AVX-512 Acceleration of SQIsign 3.0 Verification

Yunqi Wen[1], Xuzhe Liu[1], Zhi Hu[1*], Saiyu Wang[2], Chang-An Zhao[2]

[1]School of Mathematics and Statistics, Central South University, Changsha, China.
[2]School of Mathematics, Sun Yat-Sen University, Guangzhou, China.

*Corresponding author(s). E-mail(s): huzhi_math@csu.edu.cn;
Contributing authors: zhaochan3@mail.sysu.edu.cn;

Abstract

SQIsign is a compact isogeny-based signature scheme. We present a low-latency AVX-512 implementation of complete SQIsign 3.0 verification at security levels I, III, and V. Our implementation provides vectorized curve-side verification, including the Superglue transition, and introduces dynamic-programming strategies for the degree-$\mathbf{2^\ell}$ and theta-model $\mathbf{(2,2)}$-isogeny chains. The strategies account for batched point evaluation and the preparation of SIMD data. On an Intel Core i7-11800H running native Ubuntu 24.04, the complete verifiers achieve **2.14**×, **2.08**×, and **2.34**× speedups, respectively, over the SQIsign 3.0 Broadwell reference verifiers (the optimized 64-bit assembly implementation).

Keywords: SQIsign, AVX-512, dynamic programming, isogeny chains, verification

1 Introduction

The SQIsign scheme combines supersingular elliptic curves and quaternion algebras to obtain compact isogeny-based signatures [1]. The latest SQIsign 3.0 version increases its parameter sizes in response to Wesolowski's attack [2]. Its curve-side computation uses arithmetic over a large quadratic extension field, elliptic-curve operations, and long smooth-degree isogeny chains, whereas the quaternion–ideal layer uses bounded multi-precision integers. SQIsign 3.0 stores these integers in fixed-capacity representations, but its key-generation and signing implementation remains variable-time. In this work we focus on exploiting fast implementation for public verification, whose arithmetic and isogeny computations are entirely curve-side.

This paper implements public verification at security levels I, III, and V of SQIsign 3.0 [1]. Verification reconstructs public challenge-side data, recovers the commitment-side isogeny computation, hashes the resulting public values, and performs the final comparison. Its fixed-size field, curve, and public-chain computations provide a concrete target for AVX-512 acceleration over the released verifier's Broadwell reference backend.

In addition, we parallelize curve-side computations in SQIsign 3.0 key generation and signing. These operations retain variable-time computations; implementation details and benchmarks for these extensions are presented in Appendix A.

AVXSIKE [3] provides radix-2^{51} field arithmetic and SIMD organization for Montgomery-curve and isogeny computations. SQIsign on ARM [4] combines static lazy reduction with vectorized elliptic-curve and theta-coordinate arithmetic. For SQIsign 2.0, Wang et al. [5] develop an end-to-end AVX-512 implementation covering finite-field, elliptic-curve, pairing, and isogeny computations. Their work demonstrates the effectiveness of reorganizing arithmetic dependencies and batching higher-level operations, while maintaining a common vector representation across much of the curve-side computation. These results highlight the importance of algorithm-level SIMD scheduling beyond the optimization of individual field kernels.

Targeting SQIsign 3.0 requires parameter-specific reduction bounds and revised parallel groupings for its updated curve-side computations. Challenge recovery uses a degree-2^ℓ isogeny chain with terminal curve normalization; commitment recovery uses a theta-model $(2,2)$-isogeny chain with an x-only pre-gluing ladder and Superglue. We redesign parallel groupings and data reuse across these computations, and match public chain strategies to their SIMD costs to minimize single-verification latency.

The contributions of this work are as follows:

- We adapt and reorganize prime- and extension-field arithmetic for all three SQIsign 3.0 security levels. Following the radix-2^{51} SIMD approach of AVXSIKE [3] and the static lazy-reduction principle used in SQIsign on ARM [4], we specialize the reduction bounds and reorganize multiplication, reduction, and data packing for the new parameter sets. The resulting arithmetic supplies compatible layouts for successive elliptic-curve and theta-coordinate operations.
- We design AVX-512 parallel implementations of the higher-level curve-side arithmetic required by complete SQIsign 3.0 verification, covering elliptic-curve and basis operations as well as one- and two-dimensional isogeny computations. We adapt established vectorization techniques for existing primitives [3, 4] and develop SIMD schedules for computations introduced or revised in SQIsign 3.0, including the x-only pre-gluing ladder, Superglue, and updated generic theta formulas [1, 6]. These designs organize parallel work across coordinates, elliptic factors, and saved points, while reusing packed data and common coefficients across successive operations.
- We extend established ordered-tree isogeny strategies [7, 8] with cost models that connect chain scheduling to the AVX-512 execution state. The one-dimensional model incorporates terminal curve normalization and batched saved-point evaluation. The two-dimensional model jointly optimizes pre-gluing descent, Superglue, and generic theta isogenies, distinguishing on-demand mathematical precomputation

from reusable SIMD preparation. This reformulation captures how strategy choices change both evaluation batches and the preparation costs incurred along the chain.

On an Intel Core i7-11800H, the complete Level-I, Level-III, and Level-V verifiers achieve 2.14×, 2.08×, and 2.34× speedups over the corresponding SQIsign 3.0 Broadwell reference verifiers. The DP schedules reduce full-verification cycles by a further 1.69%, 1.84%, and 1.40% relative to the AVX-512 reference-style schedules, respectively.

The remainder of this paper is organized as follows. Section 2 reviews SQIsign verification and AVX-512. Section 3 presents the finite-field arithmetic and layouts. Section 4 describes the vectorized degree-2^ℓ isogeny chain and its dynamic program. Section 5 presents the theta-model $(2,2)$-isogeny chain, including Superglue and its joint strategy optimization. Section 6 reports performance results, and Section 7 concludes.

2 Preliminaries

This section reviews the SQIsign workflow specified in the latest public reference release [1], then introduces the AVX-512 execution model and the SIMD notation used throughout the paper.

2.1 SQIsign Protocol Workflow

SQIsign applies the Fiat–Shamir transform [9] to an isogeny-based Σ-protocol. Its public parameters contain a prime p, a supersingular curve $E_0/\mathbb{F}_{p^2}$, and its known endomorphism-ring description. Key generation samples a secret isogeny $\varphi_{\mathsf{sk}} : E_0 \to E_{\mathrm{pk}}$ and publishes E_{pk} with a compact basis hint. For a signature, the prover chooses an ephemeral isogeny $\varphi_{\mathrm{com}} : E_0 \to E_{\mathrm{com}}$; hashing the public transcript and message determines the degree-2^λ challenge isogeny $\varphi_{\mathrm{chl}} : E_{\mathrm{pk}} \to E_{\mathrm{chl}}$, where $\lambda = 128, 192, 256$ at Levels I, III, and V, respectively. A compact representation of the odd-degree response isogeny $\varphi_{\mathrm{rsp}} : E_{\mathrm{chl}} \to E_{\mathrm{com}}$ connects the two sides without revealing φ_{sk}.

Figure 1 gives the high-level relation between the corresponding isogenies.

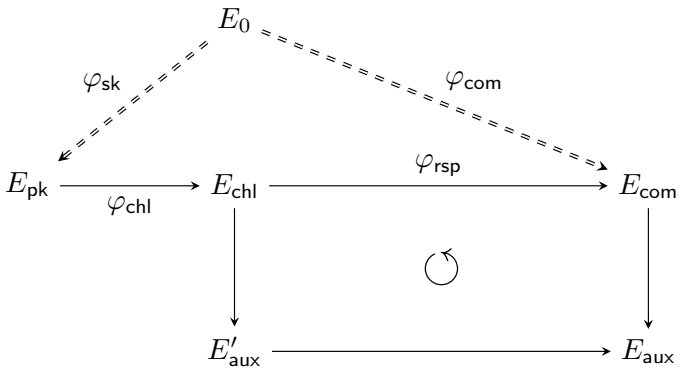

Fig. 1 SQIsign 3.0 signature construction [1, Fig. 3].

SQIsign has distinct quaternion–ideal and curve-side domains. Key generation and signing use the Deuring correspondence and bounded multi-precision integer

arithmetic. Their SQIsign 3.0 implementation is still variable-time; public verification is entirely curve-side, using fixed-size elements of $\mathbb{F}_{p^2}$, elliptic curves, torsion bases, and isogenies. This work accelerates only that curve-side verification branch. SQIsign2D-East and SQIsign2D-West organize related higher-dimensional response computations differently [10–12]; our target is the verification path of the current SQIsign reference specification.

Compared with SQIsign 2.0, SQIsign 3.0 shortens and fixes the challenge chain and restricts the response to odd degree, removing backtracking and the additional short 2-power response chain. Both the challenge and theta-chain lengths are therefore fixed for each parameter set [1].

Verification path. The verifier decodes the public key and signature and checks their structural validity. From the public-key basis hint and challenge, it forms a kernel and computes the challenge isogeny to obtain E_{chl}. It then recovers bases on E_{chl} and the auxiliary curve E_{aux} from the supplied hints and applies the response basis-change matrix. The paired basis points specify the kernel of a theta-model $(2,2)$-isogeny chain from $E_{\mathrm{chl}} \times E_{\mathrm{aux}}$ to a product containing E_{com}. Relative to SQIsign 2.0, the Jacobian preparation and gluing computation are replaced by an x-only pre-gluing ladder and Superglue [1, 6]; updated generic theta formulas and splitting then recover E_{com}. Finally, domain-separated SHAKE256 hashes the encoded public key, normalized commitment-curve coefficient, and message, replacing the j-invariant-based iterated hash of SQIsign 2.0. Verification succeeds when the resulting challenge matches the signature and all structural checks pass. Sections 4 and 5 describe the one- and two-dimensional chain computations.

2.2 The AVX-512 Instruction Set and SIMD Organization

This work uses AVX-512 instructions. The architecture provides thirty-two 512-bit `ZMM` registers, each containing eight 64-bit lanes. These lanes provide the physical storage used by the field and higher-level layouts; they do not necessarily represent eight independent protocol instances. The multiply-add pair `VPMADD52LUQ`/`VPMADD52HUQ` accumulates the low and high halves of products of 52-bit unsigned operands. Section 3 uses these instructions in a radix-51 Montgomery schedule and gives the associated carry and reduction rules.

Table 1 lists the 512-bit instruction classes used by the implementation. Latency and reciprocal throughput are microarchitecture-specific; the table reports the register-only Tiger Lake measurements of uops.info [13].

SIMD organization. Following the nested SIMD organization of AVXSIKE [3], a vectorized curve or theta kernel is denoted by a $(w \times x \times y \times z)$-way schedule: w is the number of independent curve- or theta-level groups, x the number of $\mathbb{F}_{p^2}$ operations in each group, y the exposed $\mathbb{F}_p$ product, square, or coordinate slots, and z the 64-bit lanes used by each prime-field slot. When all eight lanes of a ZMM register are occupied, $wxyz = 8$. This notation describes parallelism inside one verification, not a batch of independent signatures.

For field kernels, $w = 1$ is implicit: $(8 \times 1 \times 1)$ evaluates eight independent $\mathbb{F}_{p^2}$ operations, whereas $(4 \times 2 \times 1)$ evaluates four operations with two exposed prime-field

Table 1 Relevant 512-bit AVX-512 instructions. Latency and reciprocal throughput (CPI) are Tiger Lake measurements reported by uops.info.

Class	Mnemonic	Role in this implementation	Latency	CPI
Multiply-add	`VPMADD52LUQ`	low product accumulation	4	1.00
Multiply-add	`VPMADD52HUQ`	high product accumulation	4	1.00
Arithmetic	`VPADDQ/VPSUBQ`	limb addition and subtraction	1	0.50
Shift	`VPSLLQ/VPSRLQ`	carry alignment and extraction	1	1.00
Logic	`VPANDD/VPXORD`	limb masks and zeroing	1	0.50
Permutation	`VPERMQ/VPERMI2Q/VPERMT2Q`	pack, unpack, and transpose	3	1.00
Mask	`VPBLENDMQ`	lane-wise selection	1	0.50

slots each. The concrete radix-51 representations and lane packings are specified in Section 3.

3 AVX-512 Finite-Field Arithmetic and Layouts

We provide AVX-512 arithmetic for all three SQIsign 3.0 parameter sets. With radix $B = 2^{51}$, write $p = KB^{n-1} - 1$ and $R = B^n$. Table 2 lists the constants for each level; the limb counts are $n = 7, 10, 14$, respectively. The following concrete representation uses Level III as an example; the SIMD organization is shared across levels, with parameter-specific limb counts and reduction bounds. A layout-aware cost includes field arithmetic and the packing, permutation, padding, and unpacking required by the chosen layout.

Table 2 Radix-2^{51} parameters for SQIsign 3.0.

Level	K	p	R
I	$3 \cdot 2^{18}$	$3 \cdot 2^{324} - 1$	B^7
III	$27 \cdot 2^{41}$	$27 \cdot 2^{500} - 1$	B^{10}
V	34	$17 \cdot 2^{664} - 1$	B^{14}

3.1 Radix-2^{51} Prime-Field Arithmetic

For the Level-III example, an element $x \in \mathbb{F}_p$ is stored in Montgomery form $xR \bmod p$ as ten radix-B limbs,

$$x = \sum_{i=0}^{9} x_i B^i, \qquad 0 \le x_i < B.$$

For a vector X of eight field elements, limb i is stored in one ZMM register as

$$X_i = (x_i^{(0)}, x_i^{(1)}, \ldots, x_i^{(7)}).$$

Thus the representation is structure-of-arrays across field elements and array-of-limbs within one field element. It uses the eight 64-bit lanes of a ZMM register directly; each

51-bit limb is a valid input to `VPMADD52LUQ` and `VPMADD52HUQ`, while the unused high bit gives headroom for additions and carries.

