \equiv x + D \pmod 2, \qquad B \equiv y + C \pmod 2. \tag{3}$$

For a parity-compatible representative $(x, y)$, Eqs. (1) and (2) give

$$\beta_1^{(x,y)} = \frac{A + nx}{2} + \frac{B + ny}{2} i + cj + dk, \quad \beta_2^{(x,y)} = \frac{A - nx}{2} + \frac{B - ny}{2} i + cj + dk.$$

Write $\theta^{(x,y)} = q_0 + q_1 i + q_2 j + q_3 k$, a direct calculation gives

$$\begin{aligned} q_0 &= \frac{A^2 + B^2 - n^2(x^2 + y^2) + p(C^2 + D^2)}{4n}, \\ q_1 &= \frac{Bx - Ay}{2}, \quad q_2 = \frac{Cx - Dy}{2}, \quad q_3 = \frac{Dx + Cy}{2}. \end{aligned} \tag{4}$$

According to Algorithm 4, the modulo-4 profile tested by EndType is

$$\mathbf{t}(x, y) = (q_0, q_1, q_3, q_2) \bmod 4.$$

By Lemma 3.4, we have $s \equiv t \pmod 2$. In this case, claim that every accepted representative satisfies $x \equiv y \equiv 0 \pmod 2$. Suppose instead that $x$ and $y$ are both odd. Integrality of $q_2$ and $q_3$ then implies $C \equiv D \pmod 2$.

Every profile in $\mathcal{T}_1 \cup \mathcal{T}_2$ has exactly one of $q_2, q_3$ to be odd. On the other hand,

$$4(q_2^2 + q_3^2) = (C^2 + D^2)(x^2 + y^2).$$

Since $x^2 + y^2 \equiv 2 \pmod 8$ , the fact that $q_2^2 + q_3^2$ is odd forces $C \equiv D \equiv 1 \pmod 2$.

It follows from Eq. (3) that $A$ and $B$ are even. Write $A = 2A', B = 2B'$. All accepted (non-zero) elements of EndType have $q_1$ even, and hence

$$q_1 = B'x - A'y \equiv 0 \pmod 2.$$

The congruence follows because the second coordinate of every profile in $\mathcal{T}_1 \cup \mathcal{T}_2$ is even. By the hypothesis that $x$ and $y$ are odd, this gives $A' \equiv B' \pmod 2$, and therefore

$$A^2 + B^2 = 4(A'^2 + B'^2) \equiv 0 \pmod 8.$$

Using the fact that $p \equiv -1 \pmod 8$, $n^2 \equiv 1 \pmod 8$, and the oddness of $x, y, C, D$, the first equation of Eq. (4) gives

$$4nq_0 \equiv 0 - 2 - 2 \equiv 4 \pmod 8.$$

Thus $q_0$ is odd, contradicting the fact that every element in $\mathcal{T}_1 \cup \mathcal{T}_2$ has even first coordinate. Consequently,

$$x \equiv y \equiv 0 \pmod 2.$$

Write $x = 2x'$ and $y = 2y'$. Equation (3) now reduces to $A \equiv D \pmod 2, B \equiv C \pmod 2$. The four effective representatives therefore have the following EndType profiles:

$$\begin{aligned}
\mathbf{t}(v_0) &= \big(q_0,\ Bx' - Ay',\ Dx' + Cy',\ Cx' - Dy'\big),\\
\mathbf{t}(v_1) &= \big(q_0,\ Bx' + Ay',\ Dx' - Cy',\ Cx' + Dy'\big),\\
\mathbf{t}(v_2) &= \big(q_0,\ By' - Ax',\ Dy' + Cx',\ Cy' - Dx'\big),\\
\mathbf{t}(v_3) &= \big(q_0,\ By' + Ax',\ Dy' - Cx',\ Cy' + Dx'\big),
\end{aligned} \qquad \pmod 4.$$

For each of these profiles, the difference between the second and fourth coordinates is a sum of terms with coefficients $A - D$ and $B - C$, both even. Hence these two coordinates have the same parity.

Every profile in $\mathcal{T}_2$, however, has an even second coordinate and an odd fourth coordinate. Hence, profiles in $\mathcal{T}_2$ cannot occur in this branch. It remains to consider Type 1. Recall that its 4 profiles are exactly in the form $(2, u, \varepsilon, 2 - u)$, for $u \in \{0, 2\}, \varepsilon \in \{1, 3\}$.

Equivalently, the third coordinate is odd, the second and fourth coordinates are even, and their *sum* is 2 modulo 4.

Suppose first that $v_0$ has Type 1. Passing from $v_0$ to $v_1$ changes the last three coordinates by

$$2Ay', \qquad -2Cy', \qquad 2Dy',$$

respectively. Thus, their parities are unchanged. Moreover,

$$\begin{aligned}
&\big(Bx' + Ay'\big) + \big(Cx' + Dy'\big)\\
&\quad - \big(Bx' - Ay'\big) - \big(Cx' - Dy'\big) = 2(A + D)y' \equiv 0 \pmod 4,
\end{aligned}$$

where the last congruence follows from $A \equiv D \pmod 2$. Therefore, $v_1$ also has Type 1. Similarly, if $v_1$ is accepted, then $v_0$ has Type 1.

