Complexity Analysis and Security Implications of the New Isogeny-Path Algorithm

Maher Mamah

200 University Ave, University of Waterloo, Waterloo, ON, Canada
mmamah@uwaterloo.ca

Abstract. Very recently, Wesolowski (*ePrint* 2026/1486) proposed a novel heuristic algorithm for solving the supersingular isogeny-path problem in time and memory $p^{1/3+o(1)}$, where p is the characteristic of the underlying field. Although this improves asymptotically on the previous $p^{1/2}\log^{O(1)}(p)$ complexity, its concrete impact on isogeny-based schemes, particularly SQIsign, remains unclear because of the superpolynomial overhead hidden in $p^{o(1)}$ and the exponential memory requirement.
In this work, we assess the cost of Wesolowski's attack, study its time–memory tradeoffs, and investigate optimizations based on the van Oorschot – Wiener (vOW) technique. Our analysis shows that, over the practical memory ranges considered, Wesolowski's algorithm is expected to improve on the state of the art only at NIST Level I. At higher security levels, its prohibitive memory requirements offset the asymptotic advantage over accelerated Delfs–Galbraith methods. With high parallelization, however, the vOW variant can recover this advantage.
We further study quantum claw-finding improvements. While Grover search can essentially eliminate the large memory requirement, it offers little improvement in running time, whereas Tani's algorithm provides a stronger gate–memory tradeoff at the cost of substantial coherent quantum memory. Finally, we provide an optimized SageMath implementation and compare the theoretical time and space complexities against their empirical counterparts.

1 Introduction

Isogeny-based cryptography is a prominent approach to post-quantum cryptography whose security is based on hard computational problems involving isogenies between supersingular elliptic curves. One of its fundamental problems is the supersingular isogeny-path problem: given supersingular elliptic curves E and E', find an isogeny $\varphi : E \to E'$. A closely related problem that appears prominently in the literature, particularly in connection with the CGL hash function [CGL09] and SQIsign [FKL+20], is the EndRing problem: given a supersingular elliptic curve E, compute its endomorphism ring $\mathrm{End}(E)$.

The supersingular isogeny-path problem has been shown to be polynomial-time equivalent to the endomorphism-ring problem [Wes22,EHL+18]. Since EndRing has been studied since Kohel's foundational work [Koh96], recent cryptanalytic advances have increasingly focused on improving algorithms for EndRing. A particularly relevant low-memory algorithm is that of Eisenträger et

al. [EHL+20], which computes $\mathrm{End}(E)$ in $O(p^{1/2}\log^2 p)$ time while using only negligible polynomial space at the parameter sizes of interest. Their approach first searches for an isogeny $\varphi : E \to E^{(p)}$ from E to its Frobenius conjugate. Composing this isogeny with the Frobenius map yields a non-scalar endomorphism of E. The procedure is then repeated to obtain sufficiently many independent endomorphisms from which the maximal order $\mathrm{End}(E)$ can be recovered. Their analysis relies on GRH together with additional heuristics.
Page and Wesolowski [PW24] gave an unconditional $\widetilde{O}(p^{1/2})$-time algorithm with exponential memory and established a polynomial-time equivalence between the OneEnd and EndRing problems. This reduction, however, is not tight: as noted in [ABF+25, Section 6], using a OneEnd solver for EndRing incurs a multiplicative running-time overhead of roughly $2^{94}\log^{13}(p)$.

Delfs and Galbraith [DG16] obtain a comparable $p^{1/2}\log^{O(1)}(p)$ running time through a different approach. In the group-action inversion stage over curves defined over $\mathbb{F}_p$, a naive bidirectional search would require $\widetilde{O}(p^{1/4})$ stored vertices. This memory requirement can be reduced dramatically using Pollard-ρ-type collision searches with distinguished points. Galbraith and Stolbunov [GS11], building on the GHS approach, develop such a low-storage method. An accelerated version of this algorithm with concrete analysis of its running time was proposed in [SCS22]. Very recently, [PRSU26] gave a GPU-optimized version with explicit estimates for the number of operations needed to break SQIsign parameters.

In a recent breakthrough, Wesolowski [Wes26] proposed a heuristic $p^{1/3+o(1)}$-time and -space algorithm for EndRing. Even though the paper aims at the Isogeny-path problem, it does so through solving EndRing first. As in [EHL+20], the algorithm proceeds by first constructing an isogeny from E to its Frobenius conjugate. Its main new ingredient is the fact that every supersingular elliptic curve admits an isogeny $E \longrightarrow E^{(p)}$ of degree at most $(p/2)^{1/3}$ [AOV26]. This reduces the search for such an isogeny to a claw-finding problem over suitable sets of smooth isogenies, which can then be attacked using a meet-in-the-middle strategy.