(8×1)-way prime-field layout and linear arithmetic. The underlying prime-field work is the $(y \times z)$ projection of the SIMD organization defined in Section 2.2: eight independent $\mathbb{F}_p$ operations occupy one lane each. The representation above is therefore an (8×1)-way prime-field layout. The canonical addition and subtraction routines perform radix-B carry or borrow propagation and a mask-selected correction by p. The higher layers more commonly use the range-preserving variants: they add limbs directly, or subtract after adding $2p$, $4p$, or $8p$ as a public constant. These variants maintain the corresponding $[0, 4p)$, $[0, 8p)$, and $[0, 16p)$ bounds without an unnecessary strong reduction. Extension-field addition and subtraction apply the same prime-field operation independently to the real and imaginary coordinates.

Streaming prime-field Montgomery product. The choice of radix is aligned with the special form of the prime:

$$p = KB^9 - 1 = (B-1) + (B-1)B + \cdots + (B-1)B^8 + (K-1)B^9.$$

Thus the reduction code uses the compact constant K. In particular, $p \equiv -1 \pmod{B}$, so the Montgomery digit for a current coefficient t_i is simply $m_i = t_i \bmod B$. Adding $m_i p = m_i K B^9 - m_i$ cancels the current low coefficient and contributes only the delayed term $m_i K$ at coefficient $i + 9$.

A prime-field multiplication consequently uses a streaming Comba–REDC schedule: it scans product coefficients in increasing radix order and retains $m_0, \ldots, m_9$ only until their delayed uses, instead of storing a full product and a separate Montgomery-reduction (REDC) array. Each accumulator update in Algorithm 1 is one paired `VPMADD52LUQ`/`VPMADD52HUQ` update. Since radix-B carry propagation is by 2^{51}, the high accumulator contributes as $2h$. The live state consists of the low and high accumulators, one carry, and the ten delayed digits, without a separate product array.

Squaring and extension-field reduction. Prime-field squaring uses the same procedure, except that the Comba scan doubles off-diagonal products and inserts the diagonal squares once. It therefore uses the identical delayed $m_i K$ reduction and output range. The extension-field kernels preserve these range contracts and use layout-specific product schedules described below. In particular, the principal $(8 \times 1 \times 1)$ multiplier is a three-product Karatsuba kernel.

Inversion and square roots. When all eight lanes are active, inversion and square root use fixed constant-time addition chains. For $a = a_{\mathrm{re}} + a_{\mathrm{im}} i$, extension-field inversion uses $\overline{a}/(a_{\mathrm{re}}^2 + a_{\mathrm{im}}^2)$, while square root uses norm-based reconstruction followed by verification by squaring. At a public sparse boundary with fewer than eight required lanes, the active values use the corresponding SQIsign 3.0 Broadwell scalar implementation instead of padded exponentiation chains. Conversions at this boundary preserve the field value and use the Montgomery representation of the selected parameter set.

Algorithm 1 FpMul8_1v3(A, D)

Require: $A, D \in [0, 4p)$ in carry-normalized radix-B Montgomery form, $B = 2^{51}$, $p = KB^9 - 1$
Ensure: $C = ADR^{-1} \bmod p$ in $[0, 2p)$, for eight lanes in parallel

$carry \leftarrow 0$; $m[0], \ldots, m[9] \leftarrow 0$
for $t \leftarrow 0$ to 19 **do**
  $(\ell, h) \leftarrow (carry, 0)$
  for all j with $0 \le j < 10$ and $0 \le t - j < 10$ **do**
    $(\ell, h) \leftarrow (\ell + (A_j D_{t-j} \bmod 2^{52}),\ h + \lfloor A_j D_{t-j}/2^{52} \rfloor)$ ▷ one `VPMADD52LUQ/HUQ` pair
  end for
  if $9 \le t \le 18$ **then**
    $(\ell, h) \leftarrow (\ell + (m[t-9]K \bmod 2^{52}),\ h + \lfloor m[t-9]K/2^{52} \rfloor)$ ▷ one `VPMADD52LUQ/HUQ` pair
  end if
  $d \leftarrow \ell \bmod B$
  $carry \leftarrow \lfloor \ell/B \rfloor + 2h$
  if $t < 10$ **then**
    $m[t] \leftarrow d$ ▷ d cancels the current low coefficient
  else
    $C_{t-10} \leftarrow d$
  end if
end for
return C

3.2 Extension-Field Layouts

We represent

$$\mathbb{F}_{p^2} = \mathbb{F}_p[i], \qquad i^2 = -1,$$

and write $a = a_{\mathrm{re}} + a_{\mathrm{im}} i$. Here i is an extension-field generator satisfying $i^2 = -1$. In $a^{(\ell)}_{\mathrm{re},j}$, j indexes the limbs and ℓ indexes the SIMD lanes. A multiplication evaluates

$$ab = (a_{\mathrm{re}} b_{\mathrm{re}} - a_{\mathrm{im}} b_{\mathrm{im}}) + (a_{\mathrm{re}} b_{\mathrm{im}} + a_{\mathrm{im}} b_{\mathrm{re}}) i.$$

The verifier selects a layout from the independent values exposed at each call site. For field kernels the outer factor $w = 1$ is implicit in the $(w \times x \times y \times z)$ convention of Section 2.2. The $(x \times y \times z)$-way notation specifies the lane organization used for multiplication and squaring. Long-lived values retain the layout consumed by their next operation.

$(8 \times 1 \times 1)$ layout. For eight independent extension-field values, the real and imaginary components are held in separate prime-field vectors:

$$(a^{(0)}_{\mathrm{re}}, \ldots, a^{(7)}_{\mathrm{re}}), \qquad (a^{(0)}_{\mathrm{im}}, \ldots, a^{(7)}_{\mathrm{im}}).$$

This layout supports componentwise evaluation of eight independent points under the same isogeny. Its multiplication uses Karatsuba:

$$P_0 = a_{\mathrm{re}} b_{\mathrm{re}}, \qquad P_1 = a_{\mathrm{im}} b_{\mathrm{im}},$$
$$P_s = (a_{\mathrm{re}} + a_{\mathrm{im}})(b_{\mathrm{re}} + b_{\mathrm{im}}).$$

The implementation first writes the three unreduced product scans to three 21-coefficient vector buffers, then forms $(P_0 - P_1,\ P_s - P_0 - P_1)$ coefficient-wise and applies two signed Montgomery reductions. It therefore trades the product storage for one fewer prime-field product scan than schoolbook multiplication. Squaring computes

$$a^2 = (a_{\mathrm{re}}^2 - a_{\mathrm{im}}^2) + (2a_{\mathrm{re}}a_{\mathrm{im}})i,$$

accumulating the two signed square contributions to the real coefficient and twice the cross product to the imaginary coefficient. Diagonal square terms are inserted once and off-diagonal terms twice. Both coefficient streams are reduced as they are formed.

$(4 \times 2 \times 1)$ layout. For four extension-field values, each limb is interleaved as

$$(a_{\mathrm{re}}^{(0)}, a_{\mathrm{im}}^{(0)}, a_{\mathrm{re}}^{(1)}, a_{\mathrm{im}}^{(1)}, a_{\mathrm{re}}^{(2)}, a_{\mathrm{im}}^{(2)}, a_{\mathrm{re}}^{(3)}, a_{\mathrm{im}}^{(3)}).$$

The adjacent real–imaginary coordinates make this layout suitable for four-point curve and theta operations. Its native multiplication kernel streams the interleaved product coefficients of each adjacent complex pair and recombines them once per coefficient before reduction. Squaring uses $(a_{\mathrm{re}}+a_{\mathrm{im}})(a_{\mathrm{re}}-a_{\mathrm{im}}+4p)$ in the real slots and $(2a_{\mathrm{re}})a_{\mathrm{im}}$ in the imaginary slots. The $4p$ offset makes the difference nonnegative without changing its residue. Lane permutations and premixing supply these operands directly in the interleaved layout.

Packing, transposition, and lazy reduction. A transpose changes only lane placement and is used at boundaries between the $(8 \times 1 \times 1)$ and $(4 \times 2 \times 1)$ forms. Premixing forms the real–imaginary sums or differences required by a product schedule, and postmixing restores its two extension-field coordinates. A layout is *resident* when consecutive kernels consume and produce the same packing; formulas exposing only two $\mathbb{F}_{p^2}$ values use the compact $(2 \times 4 \times 1)$ boundary layout.

Lazy reduction retains bounded, noncanonical representatives between nonlinear layers. If $0 \le T$ and the Montgomery reduction uses $0 \le m < R$, then

$$\mathrm{REDC}(T) = \frac{T + mp}{R} < \frac{T}{R} + p.$$

Consequently, inputs $a, b < cp$ give $\mathrm{REDC}(ab) < (1 + c^2 p/R)p$. The ratios for our parameters satisfy

$$\frac{p}{R} < \begin{cases} 3/2^{33}, & \text{Level I,} \\ 27/2^{10}, & \text{Level III,} \\ 17/2^{50}, & \text{Level V.} \end{cases}$$

For $c = 4$, even the Level-III bound is $(1 + 16 \cdot 27/1024)p = 91p/64 < 2p$. Thus the uncorrected prime-field core in Algorithm 1 accepts $[0, 4p)$ inputs at all three levels. Squaring has the same bound. The general prime-field interface instead accepts $[0, 8p)$ inputs and returns $[0, p)$. REDC then gives a value below $2p$ at Levels I and V, and below $43p/16 < 3p$ at Level III. Canonicalization uses one or two conditional subtractions of p, respectively, implemented by fixed instruction sequences and masks.

Table 3 Level-III finite-field arithmetic for SQIsign 3.0. Costs are median cycles; AVX-512 uses radix-51 and the reference uses the Broadwell backend.

Operation	Instance	Impl.	Vectorization	#Inst.	Cycles	Cyc./inst.	Speed-up
$\mathbb{F}_p$ modular multiplication	`fp_mul`	ref. (64-bit asm.)	1-way	1	99.15	99.15	1.00×
	`fp_mul8x1w`	AVX-512	(8×1)-way	8	332.63	41.58	2.38×
$\mathbb{F}_p$ modular squaring	`fp_sqr`	ref. (64-bit asm.)	1-way	1	99.15	99.15	1.00×
	`fp_sqr8x1w`	AVX-512	(8×1)-way	8	156.86	19.61	5.06×
$\mathbb{F}_{p^2}$ multiplication	`fp2_mul`	ref. (64-bit asm.)	1-way	1	377.66	377.66	1.00×
	`fp2_mul_8x1x1w`	AVX-512	$(8 \times 1 \times 1)$-way	8	870.48	108.81	3.47×
	`fp2_mul_4x2x1w`	AVX-512	$(4 \times 2 \times 1)$-way	4	491.42	122.86	3.07×
	`fp2_mul_2x4x1w`	AVX-512	$(2 \times 4 \times 1)$-way	2	362.30	181.15	2.08×
$\mathbb{F}_{p^2}$ squaring	`fp2_sqr`	ref. (64-bit asm.)	1-way	1	220.58	220.58	1.00×
	`fp2_sqr_8x1x1w`	AVX-512	$(8 \times 1 \times 1)$-way	8	552.89	69.11	3.19×
	`fp2_sqr_4x2x1w`	AVX-512	$(4 \times 2 \times 1)$-way	4	359.68	89.92	2.45×
	`fp2_sqr_2x4x1w`	AVX-512	$(2 \times 4 \times 1)$-way	2	369.15	184.57	1.20×

Extension-field multiplication combines the raw coefficients before REDC. The Karatsuba real numerator is shifted by pR only when negative; both it and the imaginary numerator reduce into $[0, 2p)$. The Level-III implementation retains a conservative weak subtraction of $2p$ on the imaginary result. Signed-product and premixed-square kernels use their corresponding fixed corrections to obtain the same output bound. Thus packed multiplication and squaring accept $[0, 4p)$ inputs and return $[0, 2p)$ outputs. Linear operations may temporarily use wider ranges; carry propagation and weak reduction restore the next kernel's input bound. Canonical residues are formed when encoding or testing values.

The physical layouts above include one vector per radix limb. In later algorithms we suppress that limb index and display only the logical $\mathbb{F}_{p^2}$ order. Thus M_8 and S_8 denote complete $(8 \times 1 \times 1)$-way multiplication and squaring layers, whereas M_4 and S_4 denote their $(4 \times 2 \times 1)$-way counterparts.

3.3 Field Benchmarks

Table 3 compares Level-III field arithmetic with the SQIsign 3.0 Broadwell backend on the platform of Section 6. `Impl.` distinguishes the reference from AVX-512, `#Inst.` gives the independent operations per call, and `Cyc./inst.` divides the call cost accordingly. The `fp_mul8x1w` and `fp_sqr8x1w` rows measure the carry-normalized $[0, 4p) \to [0, 2p)$ streaming Comba–REDC core before the final canonical correction. The reference `fp_sqr` dispatches to `fp_mul`, so both reference entries use the same multiplication-kernel cost.

The $(8 \times 1 \times 1)$ kernels minimize per-instance cost for eight resident values, while $(4 \times 2 \times 1)$ serves four-point curve and theta operations; $(2 \times 4 \times 1)$ is used at narrow boundaries. The retained layout is selected by the next operation in the dataflow.

4 Vectorized Degree-2^ℓ Isogeny Chain

SQIsign 3.0 verification recovers the challenge curve through a degree-2^ℓ isogeny chain. We first describe its vectorized operations and chain evaluation, then derive a public strategy using their SIMD costs.

4.1 Vectorized Chain Computation

Montgomery x-only arithmetic. The challenge isogeny has $\ell = 128, 192, 256$ at Levels I, III, V. Its kernel representative K carries two extra torsion bits: it has order $2^{\ell+2}$, and the isogeny has kernel $\langle [4]K \rangle$. Kummer points use $(X : Z)$ coordinates on $E_{A:C}$ [14, 15]. The curve constants $(C_{24}, A_{24}^+) = (4C, A + 2C)$ are prepared once for each curve. The operation $\mathsf{xDBL}(K; E)$ applies a Hadamard layer, paired squarings and paired multiplications to compute $[2]K$; $\mathsf{xDBL}^j(K; E)$ means j successive doublings. The duplicated two-coordinate layout preserves X, Z across these calls and uses the $(2 \times 4 \times 1)$ nonlinear kernels. Persistent inputs remain in $[0, 2p)$, with the temporary bounds described in Section 3.2.