For $v_2, v_3$, notice that their third coordinates, for the same reason, satisfy

$$Dy' \pm Cx' \equiv Cx' - Dy' \pmod 2.$$

The right-hand side is the fourth coordinate of $\mathbf{t}(v_0)$, which is even for Type 1. Hence, the third coordinates of both $\mathbf{t}(v_2)$ and $\mathbf{t}(v_3)$ are even, so neither can have Type 1. Since Type 2 has already been excluded, both are rejected. Thus $v_0$ and $v_1$ are accepted simultaneously, while $v_2$ and $v_3$ are rejected; hence $\mathcal{S} = \{0, 1\}$.

Applying the same flow after interchanging $x'$ and $y'$ shows that $v_2$ and $v_3$ are accepted simultaneously and exclude $v_0, v_1$. Hence the only remaining nonempty possibility is $\mathcal{S} = \{2, 3\}$. Together with the empty case, this proves the claim. □

**Lemma 3.6** (Acceptance classification for even $n$)**.** Assume that $e \geq 3$ and that $n$ is even. Let $\mathcal{S}$ denote the index set of the four effective representatives generated by Qlapoti+ that pass the EndType test. Then

$$\mathcal{S} \in \{\varnothing, \{0,2\}, \{1,3\}\}.$$

*Proof.* Use the same notation $A = ns + 2a, B = nt + 2b, C = 2c, D = 2d$ from the proof of Lemma 3.5. Since $n$ is even, the order relations $2a \equiv D \pmod 2$ and $2b \equiv C \pmod 2$ give

$$A \equiv D \pmod 2, \qquad B \equiv C \pmod 2.$$

Moreover, since $\gcd(2a, n) = 1$ by construction, $2a$ is odd. Consequently, $A$ and $D$ are odd.

Recall from (4) that

$$q_1 = \frac{Bx - Ay}{2}, \qquad q_2 = \frac{Cx - Dy}{2}, \qquad q_3 = \frac{Dx + Cy}{2},$$

and that the profile tested by EndType is $\mathbf{t}(x, y) = (q_0, q_1, q_3, q_2) \bmod 4$.

By Lemma 3.4, we have $s \equiv t \pmod 2$. Since sign changes and swapping preserve the common parity of the two coordinates, we may, without loss of generality, denote an accepted representative by $(x, y)$. We claim that $x$ and $y$ are both odd.

Suppose instead that $x$ and $y$ are both even, and write $x = 2X$ and $y = 2Y$. From the Qlapoti norm equation,

$$q_0 = 2^{e-1} - \frac{n(x^2 + y^2)}{2} = 2^{e-1} - 2n(X^2 + Y^2) \equiv 0 \pmod 4,$$

because the last congruence uses $e \geq 3$ and $n$ is even. Thus Type 1 is not possible.

Moreover, $q_1 = BX - AY$ and $q_2 = CX - DY$. Since $A \equiv D \pmod 2$ and $B \equiv C \pmod 2$, the second and fourth coordinates $q_1, q_2$ have the same parity. Every profile in $\mathcal{T}_2$, however, has an even second coordinate and an odd fourth coordinate. Hence Type 2 is also impossible. Consequently, every accepted representative satisfies $x \equiv y \equiv 1 \pmod 2$.

Integrality of $q_2$ and $q_3$ then implies $C \equiv D \pmod 2$. Since $D$ is odd, $C$ is also odd. It follows that $A \equiv B \equiv C \equiv D \equiv 1 \pmod 2$.

For the four effective representatives, the EndType profiles are

$$\begin{aligned}
\mathbf{t}(v_0) &= \left(q_0, \frac{Bx - Ay}{2}, \frac{Dx + Cy}{2}, \frac{Cx - Dy}{2}\right), \\
\mathbf{t}(v_1) &= \left(q_0, \frac{Bx + Ay}{2}, \frac{Dx - Cy}{2}, \frac{Cx + Dy}{2}\right), \\
\mathbf{t}(v_2) &= \left(q_0, \frac{By - Ax}{2}, \frac{Dy + Cx}{2}, \frac{Cy - Dx}{2}\right), \\
\mathbf{t}(v_3) &= \left(q_0, \frac{By + Ax}{2}, \frac{Dy - Cx}{2}, \frac{Cy + Dx}{2}\right),
\end{aligned} \pmod 4.$$

Suppose first that $v_0$ is accepted. Write $\mathbf{t}(v_j) = (q_0, u_j, r_j, w_j)$ for $j \in \{0, 1, 2, 3\}$, and put $h = (x - y)/2$. The profiles of $v_0$ and $v_2$ satisfy

$$u_0 - u_2 = (A + B)h, \quad r_0 - r_2 = (D - C)h, \quad w_0 - w_2 = (C + D)h.$$

Since $A, B, C, D$ are odd, corresponding coordinates have the same parity.

If $h$ is even, all three differences are divisible by 4, so $v_0$ and $v_2$ have the same EndType.

Suppose that $h$ is odd. Then $y \equiv -x \pmod 4$. If $v_0$ has Type 1, then $u_0 + w_0$ is even, while

$$2(u_0 + w_0) = (B + C)x - (A + D)y \equiv x(A + B + C + D) \pmod 4.$$

Since $x$ is odd, this gives $A + B + C + D \equiv 0 \pmod 4$. Therefore,

$$(u_0 + w_0) - (u_2 + w_2) = (A + B + C + D)h \equiv 0 \pmod 4.$$

Together with the equal coordinate parities, this shows that $v_2$ also has Type 1.