Despite the improved asymptotic exponent, the concrete effect of this algorithm on the security of the EndRing and supersingular isogeny-path problems is far from immediate. The $p^{o(1)}$ term hides substantial superpolynomial factors at cryptographic parameter sizes, while the meet-in-the-middle procedure requires exponential memory. Determining whether the new algorithm actually improves concrete attacks therefore requires a careful analysis of both its running time and memory requirements.

It is worth noting that this new algorithm does not significantly affect group-action-based protocols such as CSI-FiSh [BKV19] and CSIDH [CLM+18], which already use substantially larger prime characteristics due to Kuperberg's algorithm. Our focus is therefore on protocols such as SQIsign and the CGL hash function, whose parameter choices are more directly affected.

Contributions. Our main contributions are as follows.

- We give a concrete complexity analysis of Wesolowski's $p^{1/3+o(1)}$ algorithm, making explicit the smoothness, modular-polynomial, and list-generation costs hidden in the $p^{o(1)}$ term. We optimize the smoothness bound B under time and memory constraints and provide estimates for 256- and 512-bit characteristics.
- We study the classical time–memory tradeoff and instantiate the van Oorschot–Wiener collision search [vOW99] and analyze its advantage.
- We compare these attacks with the Delfs–Galbraith line of work [SCS22,PRSU26]. Our estimates show that Wesolowski's algorithm can outperform these solvers at NIST Level I, but its memory requirements become prohibitive at higher security levels. In contrast, the parallelized vOW variant can outperform the accelerated Delfs–Galbraith algorithms when sufficiently many processors are available.
- We investigate quantum claw-finding improvements using the cost model of Jaques and Schanck [JS19]. We analyze Grover search and Tani's quantum walk in terms of gates, depth, and coherent quantum memory, showing that Tani's $O(N^{2/3})$ advantage is only query-theoretic and becomes a substantial time–memory tradeoff once the quantum data structure is included. Grover largely removes the search-memory bottleneck but offers little improvement in running-time dependence.
- We provide an optimized implementation of Wesolowski's algorithm and compare its empirical running time and space usage with our theoretical bounds. The measurements closely track the predicted complexities, and we additionally compare the implementation with SuperSolver of [SCS22].

Remark 1. Several subroutines appearing in these attacks are sufficiently involved that deriving exact implementation-level operation counts would obscure the main comparison. To keep the analysis tractable, we set hidden multiplicative constants to one and make explicit the terms that affect the exponential complexity. Our numerical results should therefore be viewed as leading-term estimates, rather than exact implementation costs. We, nevertheless, give some numerical estimates on these hidden constants in Section 7.

2 Wesolowski's Algorithm

In what follows we assume familiarity with Wesolowski's setup [Wes26] and refer the reader to section 2 therein for the relevant background.
Wesolowski's algorithm [Wes26] reduces the computation of a non-scalar endomorphism of a supersingular elliptic curve to finding an isogeny from the curve to its Frobenius conjugate. Let $E/\mathbb{F}_{p^2}$ be supersingular, and let $E^{(p)}$ denote its Frobenius conjugate. Given a separable isogeny

$$\phi : E \longrightarrow E^{(p)},$$

one obtains a non-scalar endomorphism of E by composing ϕ with the p-power Frobenius isogeny

$$\pi_p : E^{(p)} \longrightarrow E^{(p^2)} = E.$$

Indeed, $\pi_p \circ \phi$ has inseparable degree p, and hence cannot be a scalar endomorphism.

The attack exploits the recent result of [AOV26] that there always exists an isogeny $E \to E^{(p)}$ of degree at most

$$\left(\frac{p}{2}\right)^{1/3}.$$

Fix a smoothness bound B, and set

$$X = B^{1/2}\left(\frac{p}{2}\right)^{1/6}.$$

For a supersingular curve E, define

$$\mathcal{L}(E, X, B) = \{\psi : E \to E' \mid \ker(\psi) \text{ is cyclic and } \deg(\psi) < X \text{ is } B\text{-smooth}\}. \tag{1}$$

The elements of $\mathcal{L}(E, X, B)$ are enumerated by extending non-backtracking isogeny paths of prime degrees $\ell \leq B$. At each step, the ℓ-isogenous neighbors of an intermediate curve E' are obtained from the roots of the specialized modular polynomial $\Phi_\ell\big(j(E'), Y\big) \in \mathbb{F}_{p^2}[Y]$.