Isogeny construction and batched evaluation. The helper $\mathsf{xISOG}_4(E, K_4)$ takes an order-4 kernel and returns the degree-4 codomain E' and its kernel-point structure $\boldsymbol{c}_4$, comprising three coefficient vectors. The evaluator $\mathsf{xEVAL}_4(\mathcal{F}, \boldsymbol{c}_4)$ applies that isogeny to every point of the saved frontier $\mathcal{F}$. A frontier is the ordered list of points whose images are needed by later steps. This verification call supplies no external image points.

For $s = 8a + r$ saved points, a complete eight-point blocks are followed by no block when $r = 0$, a one-point block when $r = 1$, a two-point block when $r = 2$, a four-point block when $r = 3, 4$, or an eight-point block when $r = 5, 6, 7$. Unused slots repeat the last real input, and only the real outputs are retained. The eight-point form uses coordinate arrays, the four-point form interleaves coordinates, and the narrower forms preserve the paired or duplicated layouts. The one-point coefficient form is prepared only when this public decomposition requires it.

Terminal curve normalization. The $L = \ell/2$ degree-4 isogenies leave the image of K with order 4. SQIsign 3.0 uses this point for canonical curve recovery rather than applying another degree-4 isogeny. Write this operation as $\mathsf{NormalizeCurve}(X : Z)$ [1]. It forms the three pairs

$$(X : Z), \quad (X + Z : X - Z), \quad (Z - iX : X - iZ)$$

and, for each pair $(x : z)$, computes $A = 2(x^4 + z^4)/(x^4 - z^4)$, rejecting a zero denominator. It selects the largest of these three coefficients and their negatives under the reference ordering: canonical imaginary component first, then real component. The three candidate computations are packed into eight lanes with repetitions, and the final comparison uses canonical encodings. The returned Montgomery curve is $E_{A:1}$. This same recovery operation is used after the final theta splitting.

Algorithm 2 writes the strategy recursively for clarity. Its root has $n = L + 1$, an empty frontier, and $b_{\text{end}} = \mathsf{true}$. The additional terminal represents curve normalization.

A subtree with $b_{\text{end}} = \mathsf{false}$ consists entirely of degree-4 isogenies. The preorder cut program and its next-entry index k determine the left-first traversal.

Algorithm 2 TwoIsogChain
$(E, K, \mathcal{F}, n, b_{\text{end}}; \mathsf{cut}, k)$

Require: Curve E, kernel representative K of order 4^n, saved frontier $\mathcal{F}$, terminal flag b_{end}, and preorder cut program
Ensure: Curve, required frontier images, and updated program index

if $n = 1$ **then**
  if $b_{\text{end}} = \mathsf{true}$ **then**
    return $(\mathsf{NormalizeCurve}(K), \mathcal{F}, k)$
  end if
  $(E, \boldsymbol{c}_4) \leftarrow \mathsf{xISOG}_4(E, K)$
  $\mathcal{F} \leftarrow \mathsf{xEVAL}_4(\mathcal{F}, \boldsymbol{c}_4)$
  return $(E, \mathcal{F}, k)$
end if
$i \leftarrow \mathsf{cut}[k]$; $k \leftarrow k + 1$
$K_{\text{L}} \leftarrow \mathsf{xDBL}^{2i}(K; E)$
$\mathcal{F}_{\text{L}} \leftarrow (K; \mathcal{F})$ ▷ save the parent kernel first
$(E', \mathcal{F}'_{\text{L}}, k) \leftarrow$
  $\mathsf{TwoIsogChain}(E, K_{\text{L}}, \mathcal{F}_{\text{L}}, n - i, \mathsf{false}; \mathsf{cut}, k)$
$(K_{\text{R}}; \mathcal{F}_{\text{R}}) \leftarrow \mathcal{F}'_{\text{L}}$
return $\mathsf{TwoIsogChain}(E', K_{\text{R}}, \mathcal{F}_{\text{R}}, i, b_{\text{end}}; \mathsf{cut}, k)$

4.2 Dynamic Programming

An ordered strategy tree specifies when a kernel is saved, doubled, and evaluated. Its left child executes first while retaining the parent kernel; the image of that kernel then starts the right child. For n remaining terminals, the SQIsign 3.0 reference chooses $i = \lfloor n/2 \rfloor$, giving subtrees of sizes $n - i$ and i after $2i$ doublings [1]. AVX-512 batching changes the relative costs of descent and saved-point evaluation, so we select cuts using the vectorized operations of Section 4.1.

Optimization problem. Following the ordered isogeny-tree framework of De Feo, Jao, and Plût [7], we minimize the cost of descent and propagation of saved kernels. Table 4 defines the state and its costs. A regular leaf includes the complete degree-4 construction and frontier evaluation, including the coefficient layout required by its public block decomposition. Thus changing a cut changes the batch sizes and layout costs incurred at subsequent leaves.

Boundaries and ordered recurrence. The terminal flag can be true only at $s = 0$: no saved ancestor survives the final normalization. A legal state has $n \geq 1$ and $0 \leq s < S$. Its leaf cost is

$$\mathcal{C}(1, s, b_{\text{end}}) = \begin{cases} \text{NORMcost}, & b_{\text{end}} = \mathsf{true},\ s = 0, \\ \text{LEAFcost}_4(s), & b_{\text{end}} = \mathsf{false}. \end{cases}$$

Table 4 Notation for the degree-2^ℓ chain strategy.

Symbol	Meaning
ℓ	Challenge exponent, $128, 192, 256$ at Levels I, III, V.
L	Number of degree-4 isogenies, $\ell/2$.
n	Number of remaining tree terminals; the root has $L+1$ including normalization.
s	Number of saved ancestor kernels, hence the evaluation-frontier size.
b_{end}	True when the subtree contains the final curve-normalization terminal.
i	Ordered cut: $n-i$ terminals in the left child, i in the right child.
$\text{DBLcost}_4(i)$	Cost of $2i$ consecutive doublings in the resident one-point layout.
$\text{LEAFcost}_4(s)$	Degree-4 construction and full evaluation of s saved points under the stated 1/2/4/8 block rule.
NORMcost	Final four-torsion-point-to-curve normalization cost.
S	Stack capacity, 16 frames including the current kernel.
$\mathcal{C}(n, s, b_{\text{end}})$	Minimum cost of the indicated ordered subtree.

For $n > 1$, a cut requires $s + 1 < S$, and

$$\mathcal{C}(n, s, b_{\text{end}}) = \min_{1 \le i < n} \big[\text{DBLcost}_4(i) + \mathcal{C}(n-i, s+1, \mathsf{false}) + \mathcal{C}(i, s, b_{\text{end}}) \big].$$

Illegal states have cost $+\infty$. The left child charges all evaluations of the retained parent kernel; its resulting image starts the right child. The root is $\mathcal{C}(L+1, 0, \mathsf{true})$, with $L+1 = 65, 97, 129$. Algorithm 3 stores the minimizing cuts, which are then emitted in preorder for Algorithm 2.

Algorithm 3 $\text{DP}_{2^\ell}(n, s, b_{\text{end}})$

Require: State (n, s, b_{end}) and costs from Table 4
Ensure: Minimum cost and minimizing cut
if $n < 1$ or $s \ge S$ or ($b_{\text{end}} = \mathsf{true}$ and $s \neq 0$) **then**
  return $(+\infty, 0)$
end if
if this state is memoized **then**
  return its stored cost and cut
end if
if $n = 1$ **then**
  if $b_{\text{end}} = \mathsf{true}$ **then**
    Store and return $(\text{NORMcost}, 0)$
  else
    Store and return $(\text{LEAFcost}_4(s), 0)$
  end if
end if

```
mincost ← +∞; mincut ← 0
if s + 1 < S then
    for i ← 1 to n − 1 do
        (Lcost, −) ← DP_{2^ℓ}(n − i, s + 1, false)
        (Rcost, −) ← DP_{2^ℓ}(i, s, b_end)
        c_i ← DBLcost_4(i) + Lcost + Rcost
        if c_i < mincost then
            (mincost, mincut) ← (c_i, i)
        end if
    end for
end if
Store and return (mincost, mincut) for this state
```

5 Vectorized Theta-Model (2, 2)-Isogeny Chain

Commitment recovery computes one gluing transition and $N-1$ generic $(2,2)$-isogenies, with $N = 196,302,400$ at Levels I, III, V. The inputs are two elliptic bases; final splitting recovers the commitment curve. Every algorithm returns $\perp$ on a failed validity check, and callers propagate that failure.

Table 5 records the packed representations used throughout this chapter. The symbols $\mathsf{M}_4, \mathsf{S}_4, \mathsf{M}_8, \mathsf{S}_8$ retain their Section 3.2 meanings. Four-coordinate vectors and 2×2 matrices use zero-based indices; $r, j \in \{1, 2\}$ index a product point and its elliptic factor. The four-coordinate Hadamard transform is

$$\begin{aligned}\mathsf{H}_4(x, y, z, t) = (&x + y + z + t,\ x - y + z - t,\\ &x + y - z - t,\ x - y - z + t).\end{aligned}$$

We also use projective coordinate inversion,

$$\mathsf{InvertPoint}(x, y, z, t) = (yzt, xzt, xyt, xyz).$$

For nonzero coordinates this is proportional to $(1/x, 1/y, 1/z, 1/t)$. It is computed without field inversion: first form xy and zt, then multiply (y, x, t, z) by (zt, zt, xy, xy) in one M_4 layer.

Table 5 Packed representation and coordinate notation.

Symbol	Meaning
$P_1 \mid P_2$	Two independently evaluated theta points. Gluing stores them as $[x_1, x_2, y_1, y_2, z_1, z_2, t_1, t_2]$; generic pair kernels use $[x_1, y_1, z_1, t_1, x_2, y_2, z_2, t_2]$.
$X_{r,j}, Z_{r,j}$	Four elliptic slots ordered $(1,1), (1,2), (2,1), (2,2)$ during the gluing ladder and barycentric calculation.
$\star$	Coordinate-wise multiplication. A two-value pair uses the $(2 \times 4 \times 1)$ kernel.
$\mathsf{H}_{4\times 2}$	Independent H_4 transforms on the two theta points, in their current physical ordering.
$C^{(2)}$	The tuple C supplied to both points, written $C \mid C$.
$\mathbf{M}_{:,k}$	Column k of the four-by-four change-of-coordinate matrix.
$\mathbf{1}_q$	A length-q all-one vector; $a\mathbf{1}_q$ repeats a.

5.1 Pre-gluing Ladder and Superglue

Compared with the gluing computation in SQIsign 2.0, SQIsign 3.0 adopts Duparc's Superglue formulas, using x-only data and barycentric coordinates in place of the earlier Jacobian-coordinate preparation [1, 6]. We implement this transition with AVX-512 arithmetic.

x-only pre-gluing ladder. Write $B_j = (P_j, Q_j, P_j - Q_j)$ on factor E_j, and form the two product generators (P_1, P_2) and (Q_1, Q_2). Their four elliptic components remain in the factor-adjacent order (P_1, P_2, Q_1, Q_2). The initial coordinates are normalized to $Z = 1$ by one Montgomery batch inversion of their four nonzero denominators. The two pair products and inverse recovery use the narrow packed kernels; the single inverse uses the reference scalar boundary of Section 3.2. Both curves are normalized, so their ladder constants are $A_{24,j} = (A_j + 2)/4$.

For one original component T, the ladder retains the pair $[2^e]T$ and $[2^e - 1]T$. A differential addition and doubling produce $[2^{e+1} - 1]T$ and $[2^{e+1}]T$, preserving their difference T. Algorithm 4 executes the four copies of this recurrence with M_4 and S_4. Its public descent sequence $(r_0, \ldots, r_{h-1})$ is obtained from the $b_{\mathrm{sg}} = \mathsf{true}$ minimizing cuts of Algorithm 11. Each saved entry contains a product-point pair and its predecessor pair, written $\mathcal{S}_j.P$ and $\mathcal{S}_j.D$. The remaining subtree size starts at $n_0 = N$ and satisfies $n_{j+1} = n_j - r_j$; the terminal size is $n_h = 1$.

Algorithm 4 PrepareKernelPairs
$(B_1, B_2, E_1, E_2, N; \mathsf{preCut})$

Require: Normalized Montgomery curves, compatible factor bases, and descent lengths $\mathsf{preCut} = (r_0, \ldots, r_{h-1})$
Ensure: Saved ladder entries $\mathcal{S}_0, \ldots, \mathcal{S}_h$ and remaining sizes $n_0, \ldots, n_h$

Normalize (P_1, P_2, Q_1, Q_2) to obtain the four-vector $(X_0 : \mathbf{1}_4)$
$(X, Z) \leftarrow (X_0, \mathbf{1}_4)$; $(X_D, Z_D) \leftarrow (\mathbf{1}_4, 0)$ ▷ predecessors initially equal the identity
$A_{24} \leftarrow (A_{24,1}, A_{24,2}, A_{24,1}, A_{24,2})$; $n_0 \leftarrow N$
for $j \leftarrow 0$ to $h - 1$ **do**
  $\mathcal{S}_j \leftarrow ((X : Z), (X_D : Z_D))$
  for $k \leftarrow 1$ to r_j **do**
    $a \leftarrow X + Z$; $b \leftarrow X - Z$
    $c \leftarrow X_D + Z_D$; $d \leftarrow X_D - Z_D$
    $a_2 \leftarrow \mathsf{S}_4(a)$; $b_2 \leftarrow \mathsf{S}_4(b)$
    $u \leftarrow \mathsf{M}_4(a, d)$; $v \leftarrow \mathsf{M}_4(b, c)$
    $\Delta \leftarrow a_2 - b_2$
    $X' \leftarrow \mathsf{M}_4(a_2, b_2)$
    $Z' \leftarrow \mathsf{M}_4(b_2 + \mathsf{M}_4(A_{24}, \Delta), \Delta)$
    $X'_D \leftarrow \mathsf{S}_4(u + v)$
    $Z'_D \leftarrow \mathsf{M}_4(X_0, \mathsf{S}_4(u - v))$
    $(X, Z, X_D, Z_D) \leftarrow (X', Z', X'_D, Z'_D)$
  end for
  $n_{j+1} \leftarrow n_j - r_j$
end for
$\mathcal{S}_h \leftarrow ((X : Z), (X_D : Z_D))$ ▷ order-8 product generators
return $(\mathcal{S}_0, \ldots, \mathcal{S}_h;\ n_0, \ldots, n_h)$

Matrix construction. The terminal ladder entry supplies two order-8 product points $K_r = (K_{r,1}, K_{r,2})$, $r = 1, 2$; $[4]K_1$ and $[4]K_2$ generate the gluing kernel. On each factor j, projective symmetric-element construction from $[2]K_{1,j}, [2]K_{2,j}$ gives a nonzero common scale δ_j and three trace-zero matrices $G_r^{(j)}$, $r = 1, 2, 3$, whose projective classes represent 2-torsion translations [6]. Together they give $\mathbf{M} \in \mathrm{GL}_4(\mathbb{F}_{p^2})$ with

$$\theta(P_1, P_2) = \mathbf{M}\,(X_1X_2, X_1Z_2, Z_1X_2, Z_1Z_2)^{\mathsf{T}}.$$

The common scales are retained rather than divided out, since a nonzero common factor does not change the represented theta point.