If $v_0$ has Type 2, then $u_0$ and $r_0$ are even. Using $y \equiv -x \pmod 4$, we obtain

$$2u_0 = Bx - Ay \equiv x(A + B) \pmod 4,$$
$$2r_0 = Dx + Cy \equiv x(D - C) \pmod 4.$$

Hence $A + B \equiv 0 \pmod 4$ and $C - D \equiv 0 \pmod 4$. It follows that $u_0 \equiv u_2 \pmod 4$ and $r_0 \equiv r_2 \pmod 4$. Together with the common scalar coordinate and the equal parity of $w_0, w_2$, this shows that $v_2$ also has Type 2.

Thus, if $v_0$ is accepted, then $v_2$ is accepted with the same EndType. Exchanging $x$ and $y$ gives the converse implication, so $v_0$ and $v_2$ are accepted simultaneously.

Moreover, we know

$$u_0 + u_1 = Bx, \qquad u_2 + u_3 = By.$$

Since $B, x, y$ are odd, both sums are odd. As every accepted profile has an even second coordinate, the acceptance of $v_0$ and $v_2$ forces $u_1$ and $u_3$ to be odd. Hence $v_1, v_3$ are rejected, and therefore $\mathcal{S} = \{0, 2\}$.

Replacing $y$ by $-y$ gives the symmetric implication: $v_1$ and $v_3$ are accepted simultaneously and exclude $v_0, v_2$. Hence $\mathcal{S} = \{1, 3\}$. If no representative is accepted, then $\mathcal{S} = \varnothing$. These are the only possibilities, and therefore $\mathcal{S} \in \big\{\varnothing, \{0, 2\}, \{1, 3\}\big\}$. □

Hence, by the simple proof at the beginning of this subsection, we have the following corollary.

**Corollary 3.1.** Assume that $p \equiv -1 \pmod 8$ and that the heuristic parity and profile-distribution models in Items 1 and 3 hold. Conditioned on reaching the integral-$\theta$ branch, the original Qlapoti procedure has probability approximately $1/8$ that EndType returns a nonzero value. In the four-representative branch, Qlapoti+ has probability approximately $1/4$ that one of the representatives gives a nonzero EndType value.

**Exhaustiveness with respect to other Cornacchia representations.** The following lemma shows the representatives considered in this work is optimal among the all sum-of-two-squares representations of a composite target.

**Lemma 3.7** (EndType exhaustiveness of the effective representatives)**.** Fix $\alpha, s, t, n$, and let $(x, y)$ and $(x', y')$ be two parity-compatible representations of the same positive Cornacchia target

$$x^2 + y^2 = x'^2 + y'^2 = z.$$

Assume that ModCondition$(z, s, t)$ holds, this excludes the case $z \equiv 0 \pmod 8$. Then the four effective sign-and-swap variants of $(x, y)$ contain an EndType-accepted representative if and only if the four effective variants of $(x', y')$ do.

*Proof.* Suppose first that one of the two four-representative reaches the integral-$\theta$ branch; otherwise neither group contains an EndType-accepted representative and the claim is immediate. By Lemma 3.4, $s \equiv t \pmod 2$, and parity compatibility gives $x \equiv y \pmod 2$ and $x' \equiv y' \pmod 2$. The conditions in ModCondition leave two cases.

If $s, t$ are odd, then $z \equiv 2 \pmod 4$, so all four coordinates $x, y, x', y'$ are odd. Every odd square is 1 or 9 $\pmod{16}$. If $z \equiv 10 \pmod{16}$, each representation has one coordinate in $\{\pm 1\}$ $\pmod 8$ and the other in $\{\pm 3\} \pmod 8$; hence any two representations differ modulo 8 by signs

and possibly a swap. If $z \equiv 2 \pmod{16}$, the two coordinates of each representation have the same square modulo 16. After signs and possibly a swap, one therefore has either

$$(x', y') \equiv (x, y) \pmod 8 \qquad \text{or} \qquad (x', y') \equiv 3(x, y) \pmod 8.$$

If $s, t$ are even, then $z \equiv 0 \pmod 4$; it suffices to consider $z \equiv 4 \pmod 8$. Thus all coordinates are even. Writing $x = 2X$, $y = 2Y$, $x' = 2X'$, and $y' = 2Y'$, we have $X^2 + Y^2 = X'^2 + Y'^2 = z/4$, which is odd. Hence one coordinate is odd and the other is even. Modulo 4, the odd coordinate is $\pm 1$, whereas the even coordinate is forced to be 0 when $z/4 \equiv 1 \pmod 8$ and 2 when $z/4 \equiv 5 \pmod 8$. Consequently, $(X', Y')$ differs from $(X, Y)$ modulo 4 only by signs and a possible swap, and therefore the same holds for $(x', y')$ and $(x, y)$ modulo 8.

It remains to compare EndType profiles. With the notation of Eq. (4), the scalar coordinate $q_0$ depends on the representative only through $x^2 + y^2 = z$, and is therefore unchanged. The other three coordinates are linear in $x, y$ and are divided by 2. Consequently, the congruences above preserve both the integrality of these coordinates and their residues modulo 4. Thus congruent representatives modulo 8 give the same profile modulo 4. In the only additional case, $(x', y') \equiv 3(x, y) \pmod 8$, one has

$$(q_1', q_3', q_2') \equiv 3(q_1, q_3, q_2) \equiv -(q_1, q_3, q_2) \pmod 4.$$

Both $\mathcal{T}_1$ and $\mathcal{T}_2$ are invariant under

$$(t_0, t_1, t_2, t_3) \longmapsto (t_0, -t_1, -t_2, -t_3) \pmod 4.$$

Finally, simultaneous negation of a representative produces the same acceptance outcome by Lemma 3.1. Therefore the four effective representatives cover every EndType-acceptance class derived from any other parity-compatible representation of the same $z$. □

### 3.2 Quick Solvability Test

The quick solvability test is shown in Line 16 of Algorithm 5. It adds an early rejection step *inside* the *extended Cornacchia* algorithm before the probable-prime test.