Write a factor's order-4 inputs as $(X_1 : Z_1), (X_2 : Z_2)$ and set $x_r^{\pm} = X_r \pm Z_r$. Table 6 lists the six tests, with c local to that factor. After canonical reduction, Algorithm 5 uses the last satisfied condition in reference order. Multiplication by i exchanges real and imaginary coordinates and changes one sign.

Table 6 Symmetric-element conditions for one elliptic factor.

c	Zero condition	$(X : Z)$	ϵ_c
0	$x_1^- = 0$	$(X_2 : Z_2)$	$+1$
1	$x_1^+ = 0$	$(X_2 : Z_2)$	-1
2	$x_2^- = 0$	$(X_1 : Z_1)$	-1
3	$x_2^+ = 0$	$(X_1 : Z_1)$	$+1$
4	$ix_1^+x_2^- + x_2^+x_1^- = 0$	$(X_1 : Z_1)$	$+1$
5	$ix_1^+x_2^- - x_2^+x_1^- = 0$	$(X_1 : Z_1)$	-1

For the selected $(X : Z)$, set $\delta = X^2 - Z^2$, $\sigma = X^2 + Z^2$, and $\pi = 2XZ$. Write the resulting triples for the two factors as (G_1, G_2, G_3) and (H_1, H_2, H_3), with each matrix of the form $\left(\begin{smallmatrix} g_{00} & g_{01} \\ g_{10} & -g_{00} \end{smallmatrix}\right)$. Algorithm 5 forms $\mathbf{M}$ in four-product M_4 layers; $[r = 3]$ is one for $r = 3$ and zero otherwise.

Algorithm 5 GluingMatrix $(T_{1,1}, T_{2,1}, T_{1,2}, T_{2,2})$

Require: Two order-4 Kummer points $T_{r,j} = (X_{r,j} : Z_{r,j})$ on each factor j
Ensure: Change-of-coordinate matrix $\mathbf{M}$, or $\perp$

for $j \leftarrow 1$ to 2 **do**
  Reduce the four input coordinates canonically; $c \leftarrow -1$
  for $r \leftarrow 1$ to 2 **do**
    $x_r^{\pm} \leftarrow X_{r,j} \pm Z_{r,j}$
  end for
  for $k \leftarrow 0$ to 5 **do**
    if condition k in Table 6 holds **then**
      $c \leftarrow k$
    end if
  end for
  if $c = -1$ **then**
    return $\perp$
  end if

Select $(X : Z)$ and ϵ_c from Table 6
$\delta_j \leftarrow X^2 - Z^2$; $\sigma \leftarrow X^2 + Z^2$; $\pi \leftarrow 2XZ$
$G_1^{(j)} \leftarrow \left(\begin{smallmatrix} -\sigma & \pi \\ -\pi & \sigma \end{smallmatrix}\right)$
$G_2^{(j)} \leftarrow \left(\begin{smallmatrix} 0 & (-1)^{c+1}\delta_j \\ (-1)^{c+1}\delta_j & 0 \end{smallmatrix}\right)$
if $c \in \{0, 1\}$ **then**
  Exchange $G_1^{(j)}$ and $G_2^{(j)}$
end if
$G_3^{(j)} \leftarrow \epsilon_c \left(\begin{smallmatrix} i\pi & -i\sigma \\ i\sigma & -i\pi \end{smallmatrix}\right)$
if $c \in \{4, 5\}$ **then**
  Exchange $G_2^{(j)}$ and $G_3^{(j)}$
end if
end for
$(G_1, G_2, G_3) \leftarrow (G_1^{(1)}, G_2^{(1)}, G_3^{(1)})$
$(H_1, H_2, H_3) \leftarrow (G_1^{(2)}, G_2^{(2)}, G_3^{(2)})$
$m_0 \leftarrow (\delta_1\delta_2, 0, 0, 0)$
for $r \leftarrow 1$ to 3 **do**
  $v \leftarrow \mathsf{M}_4(((G_r)_{00}, (G_r)_{00}, (G_r)_{01}, (G_r)_{01}), ((H_r)_{00}, (H_r)_{01}, (H_r)_{00}, (H_r)_{01}))$
  $m_0 \leftarrow m_0 + (-1)^{[r=3]} v$
end for
$(a, b, c, d) \leftarrow m_0$; $(h, k, l) \leftarrow ((H_2)_{00}, (H_2)_{01}, (H_2)_{10})$
$m_1 \leftarrow \mathsf{M}_4((h, k, h, k), (a, a, c, c)) + \mathsf{M}_4((l, -h, l, -h), (b, b, d, d))$
$(g, k, l) \leftarrow ((G_1)_{00}, (G_1)_{01}, (G_1)_{10})$
for $r \leftarrow 2$ to 3 **do**
  $(a, b, c, d) \leftarrow m_{r-2}$
  $m_r \leftarrow \mathsf{M}_4((g, g, k, k), (a, b, a, b)) + \mathsf{M}_4((l, l, -g, -g), (c, d, c, d))$
end for
$(s_0, s_1, s_2, s_3) \leftarrow (\delta_1\delta_2, \delta_1, \delta_2, 1)$
for $r \leftarrow 0$ to 3 **do**
  $\mathbf{M}_{r,:} \leftarrow s_r m_r$ ▷ one M_4 when $r < 3$
end for
return M

Codomain and auxiliary images. Algorithm 6 returns the gluing matrix, the theta null point O, and two scaling tuples V_1, V_2 for the first pair evaluation. Each doubling level processes all four elliptic components together using the kernels of Section 4.1; each matrix row uses one M_4 layer.

Evaluation of saved pairs. For $P \neq \pm Q$, barycentric coordinates $(u : v : w)$ represent

$$x(P + Q) = (u - v : w), \qquad x(P - Q) = (u + v : w),$$

where $x(P)$ denotes $(X_P : Z_P)$. The triple is computed from $x(P), x(Q), x(P - Q)$, the last being supplied by the ladder predecessor [1, 6]. Two saved product points give four triples in the factor-adjacent order of Table 5. Their evaluation products are

Algorithm 6 PrepareGluingMap
(E_1, E_2, K_1, K_2)

Require: Curves E_1, E_2 and order-8 product points $K_r = (K_{r,1}, K_{r,2})$, $r = 1, 2$, in x-only coordinates
Ensure: Matrix $\mathbf{M}$, theta null point O, auxiliary tuples V_1, V_2, or $\perp$

for $r \leftarrow 1$ to 2 **do**
  for $j \leftarrow 1$ to 2 **do**
    $T_{r,j} \leftarrow \mathsf{xDBL}(K_{r,j}; E_j)$
    $U_{r,j} \leftarrow \mathsf{xDBL}(T_{r,j}; E_j)$
  end for
end for
Check $U_{r,j} \neq 0$, $[2]U_{r,j} = 0$ for all r, j, and $U_{1,j} \neq U_{2,j}$ for $j = 1, 2$; reject on failure
$\mathbf{M} \leftarrow \mathsf{GluingMatrix}(T_{1,1}, T_{2,1}, T_{1,2}, T_{2,2})$
if $\mathbf{M} = \perp$ **then**
  return $\perp$
end if
for $r \leftarrow 1$ to 2 **do**
  $(X_1 : Z_1, X_2 : Z_2) \leftarrow K_r$
  $v \leftarrow \mathsf{M}_4((X_1, X_1, Z_1, Z_1), (X_2, Z_2, X_2, Z_2))$
  for $k \leftarrow 0$ to 3 **do**
    $q \leftarrow \mathsf{M}_4(\mathbf{M}_{k,:}, v)$
    $(W_r)_k \leftarrow q_0 + q_1 + q_2 + q_3$
  end for
end for
$(a, b, c, d \mid e, f, g, h) \leftarrow \mathsf{H}_{4\times 2}(\mathsf{S}_8(W_1 \mid W_2))$
if $d \neq 0$ or $h \neq 0$ or any of a, b, c, e, f, g is zero **then**
  return $\perp$
end if
$(u_0, u_1, u_2, u_3) \leftarrow \mathsf{M}_4((a, b, a, b), (e, e, g, g))$
$O \leftarrow \mathsf{H}_4(u_0, u_1, u_2, 0)$
$(v_0, v_1, v_2, v_3) \leftarrow \mathsf{M}_4((c, a, f, e), (u_1, u_3, u_2, u_3))$
$V_1 \leftarrow (v_0, v_0, v_1, v_1)$; $V_2 \leftarrow (v_2, v_3, v_2, v_3)$
return $(\mathbf{M}, O, V_1, V_2)$

then grouped by factor in $(2 \times 4 \times 1)$ kernels; the matrix action, squaring, Hadamard transforms, and scaling keep both theta images packed.

Algorithm 7 takes product points P_r, Q_r, D_r with $D_r = P_r - Q_r$ in each factor; their coordinate vectors follow Table 5. The first call uses the terminal order-8 pair as P, the original pair as Q, and the tuples V_1, V_2, returning the original pair's images in dual theta coordinates. Their projective inverses supply the scaling tuples for all subsequent saved pairs, whose evaluations end with a Hadamard transform. Algorithm 10 combines these calls.

Algorithm 7 EvaluateGluingPair
$(P_1 \mid P_2, Q_1 \mid Q_2, D_1 \mid D_2; \mathbf{M}, V_1, V_2, b_{\text{dual}})$

Require: Four x-only triples $P_{r,j}, Q_{r,j}, D_{r,j} = P_{r,j} - Q_{r,j}$, matrix $\mathbf{M}$, projective scaling tuples V_1, V_2, and a public output flag
Ensure: Two theta images $Y_1 \mid Y_2$

$a \leftarrow \mathsf{M}_4(X_P + Z_P, X_Q - Z_Q)$
$b \leftarrow \mathsf{M}_4(X_P - Z_P, X_Q + Z_Q)$
$s \leftarrow a + b$; $t \leftarrow b - a$
$c \leftarrow \mathsf{M}_4(Z_D, s)$
$d \leftarrow \mathsf{M}_4(X_D, t)$; $e \leftarrow \mathsf{M}_4(Z_D, t)$
$c \leftarrow \mathsf{S}_4(c)$; $f \leftarrow \mathsf{S}_4(d)$
$u \leftarrow f + c$; $v \leftarrow f - c$; $w \leftarrow 2\mathsf{M}_4(d, e)$ ▷ four barycentric triples in parallel
Group $(u_{r,1}, v_{r,1}, w_{r,1})$ and $(u_{r,2}, v_{r,2}, w_{r,2})$ by factor, with $r = 1 \mid 2$ in each two-value tuple
$(A_{0,1} \mid A_{0,2}) \leftarrow (u_{1,1} \mid u_{2,1}) \star (u_{1,2} \mid u_{2,2})$
  $+(v_{1,1} \mid v_{2,1}) \star (v_{1,2} \mid v_{2,2})$
$(A_{1,1} \mid A_{1,2}) \leftarrow (u_{1,1} \mid u_{2,1}) \star (w_{1,2} \mid w_{2,2})$
$(A_{2,1} \mid A_{2,2}) \leftarrow (w_{1,1} \mid w_{2,1}) \star (u_{1,2} \mid u_{2,2})$
$(A_{3,1} \mid A_{3,2}) \leftarrow (w_{1,1} \mid w_{2,1}) \star (w_{1,2} \mid w_{2,2})$
$(B_{0,1} \mid B_{0,2}) \leftarrow (u_{1,1} + v_{1,1} \mid u_{2,1} + v_{2,1})$
  $\star(u_{1,2} + v_{1,2} \mid u_{2,2} + v_{2,2}) - (A_{0,1} \mid A_{0,2})$
$(B_{1,1} \mid B_{1,2}) \leftarrow (v_{1,1} \mid v_{2,1}) \star (w_{1,2} \mid w_{2,2})$
$(B_{2,1} \mid B_{2,2}) \leftarrow (w_{1,1} \mid w_{2,1}) \star (v_{1,2} \mid v_{2,2})$
$B_{3,1} \leftarrow 0$; $B_{3,2} \leftarrow 0$; $U \leftarrow 0$; $V \leftarrow 0$
for $k \leftarrow 0$ to 3 **do**
  $U \leftarrow U + \mathsf{M}_8(\mathbf{M}^{(2)}_{:k}, A_{k,1}\mathbf{1}_4 \mid A_{k,2}\mathbf{1}_4)$
  if $k < 3$ **then**
    $V \leftarrow V + \mathsf{M}_8(\mathbf{M}^{(2)}_{:k}, B_{k,1}\mathbf{1}_4 \mid B_{k,2}\mathbf{1}_4)$
  end if
end for
$Y \leftarrow \mathsf{H}_{4\times 2}(\mathsf{S}_8(U) - \mathsf{S}_8(V))$
$Y \leftarrow \mathsf{M}_8(Y, V_1 \mid V_2)$
if $b_{\text{dual}} = \mathsf{false}$ **then**
  $Y \leftarrow \mathsf{H}_{4\times 2}(Y)$
end if
return $Y_1 \mid Y_2 = Y$

The duplicated matrix columns are prepared once per gluing map and reused for every saved pair. Barycentric formation, the two matrix actions, squaring, projective scaling, and Hadamard transforms remain in SIMD layouts throughout the transition.

5.2 Generic Theta Isogenies and Chain Integration

Theta precomputation and kernel descent. Kernel doubling uses the standard theta duplication identities [16, 17]. A theta structure stores a tuple L_c proportional to the coordinatewise inverse of its current null point, a tuple D_c retained by isogeny construction, and an optional doubling tuple U_c. The Boolean b_{pre} records whether U_c has been computed. We write this state as $\mathcal{T} = (L_c, D_c, U_c, b_{\text{pre}})$; assignments to these

four fields update $\mathcal{T}$. Immediately after gluing, with null point O, these are initialized as

$$L_c = \mathsf{InvertPoint}(O),$$
$$U_c = \mathsf{InvertPoint}(\mathsf{H}_4(\mathsf{S}_4(O))).$$

After a generic isogeny, write $D_c = (d_0, d_1, d_2, d_3)$ and recover the dual null point by

$$\Omega_c = (d_0 d_3 d_1,\ d_0 d_3 d_2,\ (L_c)_3,\ (L_c)_2). \tag{1}$$

Construction has already supplied L_c proportional to its projective inverse. If another descent is required, compute $U_c = \mathsf{InvertPoint}(\mathsf{H}_4(\mathsf{S}_4(\Omega_c)))$ and set $b_{\mathrm{pre}} = \mathsf{true}$. Thus this precomputation is deferred until the curve is actually used for doubling.

For $P = P_1 \mid P_2$, the operation $\mathsf{ThetaDouble}(P; L_c^{(2)}, U_c^{(2)})$ has the schedule

$$P \xrightarrow{\mathsf{S}_8} \xrightarrow{\mathsf{H}_{4\times 2}} \xrightarrow{\mathsf{S}_8} \xrightarrow{\mathsf{M}_8(\cdot, U_c^{(2)})} \xrightarrow{\mathsf{H}_{4\times 2}} \xrightarrow{\mathsf{M}_8(\cdot, L_c^{(2)})} [2]P.$$

The first descent on a curve prepares $L_c^{(2)}, U_c^{(2)}$ and packs the active pair. Consecutive cuts on that curve continue from the packed result and reuse both constants; each saved stack entry retains its four-coordinate point layout. An isogeny changes the curve and invalidates this doubling workspace. Both reference-style and DP schedules use this same layout reuse.

Generic theta-isogeny phase. Construction and evaluation follow the theta-model $(2,2)$-isogeny formulas [18]. SQIsign 3.0 stores the codomain in dual theta coordinates after a generic step [1]. The public Boolean b_{dual} is false for the first generic step and true thereafter. We write $\mathsf{H}_\rho^b(X) = X$ when $b = \mathsf{false}$ and $\mathsf{H}_\rho(X)$ otherwise, for $\rho = 4$ or 4×2. Algorithm 8 takes two order-8 points T_1, T_2, whose [4] multiples generate the isogeny kernel. It constructs L_c, D_c and evaluates the saved frontier. The flag b_{check} is true except at the last generic step. It enables the reference's kernel tests; the final product checks are made after splitting.

In verification, the frontier contains d saved pairs, hence $2d$ theta points. The evaluation list may additionally contain external theta points. The evaluator takes as many eight-point blocks as possible, followed by two-point blocks. Its general interface also supports a one-point tail. An eight-point block uses four coordinate arrays and executes the same coordinate formula on eight independent points. A two-point block uses $\mathsf{S}_8, \mathsf{M}_8, \mathsf{H}_{4\times 2}$, and a one-point tail uses $\mathsf{S}_4, \mathsf{M}_4, \mathsf{H}_4$. The coefficient layout for each required width is prepared once per frontier. Below, $\mathsf{S}, \mathsf{M}, \mathsf{H}^b$ denote these explicitly selected realizations of coordinatewise arithmetic and per-point Hadamard transforms.

Algorithm 8 GenericThetaIsogeny
$(T_1 \mid T_2, \mathcal{F}, b_{\mathrm{dual}}, b_{\mathrm{check}})$

Require: Order-8 theta points T_1, T_2 as defined above, evaluation list $\mathcal{F}$ of theta points, and public flags
Ensure: Codomain data $\mathcal{T} = (L_c, D_c, U_c, b_{\mathrm{pre}})$ and the images of $\mathcal{F}$
1: $(a, b, c, \delta \mid e, f, g, h) \leftarrow$
$\mathsf{H}_{4\times 2}(\mathsf{S}_8(\mathsf{H}_{4\times 2}^{b_{\mathrm{dual}}}(T_1 \mid T_2)))$

if b_{check} and any of a, b, e, f, g, h is zero **then**
  return $\perp$
end if
$w \leftarrow \mathsf{M}_4((b, a, b, a), (h, g, h, g))$
$L_c \leftarrow \mathsf{M}_4(w, (g, h, e, f))$
$D_c \leftarrow (e, a, b, f)$ ▷ retained components for (1)
if b_{check} **then**
  $(u \mid v) \leftarrow (c \mid \delta) \star ((L_c)_2 \mid (L_c)_3)$
  if $u \neq v$ **then**
    return $\perp$
  end if
end if
$U_c \leftarrow \perp$; $b_{\mathrm{pre}} \leftarrow \mathsf{false}$
Prepare the required 8/2/1 coefficient layouts from L_c
for each 8-, then 2-, then 1-point block $\mathcal{B}$ of $\mathcal{F}$ **do**
  $\mathcal{B} \leftarrow \mathsf{M}(\mathsf{H}(\mathsf{S}(\mathsf{H}^{b_{\mathrm{dual}}}(\mathcal{B}))), L_c)$ ▷ broadcast L_c in the selected layout
end for
return $(\mathcal{T}, \mathcal{F})$

The stored minimizing cuts of Algorithm 11 determine two programs. Cuts from states containing gluing form preCut, used by Algorithm 4; cuts from wholly generic states form thetaCut in preorder. After gluing, $\mathcal{Q} = (Q_0, \ldots, Q_d)$ is the saved-pair stack, with Q_d active, and m_j is the number of remaining generic steps for entry j. Algorithm 9 consumes this stack in the same order as the recursive cost tree. The next unused program entry has index k, and t counts completed generic steps. Its state $\mathcal{T}$ is the codomain representation defined above. The workspace W stores the duplicated constants $W.L = L_c^{(2)}$, $W.U = U_c^{(2)}$, and the packed active pair $W.P$. ThetaDouble uses these prepared constants directly in the displayed doubling formula.

Algorithm 9 ThetaStrategy
$(N, \mathcal{Q}, \boldsymbol{m}, d, \mathcal{T}; \mathsf{thetaCut})$

Require: Saved pairs and remaining sizes from gluing, active index d, initial theta structure, and generic cut program
Ensure: Terminal theta structure $\mathcal{T}$

$k \leftarrow 0$; $t \leftarrow 0$
while $d \geq 0$ **do**
  $b_{\mathrm{packed}} \leftarrow \mathsf{false}$
  while $m_d > 1$ **do**
    $i \leftarrow \mathsf{thetaCut}[k]$; $k \leftarrow k + 1$
    if $b_{\mathrm{packed}} = \mathsf{false}$ **then**
      if $b_{\mathrm{pre}} = \mathsf{false}$ **then**
        Recover Ω_c by (1)
        $U_c \leftarrow \mathsf{InvertPoint}(\mathsf{H}_4(\mathsf{S}_4(\Omega_c)))$
        $b_{\mathrm{pre}} \leftarrow \mathsf{true}$
      end if
      $W \leftarrow (L_c^{(2)}, U_c^{(2)}, Q_d)$ in packed pair form
      $b_{\mathrm{packed}} \leftarrow \mathsf{true}$
    end if

```
        for j ← 1 to i do
            W.P ← ThetaDouble(W.P; W.L, W.U)
        end for
        m_{d+1} ← m_d − i; d ← d + 1; Q_d ← W.P ▷ store two four-coordinate points; retain packed W.P
    end while
    b_dual ← (t > 0); b_check ← (t < N − 2)
    (T, (Q_0, ..., Q_{d−1})) ←
    GenericThetaIsogeny(Q_d, (Q_0, ..., Q_{d−1}), b_dual, b_check)
    for j ← 0 to d − 1 do
        m_j ← m_j − 1
    end for
    d ← d − 1; t ← t + 1
end while
return T
```

Splitting isogeny. The final generic step supplies the dual null point $\Omega_c = (x, y, z, t)$ through (1). For verification, SQIsign 3.0 uses the fixed change to product coordinates

$$(a, b, c, d) = (x + z,\ y + t,\ y - t,\ x - z).$$

It checks that every coordinate is nonzero and that $ad = bc$, the product theta relation. The second factor $(a : b)$ must also satisfy $a^4 \neq b^4$. The first factor $(a : c)$ is passed to NormalizeCurve from Section 4.1 to recover the canonical commitment curve. The coordinate additions and products use the existing packed field layouts.

Algorithm 10 combines the ladder, gluing, generic strategy execution, and splitting; its two cut programs are derived in Section 5.3.

5.3 Dynamic Programming

The ordered-tree principle of Section 4.2 extends to the full chain, with the gluing step included in the cost model following [8]. The SQIsign 3.0 reference chooses $i = \lfloor m/2 \rfloor$ for pre-gluing subtrees with $m \geq 16$ and $i = m - 1$ otherwise; generic subtrees use $i = \lfloor m/2 \rfloor$ [1]. The direct descent for small pre-gluing subtrees avoids further saved pairs requiring gluing evaluation. Our dynamic program selects cuts using the SIMD costs and preparation reuse described above.

Optimization problem. For a fixed parameter set, the chain contains one gluing step and $N - 1$ generic $(2, 2)$-isogenies. The dynamic program chooses an ordered tree for the x-only pre-gluing ladder and the subsequent theta chain. A cut trades additional doubling against propagation of the saved parent pair. Its cost also depends on whether theta precomputation and the packed doubling workspace are already available.

We write one cost table

$$C_N(m, d, t; \boldsymbol{b}), \qquad \boldsymbol{b} = (b_{\mathrm{sg}}, b_{\mathrm{pre}}, b_{\mathrm{packed}}),$$

where the three flags are Boolean. Table 7 defines the state and cost functions. The variable t counts generic isogenies already executed, so it determines the first and last generic-step conventions without introducing an additional leaf classification.

Algorithm 10 ThetaChain
$(B_1, B_2, E_1, E_2, N; \mathsf{preCut}, \mathsf{thetaCut})$

Require: Compatible elliptic bases, normalized factor curves, public length N and the two cut programs
Ensure: Canonical commitment curve, or $\perp$

$(\mathcal{S}_0, \ldots, \mathcal{S}_h; n_0, \ldots, n_h) \leftarrow$ PrepareKernelPairs$(B_1, B_2, E_1, E_2, N; \mathsf{preCut})$
$(K_1, K_2) \leftarrow \mathcal{S}_h.P$
$(\mathbf{M}, O, V_1, V_2) \leftarrow$ PrepareGluingMap(E_1, E_2, K_1, K_2)
$Q_0 \leftarrow$ EvaluateGluingPair $(\mathcal{S}_h.P, \mathcal{S}_0.P, \mathcal{S}_h.D; \mathbf{M}, V_1, V_2, \mathsf{true})$
$(Y_1 \mid Y_2) \leftarrow Q_0$
$(V_1' \mid V_2') \leftarrow (\mathsf{InvertPoint}(Y_1) \mid \mathsf{InvertPoint}(Y_2))$
$Q_0 \leftarrow \mathsf{H}_{4\times 2}(Q_0)$
for $j \leftarrow 1$ to $h-1$ **do**
  $Q_j \leftarrow$ EvaluateGluingPair $(\mathcal{S}_j.P, \mathcal{S}_0.P, \mathcal{S}_j.D; \mathbf{M}, V_1', V_2', \mathsf{false})$
end for
for $j \leftarrow 0$ to $h-1$ **do**
  $m_j \leftarrow n_j - 1$
end for
$L_c \leftarrow \mathsf{InvertPoint}(O)$
$U_c \leftarrow \mathsf{InvertPoint}(\mathsf{H}_4(\mathsf{S}_4(O)))$
$\mathcal{T} \leftarrow (L_c, \perp, U_c, \mathsf{true})$
$\mathcal{T} \leftarrow$ ThetaStrategy$(N, \mathcal{Q}, \boldsymbol{m}, h-1, \mathcal{T}; \mathsf{thetaCut})$
Recover $\Omega_c = (x, y, z, t)$ by (1)
$(a, b, c, d) \leftarrow (x+z, y+t, y-t, x-z)$
if any of a, b, c, d is zero, or $ad \neq bc$, or $a^4 = b^4$ **then**
  return $\perp$
end if
return NormalizeCurve$(a : c)$

The mixed strategy of [8, Procedure 7] combines the original generic formulas, which use inversions in precomputation and codomain recovery, with projective formulas that eliminate these inversions at the cost of additional multiplications in doubling and point evaluation. It records a subchain by COST$(n, \mathit{flag}, \mathit{leftmost}, \mathit{precomp})$. For each cut, it accounts for the required precomputation and doublings, first solves the left subtree, and then evaluates the saved points with the formula choices returned by that subtree. It compares both formula modes for the right subtree and records the cheaper choice in a checkpoint; the *leftmost* parameter incorporates the distinct gluing-isogeny cost. On our AVX-512 path, the small number of inversion inputs cannot amortize an inversion addition chain across eight lanes, while a scalar inversion also incurs packed–scalar conversion. The measured model therefore selects the projective, inversion-free specialization throughout.

Leaf costs and legal states. The gluing leaf performs the complete transition described in Section 5.1; its output already supplies mathematical theta precomputation, but no packed doubling workspace. The generic leaf uses the flags $b_{\mathrm{dual}} = (t > 0)$ and

Table 7 Notation for the theta-model $(2,2)$-isogeny-chain DP.