**Original Implementation.** The original *extended Cornacchia* first removes the powers of the small primes in its precomputed list from $z$. Let $z_{\text{res}}$ denote the remaining factor. If $z_{\text{res}} \neq 1$, the implementation of [BCE+25] first applies a probable-prime test to $z_{\text{res}}$, which is expensive. If this test succeeds, it then applies Cornacchia to $z_{\text{res}}$ to solve for

$$x^2 + qy^2 = z_{\text{res}}$$

Consequently, a residual factor that cannot be represented as a sum of two squares may still incur the cost of a probable-prime test before it is rejected.

**Optimization.** For the Qlapoti instance of extended Cornacchia, we have $q = 1 \bmod 4$ or simply $q = 1$, the primary case in practice, and the target equation is of the form

$$x^2 + y^2 = z_{\text{res}}.$$

We recall the following classical characterization.

**Lemma 3.8** (Sum of two squares theorem)**.** A positive integer $N$ can be written as $N = x^2 + y^2$ with $x, y \in \mathbb{Z}$ if and only if every prime $p \equiv 3 \pmod 4$ occurs with even valuation in the factorization of $N$.

In particular, if $z_{\text{res}} \equiv 3 \pmod 4$, then at least one prime $p \equiv 3 \pmod 4$ occurs in $z_{\text{res}}$ with odd valuation. Therefore, $z_{\text{res}}$ cannot be represented as a sum of two squares.

We thus perform the inexpensive test

$$z_{\text{res}} \equiv 3 \pmod 4$$

before the probable-prime test and immediately reject the candidate when it holds. Candidates passing this test continue through the original probable-prime, Legendre-symbol, and Cornacchia steps. The new condition is only an early rejection test: passing it does not by itself imply that $z_{\text{res}}$ is prime or representable as a sum of two squares.

Compared with the baseline Qlapoti implementation [BCE+25], this gives more than a 20% reduction in the cost of the Qlapoti procedure itself at NIST security level 5 on the same input averaged over 1000 trials.

### 3.3 Faster Normalization

The normalization optimization is shown in Line 10 of Algorithm 5. It replaces the incremental construction and generic normalization of $\lambda\alpha_0$ with a specialized normalization step.

**Original Implementation.** For a fixed generator $\alpha_0$, [BCE+25] tests the elements $\alpha = \lambda\alpha_0$ for increasing integers $\lambda$ coprime to $n$. The original implementation does not compute this multiple by scalar multiplication. Instead, it maintains the current value of $\alpha$ and updates it incrementally as

$$\alpha \leftarrow \alpha + \alpha_0$$

whenever $\lambda$ is increased. This choice is actually an optimization proposed in [BCE+25] in replace of totally randomized $\alpha$ so that the internal computation can be reused.

Since $\lambda$ is typically small, this is slightly faster than a direct multiplication in its own. The resulting element is then passed to the generic quaternion normalization routine. Roughly speaking, if $\alpha_0 = (x_0 + x_1 i + x_2 j + x_3 k)/d$ then

$$\alpha = \frac{\lambda x_0 + \lambda x_1 i + \lambda x_2 j + \lambda x_3 k}{d}.$$

The generic normalization procedure first computes $h = \mathsf{gcd}(\lambda x_0, \lambda x_1, \lambda x_2, \lambda x_3, d)$ and returns

$$\frac{(\lambda x_0/h) + (\lambda x_1/h)i + (\lambda x_2/h)j + (\lambda x_3/h)k}{d/h}.$$

**Optimization.** The key observation of our optimization here is that if we normalize $\alpha_0$ first, then $h = \mathsf{gcd}(\lambda, d)$. We replace both steps by a direct construction. When a new $\alpha_0$ is sampled, we normalize it once and write

$$\alpha_0 = \frac{x_0 + x_1 i + x_2 j + x_3 k}{d}, \qquad \mathsf{gcd}(x_0, x_1, x_2, x_3, d) = 1.$$

For each admissible $\lambda$, we construct

$$\alpha = \lambda\alpha_0 = \frac{\lambda x_0 + \lambda x_1 i + \lambda x_2 j + \lambda x_3 k}{d}$$

directly by multiplying the four coordinates of $\alpha_0$ by $\lambda$. This avoids all intermediate quaternion additions, including those corresponding to skipped values of $\lambda$.

The direct construction also allows a specialized normalization. Since $\alpha_0$ is already normalized, we have the following secondary school lemma with skipped proof:

**Lemma 3.9.** Let $n \geq 1$, let $\lambda, d$ be positive integers, and let $x_0, \ldots, x_{n-1} \in \mathbb{Z}$. Then

$$\mathsf{gcd}(\lambda x_0, \ldots, \lambda x_{n-1}, d) = \mathsf{gcd}\big(\lambda \cdot \mathsf{gcd}(x_0, \ldots, x_{n-1}, d), d\big).$$

Hence, it suffices to compute

$$g = \mathsf{gcd}(\lambda, d)$$

and set

$$\alpha = \frac{(\lambda x_0/g) + (\lambda x_1/g)i + (\lambda x_2/g)j + (\lambda x_3/g)k}{d/g}.$$

The optimization caches a normalized representation of each newly sampled base generator $\alpha_0$. Subsequent multiples $\alpha = \lambda\alpha_0$ are reconstructed using four integer multiplications rather than using incremental update by $\alpha \leftarrow \alpha + \alpha_0$. This may be more expensive than quaternion addition for small $\lambda$, but it allows the generic normalization to be replaced by the single gcd computation $\mathsf{gcd}(\lambda, d)$, where $d$ is the cached denominator of $\alpha_0$. Divisions are required only when this gcd is greater than 1. Hence, the optimization trades a potentially more expensive update of $\alpha$ for a substantially cheaper normalization step. Compared with the baseline Qlapoti implementation [BCE+25], this gives more than a 20% reduction in the cost of the Qlapoti procedure itself at NIST security level 5 on the same input averaged over 1000 trails.

### 3.4 Cache-and-Multiply Inverse Update

The inversion optimization is shown in Lines 6 and 12 of Algorithm 5. The techniques are twofold. It first replaces the repeated inversion of $2a_\alpha = 2\lambda a_0$ with a cached inverse and the following computational trick:

**Cost**(Small Inversion w. Large Modulo) + **Cost**(Large Multiplication) < **Cost**(Large Inversion w. Large Modulo).

We further use another trick:

**Cost**(Large-by-Small Division) + **Cost**(Small Inversion w. Small Modulo) + **Cost**(Large Multiplication) < **Cost**(Small Inversion w. Large Modulo).

**Original Implementation.** For a fixed generator $\alpha_0$, [BCE+25] considers $\alpha = \lambda\alpha_0$ for increasing integers $\lambda$ satisfying $\mathsf{gcd}(\lambda, n) = 1$. To construct the target of the corresponding CVP instance, the implementation computes

$$u = (2a_\alpha)^{-1} \bmod n.$$

In the original implementation, the inverse of $2a_\alpha$ is recomputed from the incremental value $\lambda$. Here

$$a_\alpha = \lambda a_0,$$

this amounts to repeatedly computing

$$(2\lambda a_0)^{-1} \bmod n.$$

**Optimization.** The key observation is that the factor depending on the base generator can be inverted once and reused. Since $\mathsf{gcd}(2a_0, n) = 1$, when a new generator $\alpha_0$ is sampled, we compute and cache

$$u_0 = (2a_0)^{-1} \bmod n.$$

Moreover, since $\lambda$ is coprime to $n$, and hence

$$(2a_\alpha)^{-1} = (2\lambda a_0)^{-1} = (2a_0)^{-1}\lambda^{-1} \pmod n.$$

For $\lambda = 1$, the cached value can be used directly as $u = u_0$. For each subsequent value of $\lambda$, it suffices to compute $u_\lambda = \lambda^{-1} \bmod n$ and obtain

$$u = u_0 u_\lambda \bmod n.$$

The optimization thus computes the inverse associated with $\alpha_0$ only once and reuses it for all subsequent multiples of the same generator. Each reused iteration replaces the inversion of the full coordinate $2a_\alpha$ with the inversion of the typically *small* multiplier $\lambda$, followed by one modular multiplication. Concretely, in our experiments, $\lambda$ is typically 7-14 bits up to the NIST security level. In contrast, $2 \cdot \lambda \cdot a_0$ is 130-283 bits long. This makes small inversion 4.8 to 7.4 times faster with a large multiplication overhead, which is a small portion compared with a large inversion.

**Further Optimization.** For the inversion of the small $\lambda$ modulo large $n$, we use and optimize the Arazi's lemma [JP03, Lemma 1] for our circumstance here. Arazi's lemma is a classical technique in the pre-quantum literature and replaces large modulo inversion with a small modulo inversion, large multiplication and one exact division by the formula

$$\lambda^{-1} \bmod n = \frac{1 + n\left(-n^{-1} \bmod \lambda\right)}{\lambda}.$$

We further observe that the reduction modulo $\lambda$ and the exact division can be fused into a single quotient-and-remainder operation. Writing $n = q\lambda + r$, the above formula becomes

$$\lambda^{-1} \bmod n = \left(-r^{-1} \bmod \lambda\right) q + \frac{1 + \left(-r^{-1} \bmod \lambda\right) r}{\lambda},$$

where the second division involves only small integers. This makes the inversion twice faster.

Compared with the baseline Qlapoti implementation [BCE+25], this gives more than 30% reduction in the cost of the Qlapoti procedure itself at NIST security level 5 on the same input averaged over 1000 trails.

## 4 Experiments

We evaluate the proposed optimizations against the original Qlapoti C implementation [3] and its optimized SQIsign and salt-PRISM [4] in Table 1 and Table 2 respectively. The numbers do not take the precomputation suggestion into account Sec. A.

https://github.com/MaverickOtaku/Qlapoti_Plus

All experiments were performed on an Intel Core Ultra 9 275HX processor (2.7 GHz), using GCC 13.3.0, the 64-bit radix configuration, and the system GMP library. Each reported timing is averaged over four independent runs of 300 executions and is given in millions of CPU cycles. We consider both the reference and x86-64 assembly-optimized builds. The reference C implementation was compiled with `-DSQISIGN_BUILD_TYPE=ref` and `-DCMAKE_BUILD_TYPE=Release`. The x86-64 assembly-optimized implementation was compiled with `-DSQISIGN_BUILD_TYPE=broadwell` and `-DCMAKE_BUILD_TYPE=Release`. The verification timings are included only as reference values, since the proposed optimizations do not modify the verification procedure.