Symbol	Meaning
N	Complete chain length, $196,302,400$ at Levels I, III, V.
m	Number of gluing or generic isogenies in the current ordered subtree.
d	Number of saved product-point pairs; a generic frontier has $2d$ theta images.
t	Number of generic isogenies completed before the subtree.
b_{sg}	True exactly when the subtree contains the unique gluing step.
b_{pre}	True when the current curve's mathematical doubling precomputation is available.
b_{packed}	True when its doubling constants and active pair are already in the reusable SIMD workspace.
i	Cut with $m-i$ steps in the left child and i in the right child.
S	Capacity of the saved-pair stack including its active entry, $S=16$.
$C_N(m,d,t;\boldsymbol{b})$	Minimum cost of the indicated ordered subtree with the three Boolean flags.
$\mathrm{DBLcost}_{\mathrm{pre}}(i)$	Complete cost of i four-point x-only gluing-ladder iterations and their output stores.
$\mathrm{DBLcost}_{\theta}^{b}(i)$	Cost of i doublings of two theta points, with $b=b_{\mathrm{packed}}$. For $b=\mathsf{false}$, the current pair and doubling constants are first packed; for $b=\mathsf{true}$, their existing SIMD representations are used.
$\mathrm{PREcost}_{\theta}$	Additional cost of mathematical theta precomputation when the required tuple is not yet available.
$\mathrm{GLUEcost}(d)$	Matrix and codomain construction, all d saved-pair evaluations, the interleaving copy, and initial theta precomputation.
$\mathrm{LEAFcost}_{\theta}(t,d)$	Generic construction, the checks required at step t, and complete $8/2/1$ evaluation of the $2d$ saved images with shared coefficient preparation.

$b_{\mathrm{check}} = (t < N-2)$ of Algorithm 8. Thus

$$C_N(1,d,0;\mathsf{true},\mathsf{false},\mathsf{false}) = \mathrm{GLUEcost}(d),$$
$$C_N(1,d,t;\mathsf{false},b_{\mathrm{pre}},b_{\mathrm{packed}}) = \mathrm{LEAFcost}_{\theta}(t,d).$$

The gluing leaf requires $d \geq 1$, because the original kernel pair is evaluated using the terminal order-8 pair. A generic leaf requires no further doubling, regardless of its prepared state.

All states satisfy $m \geq 1$ and $0 \leq d < S$. For $b_{\mathrm{sg}} = \mathsf{true}$, we require $t=0$, $m \leq N$, and $b_{\mathrm{pre}} = b_{\mathrm{packed}} = \mathsf{false}$. For $b_{\mathrm{sg}} = \mathsf{false}$, we require $0 \leq t \leq N-2$, $m \leq N-1-t$,

and $b_{\text{packed}} \Rightarrow b_{\text{pre}}$. A subtree containing the final generic step must have $d = 0$: this is $m = N$ for a gluing-containing state or $t + m = N - 1$ for a wholly generic state. A nonterminal cut additionally requires $d + 1 < S$. Every illegal state has cost $+\infty$.

Ordered recurrence. Let

$$\eta(m, b_{\text{sg}}) = \begin{cases} m - 1, & b_{\text{sg}} = \mathsf{true}, \\ m, & b_{\text{sg}} = \mathsf{false} \end{cases}$$

count the generic isogenies executed by a subtree. For $b_{\text{sg}} = \mathsf{true}$ the descent cost is $\mathrm{DBLcost}_{\boldsymbol{b}}(i) = \mathrm{DBLcost}_{\text{pre}}(i)$. Otherwise it is

$$\begin{aligned} \mathrm{DBLcost}_{\boldsymbol{b}}(i) = {} & \mathrm{DBLcost}_{\theta}^{b_{\text{packed}}}(i) \\ & + [b_{\text{pre}} = \mathsf{false}]\, \mathrm{PREcost}_{\theta}\,. \end{aligned}$$

The bracket is 1 when its condition holds and 0 otherwise.

A left descent retains the same curve. Its left child can therefore use the theta precomputation and packed doubling constants already prepared for that curve, together with the packed pair produced by the descent. After the left subtree is evaluated, its right sibling uses a new curve; both preparations must be recomputed if needed. The exception is a left child consisting only of gluing: the gluing cost already includes the initial mathematical theta precomputation. Accordingly, set

$$\begin{aligned} \boldsymbol{b}_{\text{L}} &= \begin{cases} (\mathsf{true}, \mathsf{false}, \mathsf{false}), & b_{\text{sg}} = \mathsf{true}, \\ (\mathsf{false}, \mathsf{true}, \mathsf{true}), & b_{\text{sg}} = \mathsf{false}, \end{cases} \\ \boldsymbol{b}_{\text{R}}(i) &= (\mathsf{false},\ b_{\text{sg}} \wedge (m - i = 1),\ \mathsf{false}). \end{aligned}$$

With $t_{\text{R}} = t + \eta(m - i, b_{\text{sg}})$, the recurrence is

$$\begin{aligned} C_N(m, d, t; \boldsymbol{b}) = \min_{1 \le i < m} \big[& \mathrm{DBLcost}_{\boldsymbol{b}}(i) \\ & + C_N(m - i, d + 1, t; \boldsymbol{b}_{\text{L}}) \\ & + C_N(i, d, t_{\text{R}}; \boldsymbol{b}_{\text{R}}(i)) \big]. \end{aligned}$$

The left child includes the evaluation of the saved parent pair; its two images supply the right child's kernel. The root is $C_N(N, 0, 0; \mathsf{true}, \mathsf{false}, \mathsf{false})$. Figure 2 illustrates this dependency.

Algorithm 11 $\mathrm{COST}(m, d, t; \boldsymbol{b})$

Require: Public length N, state $(m, d, t; \boldsymbol{b})$, capacity S and the costs in Table 7
Ensure: Minimum cost and its ordered cut
1: **if** the state violates the bounds above **then**
2: **return** $(+\infty, 0)$
3: **end if**
4: **if** the state is memoized **then**
5: **return** its stored cost and cut
6: **end if**

if $m = 1$ **then**
  if $b_{\mathrm{sg}} = \mathsf{true}$ **then**
    Store and return $(\mathrm{GLUEcost}(d), 0)$
  else
    Store and return $(\mathrm{LEAFcost}_{\theta}(t, d), 0)$
  end if
end if
$mincost \leftarrow +\infty$; $mincut \leftarrow 0$
if $d + 1 < S$ **then**
  for $i \leftarrow 1$ to $m - 1$ **do**
    Set $\boldsymbol{b}_{\mathrm{L}}$ and $\boldsymbol{b}_{\mathrm{R}}(i)$ by the displayed rules
    $(Lcost, -) \leftarrow \mathsf{COST}(m - i, d + 1, t; \boldsymbol{b}_{\mathrm{L}})$
    $t_{\mathrm{R}} \leftarrow t + \eta(m - i, b_{\mathrm{sg}})$
    $(Rcost, -) \leftarrow \mathsf{COST}(i, d, t_{\mathrm{R}}; \boldsymbol{b}_{\mathrm{R}}(i))$
    $c_i \leftarrow \mathrm{DBLcost}_{\boldsymbol{b}}(i) + Lcost + Rcost$
    if $c_i < mincost$ **then**
      $(mincost, mincut) \leftarrow (c_i, i)$
    end if
  end for
end if
Store and return $(mincost, mincut)$ for this state

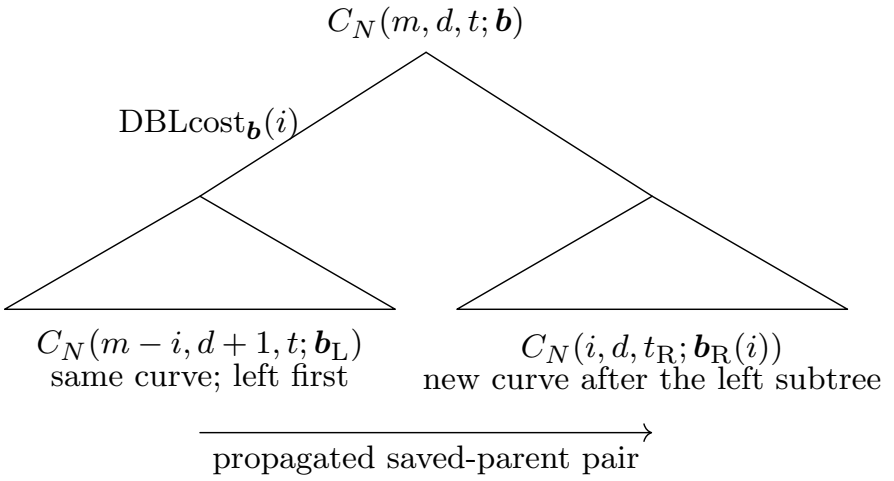

Fig. 2 Ordered theta-chain decomposition with preparation reuse.

This model is SIMD-specific in two ways. First, each cut changes the saved-pair depth and hence the 8/2/1 evaluation decomposition, including coefficient preparation, transpose, tails and output stores. Second, it distinguishes an initial packed descent from continuation on the same curve, and charges mathematical precomputation only when needed. Fixed input normalization and final splitting are outside the recurrence; they do not depend on the selected cuts. Both schedules retain the same field kernels and validity checks.

The implementation compresses the generic interval index into the Booleans $t = 0$ and $t + m = N - 1$: only the first and final generic steps have distinct costs. This stores fewer states without changing the recurrence. Replaying the minimizing cuts in preorder produces preCut and thetaCut as described before Algorithm 9. The public parameter set determines these programs; no signature-dependent optimization is performed during verification.

6 Performance Evaluation and Comparison

Table 8 Hierarchical cycle profile of the SQIsign 3.0 Broadwell verifier. Indented rows are additive within their parent, up to rounding. Costs are mean cycles including instrumentation.

	Level I		Level III		Level V	
Component	Cycles	Share	Cycles	Share	Cycles	Share
Verification total	**13,336,695**	**100.00%**	**35,193,334**	**100.00%**	**91,441,181**	**100.00%**
A. Decoding, structural checks and bookkeeping	**2,451**	**0.02%**	**3,854**	**0.01%**	**5,523**	**0.01%**
B. Challenge recovery	**2,313,217**	**17.34%**	**5,793,337**	**16.46%**	**15,299,824**	**16.73%**
Public-key basis and setup	469,639	3.52%	1,230,340	3.50%	3,095,699	3.39%
Challenge ladder	915,125	6.86%	2,047,925	5.82%	5,381,516	5.89%
Degree-2^ℓ isogeny chain	928,454	6.96%	2,515,072	7.15%	6,822,609	7.46%
C. Response-basis recovery	**1,749,563**	**13.12%**	**4,500,431**	**12.79%**	**10,959,282**	**11.99%**
Two hint bases and setup	651,715	4.89%	1,687,552	4.80%	4,240,118	4.64%
Basis-change matrix application	1,097,848	8.23%	2,812,879	7.99%	6,719,165	7.35%
D. Commitment recovery	**9,244,301**	**69.31%**	**24,831,488**	**70.56%**	**65,027,980**	**71.11%**
Initial point normalization	25,071	0.19%	61,636	0.18%	149,387	0.16%
Pre-gluing x-only descent	1,575,507	11.81%	4,112,960	11.69%	10,689,689	11.69%
Gluing transition	193,884	1.45%	377,877	1.07%	738,243	0.81%
Generic theta-isogeny phase	7,415,827	55.60%	20,200,535	57.40%	53,269,318	58.26%
Doubling and its precomputation	3,631,400	27.23%	10,297,388	29.26%	28,364,802	31.02%
Isogeny construction and checks	747,800	5.61%	1,837,306	5.22%	4,592,354	5.02%
Saved-point evaluation	2,899,381	21.74%	7,842,402	22.28%	19,983,566	21.85%
Generic setup and control	137,246	1.03%	223,439	0.63%	328,595	0.36%
Splitting and product recovery	32,112	0.24%	74,616	0.21%	175,428	0.19%
Chain setup and control	1,900	0.01%	3,865	0.01%	5,914	0.01%
E. Hash-to-challenge	**27,163**	**0.20%**	**64,225**	**0.18%**	**148,571**	**0.16%**

Benchmark methodology. We compare the SQIsign 3.0 Broadwell reference verifier (64-bit assembly field backend) with our AVX-512 verification at Levels I, III, and V. The AVX-512 reference-style and DP configurations use the same arithmetic and isogeny kernels; they differ only in their public chain schedules.

Table 9 Full-chain cycle measurements of public one- and two-dimensional isogeny chains. The degree-2^ℓ row measures the Section 4.1 chain with the verification workload's zero external image points. The theta-model $(2, 2)$ rows measure the complete chain through commitment-curve recovery. The final column gives the percentage reduction of the measured-DP schedule relative to the reference-style AVX-512 schedule.

Level	Chain	Ref. cycles	AVX-512 ref.-style	AVX-512 DP	Ref./DP	Reduction (%)
I	degree-2^ℓ	924,980	543,440	495,214	1.87×	8.9%
	theta-model $(2, 2)$	9,005,716	3,664,918	3,613,774	2.49×	1.4%
III	degree-2^ℓ	2,498,520	1,496,018	1,421,338	1.76×	5.0%
	theta-model $(2, 2)$	24,422,144	10,583,300	10,344,492	2.36×	2.3%
V	degree-2^ℓ	6,798,438	3,474,922	3,323,840	2.05×	4.3%
	theta-model $(2, 2)$	63,574,780	24,641,046	24,125,964	2.64×	2.1%

The experiments use an Intel Core i7-11800H (eight physical cores, 2.30 GHz nominal clock) running native Ubuntu 24.04 and Ubuntu Clang 18.1.3. The AVX-512 build uses `-O3`, `-march=native`, and ThinLTO; the Broadwell build retains the SQIsign 3.0 Release configuration, including `-O2` and ThinLTO. Each timed process is pinned to logical CPU 0.

Correctness testing. The SQIsign 3.0 implementation at each level is checked against known-answer vectors and signatures generated by the reference self-test. The reference random-byte generator is seeded from the operating-system random source; reference key generation and signing produce public keys, messages, and signatures that are serialized and passed to the independently built AVX-512 verifier. The same byte strings are checked with both reference-style and DP schedules. The reference signer and the AVX-512 verifier are not linked together.