[3] https://github.com/KULeuven-COSIC/Qlapoti/tree/main/C-implementation
[4] https://github.com/mariascrs/PRISM_v2

**Remark 4.1.** In the Qlapoti code, there is an unused branch, denoted BadPrime, in the extended Cornacchia routine. This branch skips the loop when a small prime factor is detected for which the corresponding Cornacchia equation has no solution. Enabling this branch only gives a modest speedup for the Qlapoti procedure itself ($\approx 4\%$–$6\%$), and an even smaller improvement in the end-to-end signing performance ($\approx 1\%$–$1.5\%$). Nevertheless, we enable this branch in both versions in the comparison below.

Table 1: Performance of the Qlapoti optimization in the reference and Broadwell implementations against the SQIsign implementation optimized in [BCE+25]. Timings are averaged over five runs of 300 executions and reported in $10^6$ cycles. The Qlapoti rows report the standalone `qlapoti_normeq` benchmark. The factor is computed as Baseline/Optimized. Verification timings are included for reference only.

| | | Reference | | | Broadwell | | |
|---|---|---|---|---|---|---|---|
| **Level** | **Operation** | **Baseline** | **Optimized** | **Factor** | **Baseline** | **Optimized** | **Factor** |
| 1 | Qlapoti/Qlapoti+ | 5.89 | 3.68 | 1.60× | 3.66 | 1.96 | 1.87× |
| | KeyGen | 23.85 | 19.23 | 1.24× | 13.79 | 10.67 | 1.29× |
| | Sign | 58.73 | 48.60 | 1.21× | 35.77 | 29.16 | 1.23× |
| | Verify | 6.49 | 6.50 | 1.00× | 3.03 | 2.95 | 1.03× |
| 3 | Qlapoti/Qlapoti+ | 40.19 | 8.96 | 4.49× | 38.23 | 7.18 | 5.33× |
| | KeyGen | 89.31 | 50.14 | 1.78× | 72.66 | 32.61 | 2.23× |
| | Sign | 203.43 | 124.26 | 1.64× | 157.92 | 88.45 | 1.79× |
| | Verify | 17.28 | 17.26 | 1.00× | 10.46 | 10.39 | 1.01× |
| 5 | Qlapoti/Qlapoti+ | 30.65 | 8.72 | 3.51× | 28.43 | 7.10 | 4.00× |
| | KeyGen | 126.70 | 91.62 | 1.38× | 84.60 | 55.09 | 1.54× |
| | Sign | 312.24 | 234.43 | 1.33× | 216.11 | 151.65 | 1.43× |
| | Verify | 34.71 | 34.61 | 1.00× | 19.55 | 19.52 | 1.00× |

Table 2: Performance comparison between the Qlapoti+ optimization in the reference and Broadwell implementations of salt-PRISM [BBC+26] and using Qlapoti. We further use the optimization in Sec. B in our version. Timings are averaged over four runs of 300 executions and are reported in $10^6$ cycles. The factor is computed as Baseline/Optimized. Verification timings are included for reference only.

| | | Reference | | | Broadwell | | |
|---|---|---|---|---|---|---|---|
| **Level** | **Operation** | **Baseline** | **Optimized** | **Factor** | **Baseline** | **Optimized** | **Factor** |
| 1 | KeyGen | 25.16 | 22.27 | 1.13× | 14.33 | 11.79 | 1.22× |
| | Sign | 40.11 | 30.09 | 1.33× | 27.36 | 18.65 | 1.47× |
| | Verify | 7.87 | 7.87 | 1.00× | 3.51 | 3.58 | 0.98× |
| 3 | KeyGen | 93.84 | 57.12 | 1.64× | 70.53 | 37.70 | 1.87× |
| | Sign | 128.72 | 76.25 | 1.69× | 104.13 | 54.91 | 1.90× |
| | Verify | 18.52 | 18.56 | 1.00× | 10.86 | 10.89 | 1.00× |
| 5 | KeyGen | 131.89 | 105.80 | 1.25× | 92.53 | 63.52 | 1.46× |
| | Sign | 200.12 | 144.69 | 1.38× | 155.75 | 97.55 | 1.60× |
| | Verify | 34.81 | 34.84 | 1.00× | 19.29 | 19.20 | 1.00× |

## 5 Conclusion

In this work, we optimized the Qlapoti procedure of [BCE+25], which searches for suitable equivalent ideals in quaternion-based isogeny computations. The resulting procedure, Qlapoti+, combines four optimizations that reduce both the number of rejected candidates and the cost of processing each candidate. We integrated Qlapoti+ into [BCE+25, BBC+26], obtaining substantial improvements in the Qlapoti computation and in the end-to-end costs of key generation and signing. Under the Broadwell benchmark, Qlapoti+ accelerates the Qlapoti computation by 1.87×–5.33×. In SQIsign, this translates into 1.29×–2.23× speedups for key generation and 1.23×–1.79× for signing across the three security levels. In salt-PRISM, the optimization results in 1.22×–1.87× speedups for key generation and 1.47×–1.90× for signing. The security analysis remains essentially the same as [BCE+25, BBC+26], except for the use of multiple representatives considered in Sec. 3.1. With our optimized two-representative strategy, this introduces at most a factor-two loss, corresponding to at most one bit of security.

After several years of improvements to quaternion and ideal-to-isogeny algorithms, many of the associated translation steps can now be performed efficiently. As a result, searching for prime norms satisfying the required arithmetic constraints has become one of the most expensive components in these isogeny-based signature schemes. Our results show that reducing the rejection overhead and arithmetic cost of this search directly improves the performance of the complete protocols.

We expect that further improvements can be obtained by refining the generation and filtering of prime candidates and by more closely coordinating the ideal-finding procedure with the subsequent ideal-to-isogeny computation. We also expect the techniques developed in this work to be useful for other isogeny-based constructions.