Cycle measurements use a serialized processor cycle counter. For the SQIsign 3.0 comparison, reference and AVX-512 verifiers consume the same signature record, with four untimed warm-up calls followed by 31 samples per run. We alternate the configurations over five rounds and report the median of their run medians. Input preparation and correctness checks are outside the timed interval. Direct chain measurements begin from a common prepared state, copied for each call. Field measurements repeat 4096 resident-layout operations per sample and divide by this count.

Verification profile. Source-level probes partition verification into decoding and structural checks, challenge recovery, response-basis recovery, commitment recovery, and hash-to-challenge. The first group also includes the final comparison and outer bookkeeping. Challenge recovery contains the public-key basis, challenge ladder, and degree-2^ℓ isogeny chain. Response-basis recovery contains the two hint bases and the basis-change matrix application. Commitment recovery contains the x-only pre-gluing ladder, Superglue, the generic theta-isogeny phase, and splitting. Hash-to-challenge absorbs normalized curve coefficients as described in Section 2.1. Nested probes record inclusive cycles and subtract directly nested children to obtain exclusive costs. These instrumented values serve for attribution; uninstrumented calls are used for end-to-end comparisons. Table 8 reports the mean over all 155 samples so the indented components

are additive within each parent. Commitment recovery accounts for approximately 69–71% of verification, dominated by theta doubling and saved-point evaluation.

Isogeny-chain measurements. For SQIsign 3.0, the degree-2^ℓ boundary of Section 4.1 starts from a prepared curve and kernel and includes the final curve normalization. No external points are supplied by this verification path. The theta-model $(2,2)$-isogeny boundary starts from the two elliptic kernel bases and ends at the normalized commitment curve. Its components are pre-gluing descent, Superglue, the generic theta-isogeny phase, and splitting/product recovery. The reference and AVX-512 paths use the same SQIsign 3.0 mathematical construction, including Superglue.

Table 9 compares these complete-chain boundaries using uninstrumented calls.

Verification performance. The complete-verifier comparison distinguishes the Broadwell reference, AVX-512 with reference-style schedules, and AVX-512 with both dynamic programs enabled. Table 10 gives full-verification speedups of 2.14×, 2.08×, and 2.34× at Levels I, III, and V. The DP schedules reduce the one-dimensional chain cost by 8.87%, 4.99%, and 4.35%, and the complete theta-chain cost by 1.40%, 2.26%, and 2.09%, respectively. The resulting full-verification reductions are 1.69%, 1.84%, and 1.40%.

Table 10 Complete SQIsign 3.0 verification. All implementations use the same inputs and median-cycle statistic.

Level	Impl.	Schedule	Cycles	Speed-up
I	ref. (64-bit asm.)	ref.	13,079,326	1.00×
	AVX-512	ref.-style	6,209,150	2.11×
	AVX-512	DP	6,104,524	2.14×
III	ref. (64-bit asm.)	ref.	34,774,169	1.00×
	AVX-512	ref.-style	16,997,474	2.05×
	AVX-512	DP	16,685,332	2.08×
V	ref. (64-bit asm.)	ref.	89,969,464	1.00×
	AVX-512	ref.-style	38,942,338	2.31×
	AVX-512	DP	38,396,614	2.34×

7 Conclusion

We presented an AVX-512 implementation of the complete public SQIsign 3.0 verification path at Levels I, III, and V. The implementation covers decoding, structural validation, challenge and commitment recovery, hashing, and the final comparison. Key generation and signing use a separate bounded multi-precision quaternion-arithmetic implementation that remains variable-time; this work focuses on the curve-side verification path.

The field layer uses seven, ten, or fourteen radix-2^{51} limbs according to the prime, with reduction bounds that determine the required final corrections. The vectorized Superglue implementation follows the SQIsign 3.0 x-only construction and preserves packed data through saved-pair evaluation. The two dynamic programs select public

schedules for the degree-2^ℓ and theta-model $(2,2)$-isogeny chains. Their costs account for saved-point evaluation, terminal normalization, pre-gluing descent, and reusable theta precomputation and layouts.

On the Core i7-11800H, the complete Level-I, Level-III, and Level-V verifiers take 6,104,524, 16,685,332, and 38,396,614 cycles, respectively. These correspond to 2.14×, 2.08×, and 2.34× speedups over the SQIsign 3.0 Broadwell verifiers. Relative to the same AVX-512 kernels with reference-style schedules, the two DP strategies remove 104,626, 312,142, and 545,724 cycles from full verification (1.69%, 1.84%, and 1.40%). The results show the benefit of matching public chain schedules to the available SIMD operations.

Beyond SQIsign, the parallelization techniques developed for one- and two-dimensional isogeny computations may be adapted to the corresponding arithmetic components of other isogeny-based schemes, such as POKÉ [19] and PRISM [20]. Their application would require adapting the data layouts and strategy cost models to the parameters and workloads of each scheme.

A Accelerating Key Generation and Signing

The main text examines public verification. This appendix describes our parallelization of curve-side computations used in SQIsign 3.0 key generation and signing and reports measurements of the complete operations.

A.1 Workflows and Implementation Boundary

Key generation samples a secret ideal, replaces it by a suitable equivalent ideal, and evaluates the associated ideal-to-isogeny computation. It then obtains a canonical torsion basis and hint on the public curve and computes the basis-change matrix using Tate pairings and binary discrete logarithms. Signing constructs a commitment curve, hashes the public data and message to the challenge, and computes the quaternion–ideal response and auxiliary curve. A response theta-model $(2,2)$-isogeny chain starts from the commitment–auxiliary product. The images of three commitment-basis points and both output factors determine the response bases and final basis-change matrices [1].

We retain the official multi-precision integer representation, quaternion and ideal algorithms, scalar recoding, and input-dependent search and retry logic. Their curve-side subroutines use the AVX-512 finite-field, elliptic-curve, and theta-coordinate backend. Ideal-to-isogeny conversion therefore combines retained algebraic routines with vectorized curve-side arithmetic. Key generation and signing remain variable-time.

Here, curve-side work denotes elliptic-curve, theta-coordinate, finite-field, and pairing/discrete-logarithm calls, including their local control flow and scalar operations. It excludes outer multi-precision and quaternion–ideal work, hashing, and random-number management.

A.2 Parallel Curve-Side Computation and Chain Scheduling

Key generation and signing use theta-model $(2,2)$-isogeny chains in two ways. First, ideal-to-isogeny conversion produces the public curve in key generation and the commitment and auxiliary curves in signing. Its quaternion–ideal computations supply the data for a kernel on $E_0 \times E_0$; a theta chain performs the curve-side isogeny evaluation, and splitting recovers the required elliptic curve and transported torsion basis. Second, signing's response construction evaluates a theta chain from $E_{\mathrm{com}} \times E_{\mathrm{aux}}$, carrying three commitment-basis points and recovering both output factors [1]. We first describe the parallel basis actions and basis changes used to prepare kernels and recover basis-change matrices, then the packed computations of these two types of theta chain and their DP schedules.

Basis actions and basis changes. For matrix and endomorphism actions on an even-torsion basis, Algorithm 12 packs the two independent bi-scalar multiplications. When the difference point is needed, a third bi-scalar multiplication follows separately. Matrix construction and reduction of its integer coefficients remain in the SQIsign 3.0 representation. $\mathsf{PackedBiscalar}_2$ computes $aP + bQ$ and $cP + dQ$ in parallel from the basis $B = (P, Q, P - Q)$ and the two columns of M.

For basis changes, five independent translation chains in the Tate-pairing calculation are grouped, their final ratios are handled in batches, and four binary discrete logarithms are evaluated concurrently. The first basis-change matrix transforms a basis used by the second, so the two matrix computations remain sequential. This grouping of independent jobs draws on earlier vectorization work [4, 19]; the pairing and discrete-logarithm formulas remain those of SQIsign 3.0.

Algorithm 12 BatchBasisAction$(B, M, f, b_{\mathrm{diff}})$

Require: $B = (P, Q, P - Q)$ on E, matrix $M = \left(\begin{smallmatrix} a & c \\ b & d \end{smallmatrix}\right)$, torsion exponent f, and public difference-point flag b_{diff}
Ensure: (R, S) and, if requested, $R - S$

Reduce a, b, c, d modulo 2^f and convert them to fixed-length scalar words
$(R, S) \leftarrow \mathsf{PackedBiscalar}_2(B, ((a, b), (c, d)), E, f)$
if b_{diff} **then**
    $D \leftarrow [a - c]P + [b - d]Q$ by one further bi-scalar multiplication
    return (R, S, D)
end if
return (R, S)

Theta chains within ideal-to-isogeny conversion. Quaternion endomorphism actions prepare both the product-curve kernel and the external basis to be transported. We parallelize these actions and evaluate the resulting theta chain using shared packed layouts.

Let E_0 be the fixed starting curve and $B_0 = (P_0, Q_0, P_0 - Q_0)$ its standard basis of $E_0[2^f]$, where f is the even-torsion exponent. For an input ideal I, the retained official routine $\mathsf{Qlapoty}(I)$ returns quaternion elements $\beta \in I$ and ϑ, together with the odd integer $d_1 = \mathrm{nrd}(\beta)/\mathrm{N}(I)$; here nrd is the reduced quaternion norm and $\mathrm{N}(I)$ is the ideal norm [1]. Set $u = d_1^{-1} \bmod 2^f$. The action of $u\vartheta$ on B_0 prepares the basis B_K used to construct the kernel, while the action of $u\beta$ prepares the external basis $B_\star$ to be

transported. These actions use the packed basis computations above. In Algorithm 14, EndomorphismBasis applies the specified quaternion element to B_0, and ProductKernel pairs the corresponding P and Q entries of B_K and B_0 to form (K_1, K_2).

Algorithm 13 gives the shared packed chain. It evaluates the three external points together with the live saved pairs. The saved-pair stack $\mathcal{Q}$, remaining lengths $\boldsymbol{m} = (m_0, \ldots, m_d)$, active index d, and theta state $\mathcal{T}$ follow Section 5.2; $\mathcal{F}$ stores the external-point images, and O denotes the elliptic identity. Initialization follows the pre-gluing and Superglue steps of Algorithm 10, with the three external points also evaluated through the gluing map. Cut execution and kernel-pair doubling follow Algorithm 9; $\mathsf{ThetaDouble}^i$ denotes i successive doublings. At each generic step, the evaluation list $\mathcal{A}$ consists of the three external-point images followed by the two theta points of each saved pair $Q_0, \ldots, Q_{d-1}$. Algorithm 8 constructs the isogeny and evaluates this combined list; the returned images update $\mathcal{F}$ and the saved pairs. Splitting uses the product-coordinate transformation of Section 5.2 for both the dual null point and the external-point images. The recovered elliptic points are mapped by the corresponding curve-normalization isomorphisms. The ideal chain retains the second output factor and its transported basis, whereas the response chain retains both factors and the evaluated point pairs.

Algorithm 13 IdealThetaChain$(E_0, K_1, K_2; P_\star, Q_\star, P_\star - Q_\star)$

Require: Prepared kernel pair on $E_0 \times E_0$, three points on its first factor, and public ideal-chain cut programs
Ensure: Second output curve E' and transported basis B'

$\mathcal{F} \leftarrow ((P_\star, O), (Q_\star, O), (P_\star - Q_\star, O))$
Apply the pre-gluing and Superglue steps of Algorithm 10 using $\mathsf{preCut}_{\mathrm{id}}$
Evaluate $\mathcal{F}$ through the gluing map
Initialize $(\mathcal{T}, \mathcal{Q}, \boldsymbol{m}, d)$ as in Algorithm 10
Store $\mathcal{F}$ before $\mathcal{Q}$ in one packed array
$k \leftarrow 0$; $b_{\mathrm{dual}} \leftarrow \mathsf{false}$
while $d \geq 0$ **do**
    while $m_d > 1$ **do**
        $i \leftarrow \mathsf{thetaCut}_{\mathrm{id}}[k]$; $k \leftarrow k + 1$
        $Q_{d+1} \leftarrow \mathsf{ThetaDouble}^i(Q_d, \mathcal{T})$
        $m_{d+1} \leftarrow m_d - i$; $d \leftarrow d + 1$
    end while
    $\mathcal{A} \leftarrow$ the combined point list of $\mathcal{F}$ and $Q_0, \ldots, Q_{d-1}$
    $(\mathcal{T}, \mathcal{A}) \leftarrow$ GenericThetaIsogeny$(Q_d, \mathcal{A}, b_{\mathrm{dual}}, \mathsf{false})$
    Write the images in $\mathcal{A}$ back to $\mathcal{F}$ and $Q_0, \ldots, Q_{d-1}$
    $b_{\mathrm{dual}} \leftarrow \mathsf{true}$
    $m_j \leftarrow m_j - 1$ for $0 \leq j < d$; $d \leftarrow d - 1$
end while
return (E', B') from splitting in Section 5.2, retaining the second factor

Algorithm 14 combines the endomorphism-based kernel and basis preparation with Algorithm 13 for signing's commitment and auxiliary ideal conversions.

Signing response chain. Algorithm 15 summarizes the packed dataflow of the signing response chain. It reuses the gluing, generic theta-isogeny, and splitting computations of

Algorithm 14 SignIdealChain(I, ρ)

Require: Prepared ideal I, role $\rho \in \{\mathrm{com}, \mathrm{aux}\}$, standard basis B_0 on E_0, and even-torsion exponent f
Ensure: Output curve E_ρ and transported basis B_ρ

$(\beta, d_1, \vartheta) \leftarrow \mathsf{Qlapoty}(I)$
$u \leftarrow d_1^{-1} \bmod 2^f$
$B_K \leftarrow \mathsf{EndomorphismBasis}(u\vartheta, B_0)$
$(K_1, K_2) \leftarrow \mathsf{ProductKernel}(B_K, B_0)$
$B_\star \leftarrow \mathsf{EndomorphismBasis}(u\beta, B_0)$
$(E_\rho, B_\rho) \leftarrow$ IdealThetaChain$(E_0, K_1, K_2; B_\star.P, B_\star.Q, B_\star.P - B_\star.Q)$
return (E_ρ, B_ρ)

the two-dimensional verification chain. The external-point images and live saved pairs share a packed array during the generic phase. Each step directly invokes Algorithm 8 on the combined evaluation list described above.

Algorithm 15 SignThetaChain$(E_{\mathrm{com}}, E_{\mathrm{aux}}, K_1, K_2; P, Q, P - Q)$

Require: Product curves, kernel pair, three external points, and public response-chain cut programs
Ensure: Both output curves and the three evaluated point pairs

$\mathcal{F} \leftarrow ((P, O), (Q, O), (P - Q, O))$
Apply the pre-gluing and Superglue steps of Algorithm 10 using $\mathsf{preCut}_{\mathrm{resp}}$
Evaluate $\mathcal{F}$ through the gluing map
Initialize $(\mathcal{T}, \mathcal{Q}, \boldsymbol{m}, d)$ as in Algorithm 10
Store $\mathcal{F}$ before the saved pairs $\mathcal{Q}$ in packed theta layout
$k \leftarrow 0$; $b_{\mathrm{dual}} \leftarrow \mathsf{false}$
while a saved pair remains **do**
    while $m_d > 1$ **do**
        $i \leftarrow \mathsf{thetaCut}_{\mathrm{resp}}[k]$; $k \leftarrow k + 1$
        $(Q_{d+1}, m_{d+1}, d) \leftarrow (\mathsf{ThetaDouble}^i(Q_d, \mathcal{T}), m_d - i, d + 1)$
    end while
    $\mathcal{A} \leftarrow$ the combined point list of $\mathcal{F}$ and $Q_0, \ldots, Q_{d-1}$
    $(\mathcal{T}, \mathcal{A}) \leftarrow$ GenericThetaIsogeny$(Q_d, \mathcal{A}, b_{\mathrm{dual}}, \mathsf{false})$
    Write the images in $\mathcal{A}$ back to $\mathcal{F}$ and $Q_0, \ldots, Q_{d-1}$
    $b_{\mathrm{dual}} \leftarrow \mathsf{true}$
    $m_j \leftarrow m_j - 1$ for $0 \le j < d$; $d \leftarrow d - 1$
end while
return both output curves and point pairs recovered by splitting as in Section 5.2

DP schedules. The ideal-to-isogeny calls in key generation and signing share an offline chain strategy. A separate strategy is used for the response chain, which has a different length and output requirement.

The DP reuses the ordered-tree pre-gluing and generic subproblems, saved-frontier depth, and preparation-state distinctions of Section 5.3 and Algorithm 11, with costs adapted to the ideal and response chains for pre-gluing descent, gluing leaves, generic descent, and generic leaves. These costs include three external-point images and the live saved pairs. The ideal mode retains only E_2 at splitting, whereas the response

mode retains both product factors. Final splitting and output recovery contribute a fixed cost for each mode. Subject to a saved-pair capacity, the offline solver emits a pre-gluing and generic cut program for each mode. The selected cuts determine descent and saved-point evaluation order while preserving the chain's mathematical map.

A.3 Performance Evaluation and Comparison

Measurements use the processor, compiler settings, and CPU affinity described in Section 6.

Key-generation and signing profile. Source-level probes measure the curve-side contribution to official Broadwell key generation and signing, using the boundary defined in Section A.1. Table 11 reports mean cycles, including instrumentation. Each signing sample uses a freshly generated key and a fixed 32-byte zero message. The curve-side category includes scalar and control work alongside SIMD-eligible arithmetic.

Table 11 Curve-side share of SQIsign 3.0 reference key generation and signing (64-bit asm.). Costs are mean cycles including instrumentation.

Level	Operation	Total cycles	Curve-side cycles	Share (%)
I	Keygen	37,503,373	27,170,773	72.45%
I	Sign	106,936,318	73,363,856	68.61%
III	Keygen	136,402,273	72,233,176	52.96%
III	Sign	377,451,040	194,354,216	51.49%
V	Keygen	267,581,086	184,266,169	68.86%
V	Sign	812,248,277	496,607,020	61.14%

Isogeny-chain measurements. The AVX-512 reference-style and DP schedules use the same inputs and are measured with identical source-level probes. Each of five seeded runs comprises 31 iterations; configuration order alternates between runs. Table 12 reports the median of the run medians. The timed boundary starts from the prepared kernel and external basis and includes pre-gluing, the generic theta-isogeny phase, and splitting. Initial integer computations and endomorphism actions used to prepare these inputs are outside the timed interval. The sign ideal-to-isogeny row reports the cost per call; signing makes two such calls.

Key-generation and signing performance. The Broadwell reference and both AVX-512 configurations use uninstrumented calls with the same five initialization seeds and 31 samples per run. We alternate the AVX-512 configuration order and report the mean of the five run medians in Table 13. The two AVX-512 configurations differ only in their public ideal-to-isogeny and response-chain schedules. The timed boundary includes the public byte-oriented API, and all configurations generate, sign, and verify their messages.

The DP chain-call savings are positive in Table 12. These full-operation differences cannot be obtained by adding the chain-call savings: the boundaries differ, and variable-time work contributes to timing variation.

Table 12 Full-chain cycle measurements of theta-model $(2, 2)$-isogeny chains in key generation and signing. The ideal-to-isogeny rows report the cost per call; signing makes two such calls. The response rows measure the complete response chain. The final column gives the percentage reduction of the DP schedule relative to the reference-style AVX-512 schedule.

Level	Chain	Ref. cycles	AVX-512 ref.-style	AVX-512 DP	Ref./DP	Reduction (%)
I	Keygen ideal-to-isogeny	17,394,326	6,507,300	6,402,438	2.72×	1.6%
	Sign ideal-to-isogeny	17,423,423	6,525,848	6,401,873	2.72×	1.9%
	Sign response	10,013,008	3,882,196	3,835,810	2.61×	1.2%
III	Keygen ideal-to-isogeny	47,147,482	18,937,958	18,385,808	2.56×	2.9%
	Sign ideal-to-isogeny	47,100,248	18,996,434	18,444,750	2.55×	2.9%
	Sign response	26,849,810	11,182,800	10,966,672	2.45×	1.9%
V	Keygen ideal-to-isogeny	121,813,166	43,291,758	42,655,586	2.86×	1.5%
	Sign ideal-to-isogeny	121,838,285	43,447,955	42,754,094	2.85×	1.6%
	Sign response	69,654,876	25,403,432	25,063,692	2.78×	1.3%

Table 13 Complete SQIsign 3.0 key generation and signing. Cycles are means of five run medians, rounded to the nearest thousand; speed-ups are relative to the official Broadwell implementation.

Level	Operation	Impl.	Schedule	Cycles	Speed-up
I	Keygen	ref. (64-bit asm.)	ref.	35,450,000	1.00×
		AVX-512	ref.-style	19,491,000	1.82×
		AVX-512	DP	19,346,000	1.83×
	Sign	ref. (64-bit asm.)	ref.	104,879,000	1.00×
		AVX-512	ref.-style	62,964,000	1.67×
		AVX-512	DP	62,859,000	1.67×
III	Keygen	ref. (64-bit asm.)	ref.	117,864,000	1.00×
		AVX-512	ref.-style	78,478,000	1.50×
		AVX-512	DP	77,853,000	1.51×
	Sign	ref. (64-bit asm.)	ref.	358,111,000	1.00×
		AVX-512	ref.-style	270,306,000	1.32×
		AVX-512	DP	267,775,000	1.34×
V	Keygen	ref. (64-bit asm.)	ref.	252,696,000	1.00×
		AVX-512	ref.-style	148,381,000	1.70×
		AVX-512	DP	147,700,000	1.71×
	Sign	ref. (64-bit asm.)	ref.	801,572,000	1.00×
		AVX-512	ref.-style	557,237,000	1.44×
		AVX-512	DP	550,209,000	1.46×

In the official SQIsign 3.0 Broadwell profile, the ideal-to-isogeny theta chain accounts for approximately 46.3%, 34.5%, and 45.3% of complete key generation at Levels I, III, and V, respectively. The two ideal-to-isogeny theta-chain calls and the response theta chain together account for approximately 42.0%, 32.0%, and 38.4% of complete signing, respectively.

Level III has a smaller curve-side share of both official Broadwell key generation and signing than Levels I and V (Table 11), which limits the fraction accelerated by our vectorization and helps explain its lower complete-operation speed-ups in Table 13.

References

[1] Aardal, M.A., Adj, G., Aranha, D.F., Basso, A., Borin, G., Canales Martínez, I.A., Chávez-Saab, J., Corte-Real Santos, M., Dartois, P., De Feo, L., Duparc, M., Espitau, T., Eriksen, J.K., Fouotsa, T.B., Gazzoni Filho, D.L., Hess, B., Invernizzi, R., Kohel, D., Leroux, A., Longa, P., Maino, L., Meyer, M., Mula, M., Nakagawa, K., Onuki, H., Panny, L., Patranabis, S., Petit, C., Pope, G., Reijnders, K., Robert, D., Rodríguez-Henríquez, F., Schaeffler, S., Vercauteren, F., Wallet, A., Wesolowski, B., Woerden, W.: SQIsign: Algorithm specifications and supporting documentation, version 3.0. Technical report, SQIsign team (2026). https://sqisign.org

[2] Wesolowski, B.: The Supersingular Isogeny Problem in Time and Memory $p^{1/3+o(1)}$. Cryptology ePrint Archive, Paper 2026/1486 (2026). https://eprint.iacr.org/2026/1486

[3] Cheng, H., Fotiadis, G., Großschädl, J., Ryan, P.Y.A.: Highly vectorized SIKE for AVX-512. IACR Transactions on Cryptographic Hardware and Embedded Systems **2022**(2), 41–68 (2022) https://doi.org/10.46586/tches.v2022.i2.41-68

[4] De Feo, L., Jian, L.-J., Wang, T.-Y., Yang, B.-Y.: SQISign on ARM. IACR Transactions on Cryptographic Hardware and Embedded Systems **2026**(4) (2026). Accepted for publication

[5] Wang, W., Wang, C., Wu, Y., Xue, Q., Zheng, J., Zhao, Y.: Exposing SIMD Parallelism in SQIsign: An AVX-512 Implementation. Cryptology ePrint Archive, Paper 2026/1713 (2026). https://eprint.iacr.org/2026/1713

[6] Duparc, M.: Superglue: Fast formulae for (2,2)-gluing isogenies. In: ASIACRYPT 2025, Part IV. Lecture Notes in Computer Science, vol. 16248, pp. 372–400. Springer, Singapore (2026). https://doi.org/10.1007/978-981-95-5113-2_12

[7] De Feo, L., Jao, D., Plût, J.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. Journal of Mathematical Cryptology **8**(3), 209–247 (2014) https://doi.org/10.1515/jmc-2012-0015

[8] Lin, J., Wang, S., Zhao, C.-A.: A note on (2,2)-isogenies via theta coordinates. Finite Fields and Their Applications **99**, 102496 (2024) https://doi.org/10.1016/j.ffa.2024.102496

[9] Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: CRYPTO'86. Lecture Notes in Computer Science, vol. 263, pp. 186–194. Springer, Berlin, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_12

[10] Nakagawa, K., Onuki, H., Castryck, W., Chen, M., Invernizzi, R., Lorenzon, G., Vercauteren, F.: SQIsign2D-East: A new signature scheme using 2-dimensional isogenies. In: ASIACRYPT 2024, Part III. Lecture Notes in Computer Science, vol. 15486, pp. 272–303. Springer, Singapore (2025). https://doi.org/10.1007/978-981-96-0891-1_9

[11] Kani, E.: The number of curves of genus two with elliptic differentials. Journal für die reine und angewandte Mathematik **485**, 93–122 (1997) https://doi.org/10.1515/crll.1997.485.93

[12] Basso, A., Dartois, P., De Feo, L., Leroux, A., Maino, L., Pope, G., Robert, D., Wesolowski, B.: SQIsign2D-West: The fast, the small, and the safer. In: ASIACRYPT 2024, Part III. Lecture Notes in Computer Science, vol. 15486, pp. 339–370. Springer, Singapore (2025). https://doi.org/10.1007/978-981-96-0891-1_11

[13] Abel, A., Reineke, J.: uops.info: Latency, Throughput, and Port Usage Information for Instructions on Recent x86 Microarchitectures. https://uops.info/table.html. Tiger Lake measurements; accessed 2026-07-16 (2026)

[14] Montgomery, P.L.: Speeding the Pollard and elliptic curve methods of factorization. Mathematics of Computation **48**(177), 243–264 (1987) https://doi.org/10.1090/S0025-5718-1987-0866113-7

[15] Costello, C., Smith, B.: Montgomery curves and their arithmetic: The case of large characteristic fields. Journal of Cryptographic Engineering **8**(3), 227–240 (2018) https://doi.org/10.1007/s13389-017-0157-6

[16] Mumford, D.B.: On the equations defining abelian varieties. I. Inventiones Mathematicae **1**(4), 287–354 (1966) https://doi.org/10.1007/BF01389737

[17] Robert, D.: Fonctions thêta et applications à la cryptographie. PhD thesis, Henri Poincaré University, Nancy, France (2010). https://tel.archives-ouvertes.fr/tel-00528942

[18] Dartois, P., Maino, L., Pope, G., Robert, D.: An algorithmic approach to (2,2)-isogenies in the theta model and applications to isogeny-based cryptography. In: ASIACRYPT 2024, Part III. Lecture Notes in Computer Science, vol. 15486, pp. 304–338. Springer, Singapore (2025). https://doi.org/10.1007/978-981-96-0891-1_10

[19] Basso, A., Maino, L.: POKÉ: A compact and efficient PKE from higher-dimensional isogenies. In: EUROCRYPT 2025, Part II. Lecture Notes in Computer Science, vol. 15602, pp. 94–123. Springer, Cham (2025). https://doi.org/10.1007/978-3-031-91124-8_4

[20] Basso, A., Borin, G., Castryck, W., Corte-Real Santos, M., Invernizzi, R., Leroux, A., Maino, L., Vercauteren, F., Wesolowski, B.: PRISM: Simple and compact identification and signatures from large prime degree isogenies. In: PKC 2025, Part III. Lecture Notes in Computer Science, vol. 15676, pp. 300–332. Springer, Cham (2025). https://doi.org/10.1007/978-3-031-91826-1_10