## References


[AAA+24] Marius A. Aardal, Gora Adj, Diego F. Aranha, Andrea Basso, Isaac Andrés Canales Martínez, Jorge Chávez-Saab, Maria Corte-Real Santos, Pierrick Dartois, Luca De Feo, Max Duparc, Jonathan Komada Eriksen, Tako Boris Fouotsa, Décio Luiz Gazzoni Filho, Basil Hess, David Kohel, Antonin Leroux, Patrick Longa, Luciano Maino, Michael Meyer, Kohei Nakagawa, Hiroshi Onuki, Lorenz Panny, Sikhar Patranabis, Christophe Petit, Giacomo Pope, Krijn Reijnders, Damien Robert, Francisco Rodríguez Henríquez, Sina Schaeffler, and Benjamin Wesolowski. SQIsign. Technical report, National Institute of Standards and Technology, 2024. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures.

[Bab86] László Babai. On lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[BBC+25] Andrea Basso, Giacomo Borin, Wouter Castryck, Maria Corte-Real Santos, Riccardo Invernizzi, Antonin Leroux, Luciano Maino, Frederik Vercauteren, and Benjamin Wesolowski. PRISM: Simple and compact identification and signatures from large prime degree isogenies. In Tibor Jager and Jiaxin Pan, editors, *PKC 2025, Part III*, volume 15676 of *LNCS*, pages 300–332, Røros, Norway, May 12–15, 2025. Springer, Cham, Switzerland.

[BBC+26] Andrea Basso, Giacomo Borin, Wouter Castryck, Maria Corte-Real Santos, Riccardo Invernizzi, Antonin Leroux, Luciano Maino, Frederik Vercauteren, and Benjamin Wesolowski. PRISM with a pinch of salt: Simple, efficient and strongly unforgeable signatures from isogenies. Cryptology ePrint Archive, Paper 2026/443, 2026.

[BCE+25] Giacomo Borin, Maria Corte-Real Santos, Jonathan Komada Eriksen, Riccardo Invernizzi, Marzio Mula, Sina Schaeffler, and Frederik Vercauteren. Qlapoti: Simple and efficient translation of quaternion ideals to isogenies. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part IV*, volume 16248 of *LNCS*, pages 174–205, Melbourne, VIC, Australia, December 8–12, 2025. Springer, Singapore, Singapore.

[BDD+24] Andrea Basso, Pierrick Dartois, Luca De Feo, Antonin Leroux, Luciano Maino, Giacomo Pope, Damien Robert, and Benjamin Wesolowski. SQIsign2D-West - the fast, the small, and the safer. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part III*, volume 15486 of *LNCS*, pages 339–370, Kolkata, India, December 9–13, 2024. Springer, Singapore, Singapore.

[DKL+20] Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. SQISign: Compact post-quantum signatures from quaternions and isogenies. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 64–93, Daejeon, South Korea, December 7–11, 2020. Springer, Cham, Switzerland.

[JP03] Marc Joye and Pascal Paillier. GCD-free algorithms for computing modular inverses. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *CHES 2003*, volume 2779 of *LNCS*, pages 243–253, Cologne, Germany, September 8–10, 2003. Springer Berlin Heidelberg, Germany.

[Ler25] Antonin Leroux. Verifiable random function from the deuring correspondence and higher dimensional isogenies. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part VII*, volume 15607 of *LNCS*, pages 167–194, Madrid, Spain, May 4–8, 2025. Springer, Cham, Switzerland.

[NO24] Kohei Nakagawa and Hiroshi Onuki. QFESTA: Efficient algorithms and parameters for FESTA using quaternion algebras. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part V*, volume 14924 of *LNCS*, pages 75–106, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

[PR23] Aurel Page and Damien Robert. Introducing clapoti(s): Evaluating the isogeny class group action in polynomial time. Cryptology ePrint Archive, Report 2023/1766, 2023.

[Wes26] Benjamin Wesolowski. The supersingular isogeny problem in time and memory $p^{1/3+o(1)}$. Cryptology ePrint Archive, Paper 2026/1486, 2026.

# A An Additional Precomputation for Isogeny-Based Signatures

Independently of the optimizations described above, we propose precomputing the Weil pairing

$$g_0 = e_{2^f}(P_0, Q_0)$$

for the fixed basis $B_0 = (P_0, Q_0)$ of the $2^f$-torsion of the base curve $E_0$. The existing parameter data, including the fixed curve, torsion basis, and endomorphism action matrices, already occupy several kilobytes. Storing one additional encoded element of $\mathbb{F}_{p^2}$ therefore incurs only a small storage overhead. In return, this precomputation removes at least one full Weil pairing from each execution of key generation and signing, reducing their running times by approximately 1.5%–3.5%, depending on the parameter set.

This precomputation applies to the final-factor selection in the ideal-to-isogeny translation. At the protocol level, key generation uses this translation to evaluate the reduced secret ideal and obtain the public-key curve. Signing uses it to construct the commitment curve from the commitment ideal and, whenever the auxiliary branch is taken, applies it again after intersecting the auxiliary ideal with the commitment-response ideal.

To be more specific, let $I$ be the input ideal. $\mathsf{Qlapoti}$ returns $\beta_1 \in I$ and the odd integer

$$d_1 = \frac{\mathsf{nrd}(\beta_1)}{\mathsf{nrd}(I)}.$$

Before the theta chain, the current implementation applies $\beta_1 d_1^{-1}$ to $B_0$, where $d_1^{-1}$ is taken modulo $2^f$. Let $M_{\beta_1}$ denote the action matrix of $\beta_1$ on $E_0[2^f]$. The transformed basis $B' = (P', Q')$ is obtained by applying

$$M = d_1^{-1} M_{\beta_1} \in \mathrm{Mat}_2(\mathbb{Z}/2^f\mathbb{Z})$$

to $B_0$.

For $M = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$, we know from the bilinearity that

$$e_{2^f}(P', Q')^{d_1} = g_0^{d_1 \det(d_1^{-1} M_{\beta_1})} = g_0^{d_1^{-1} \det(M_{\beta_1})} = g_0^{d_1^{-1} \mathsf{nrd}(\beta_1)} = g_0^{\mathsf{nrd}(I)},$$

where the exponents are taken modulo $2^f$.

The current implementation computes $e_{2^f}(P', Q')^{d_1}$ before the theta chain and compares it with the pairing on the first output factor, since the two elliptic factors are not canonically ordered. The equality above replaces this transformed-basis pairing by $g_0^{\mathsf{nrd}(I)}$. The pairing on the output curve remains necessary for selecting the correct factor, but one complete pairing is removed from every ideal-to-isogeny call.

# B A Minor Integer-Representation Optimization for PRISM

For the PRISM application, we remark an additional minor change that is known for related integer-representation routines in the literature, but is not implemented in the current PRISM. Auxiliary ideals are generated during both key generation and signing by calling $\mathsf{RandomIdealGivenNorm}(N_{\mathsf{aux}})$, which invokes $\mathsf{RepresentInteger}(M)$ for a suitable target norm $M$. The dominant part of this routine searches for a prime residual and then applies Cornacchia's algorithm. The resulting representation may nevertheless be discarded only because Cornacchia returns its two coordinates in an incompatible order. This rejection can be avoided by exchanging the coordinates when necessary.

Concretely, the routine searches for an element of norm $M$ in the fixed extremal order $O_0 = \mathbb{Z}\langle 1, i, (i+j)/2, (1+k)/2\rangle$, where $k = ij$. Write $\delta = x + yi + zj + tk$ and $\gamma = \delta/2$. Since $\mathsf{nrd}(\delta) = x^2 + y^2 + p(z^2 + t^2)$, the representation problem reduces to

$$x^2 + y^2 + p(z^2 + t^2) = 4M. \tag{5}$$

Moreover, $\gamma = (x-t)/2 + (y-z)i/2 + z(i+j)/2 + t(1+k)/2$, so $\gamma \in O_0$ precisely when $x \equiv t \pmod 2$ and $y \equiv z \pmod 2$.

For sampled $z, t$, the routine sets $C = 4M - p(z^2 + t^2)$. When $C$ is prime and $C \equiv 1 \pmod 4$, Cornacchia's algorithm returns $x, y$ satisfying $x^2 + y^2 = C$. Exactly one of $x, y$ is odd. Since $p(z^2 + t^2) = 4M - C$ is also odd, exactly one of $z, t$ is odd. Therefore, exactly one of the orderings $(x, y)$ and $(y, x)$ satisfies $x \equiv t \pmod 2$ and $y \equiv z \pmod 2$. If the ordering returned by Cornacchia is incompatible (primitivity), swapping $x$ and $y$ produces an element of $O_0$ without changing $x^2 + y^2 = C$.

The resulting element is primitive as well. Its content in the fixed $O_0$-basis is $\mathsf{gcd}(x-t, y-z, 2z, 2t)$. All four entries are even, while one of $2z, 2t$ is congruent to 2 (mod 4), so the gcd is not divisible by 4. If an odd prime $\ell$ divided the gcd, then $\ell \mid z, t$ and consequently $\ell \mid x, y$, which would imply $\ell^2 \mid C$, contradicting the primality of $C$. Hence the content equals 2, and $\gamma = \delta/2$ is primitive in $O_0$.

**Algorithm 6:** PRISM integer representation with coordinate repair

**Input :** An odd target norm $M > p$
**Output:** A primitive $\gamma \in O_0$ satisfying $\mathsf{nrd}(\gamma) = M$

**repeat**
  Sample $z$ and $t$ from the intervals used by RepresentInteger
  $C \leftarrow 4M - p(z^2 + t^2)$
  **if** $C \leq 0$ *or* $C \not\equiv 1 \pmod 4$ **then**
    **continue**
  **if** ProbPrime$(C)$ = false **then**
    **continue**
  $(x, y) \leftarrow$ Cornacchia$(C)$
  **if** $(x, y) = \perp$ **then**
    **continue**
  **if** $x \not\equiv t \pmod 2$ **then** Swap $x$ and $y$
  $\gamma \leftarrow (x + yi + zj + tk)/2$
**until** *a valid representation has been found*
**return** $\gamma$

The modification replaces a post-Cornacchia rejection by one conditional coordinate exchange, without introducing another primality test or Cornacchia execution. The exchange may change the distribution of the intermediate element $\gamma$, but $\gamma$ is only used as a seed inside RandomIdealGivenNorm. Before returning the auxiliary ideal, the routine samples a uniform unit $u \in (O_0/N_{\mathsf{aux}}O_0)^\times$ and returns $O_0\langle \gamma u, N_{\mathsf{aux}} \rangle$. Since $N_{\mathsf{aux}}$ is odd, this right multiplication randomizes the local projective class at every prime power dividing $N_{\mathsf{aux}}$. Consequently, the distribution of the returned auxiliary ideal is independent of the ordering selected for $x$ and $y$.