Suppose that the degree of the smallest isogeny $\phi : E \longrightarrow E^{(p)}$ is B-smooth. Writing

$$\deg(\phi) = \prod_{i=1}^{r} \ell_i, \qquad \ell_i \leq B,$$

one can choose a decomposition $\phi = \eta \circ \psi$ [Wes26, Proof of Lemma 3.4] such that

$$\deg(\psi) \leq X \qquad \text{and} \qquad \deg(\eta) \leq X.$$

Let $E' = \mathrm{cod}(\psi) = \mathrm{dom}(\eta)$, and define $\chi := \widehat{\eta}^{(p)}$. Then

$$\psi : E \longrightarrow E', \qquad \chi : E \longrightarrow E'^{(p)},$$

and both ψ and χ belong to $\mathcal{L}(E, X, B)$. Hence, the desired decomposition appears as a Frobenius claw in the smooth-isogeny list: the table contains two entries satisfying $\mathrm{cod}(\chi) = \mathrm{cod}(\psi)^{(p)}$. Given such a pair, the algorithm recovers the isogeny

$$\widehat{\chi}^{(p)} \circ \psi : E \longrightarrow E^{(p)}.$$

Since the smoothness condition need not hold for the original input curve, Wesolowski rerandomizes the instance by computing a non-backtracking walk $\omega : E \longrightarrow E_1$ of length $n = O(\log p)$ in the 2-isogeny graph, so that E_1 is statistically close to uniformly distributed among supersingular curves. Using [BCC+23, Lemma 14], $n = \lceil 4 \log p \rceil$ should be enough. The meet-in-the-middle procedure is then applied to E_1. If it returns an isogeny $\phi_1 : E_1 \longrightarrow E_1^{(p)}$, the algorithm outputs

$$\widehat{\omega} \circ \pi_{p,E_1} \circ \phi_1 \circ \omega \in \mathrm{End}(E) \setminus \mathbb{Z},$$

where $\pi_{p,E_1} : E_1^{(p)} \to E_1$ is the Frobenius isogeny. We then repeatedly invoke this OneEnd oracle until we obtain a second linearly independent endomorphism, and enlarge the order they generate to a maximal order, thereby recovering $\mathrm{End}(E)$ as in [EHL+20]. Under certain heuristics, this approach is more efficient than the rigorous approach outlined in [PW24].

Let D_E denote the degree of the smallest isogeny $E \to E^{(p)}$. Under the smoothness heuristic adopted in this work, one iteration succeeds with probability

$$\Pr[D_E \text{ is } B\text{-smooth}] \approx u^{-u(1+o(1))}, \qquad u = \frac{\log(p/2)}{3\log B}. \tag{2}$$

The expected number of rerandomized iterations is therefore approximately $u^{u(1+o(1))}$. The concrete running time is determined by this repetition factor together with the cost of constructing and searching $\mathcal{L}(E, X, B)$. In the following sections, we analyze the superpolynomial factors hidden by the $o(1)$ notation over the concrete parameter ranges relevant to SQIsign, the CGL hash function, and related isogeny-based constructions.

Remark 2. Since the main bottleneck in solving EndRing is obtaining a nontrivial endomorphism, which in our setting reduces to finding an isogeny from a curve to its Frobenius conjugate, we focus on this step throughout the paper. Accordingly, our running-time estimates measure the cost of finding such an isogeny and neglect lower-order polynomial overheads required to recover the full endomorphism ring.

3 Smoothness Estimates for Isogenies

3.1 Success probability of finding a Frobenius claw

The analysis of Wesolowski's algorithm requires estimating the probability that a randomly rerandomized curve contains an isogeny to its Frobenius conjugate whose degree belongs to the set of admissible smooth degrees. Previous analyses estimate this probability by requiring that a short isogeny has a smooth degree, resulting in a success probability of u^{-u} as in Equation (2). In this section, we refine this estimate by accounting for all smooth degrees that may lead to a valid Frobenius claw. Obtaining an exact expression for this probability appears difficult, and we therefore introduce a heuristic model supported by theoretical evidence and numerical experiments.

For a smoothness bound B and search bound X, let

$$\mathcal{S}(X, B) = \{m \le X : P^+(m) \le B\} \quad \text{and} \quad \Psi(X, B) = \#\mathcal{S}(X, B)$$

denote the set of B-smooth integers up to X, where $P^+(d)$ is the largest prime divisor of d, with $P^+(1) = 1$. The set of degrees considered by the meet-in-the-middle search is

$$\mathcal{D}_{B,X} = \{ab : a, b \in \mathcal{S}(X, B)\}.$$

Thus, every $n \in \mathcal{D}_{B,X}$ corresponds to an isogeny degree that can be represented as the product of two smooth factors and can therefore be searched by the claw-finding procedure.

For each admissible degree $n \in \mathcal{D}_{B,X}$, let $\mathsf{A}_{E,n}$ denote the event that the curve E admits a cyclic n-isogeny to its Frobenius conjugate $E^{(p)}$. The overall success event is

$$\mathsf{A}_{E,X,B} = \bigcup_{n \in \mathcal{D}_{B,X}} \mathsf{A}_{E,n}.$$

A heuristic estimate for the probability of this event was proposed in [BBD+24, Section 4.3]. We record it as a heuristic.

Heuristic 1. *For a fixed degree n, the probability that a supersingular curve E admits an n-isogeny to its Frobenius conjugate $E^{(p)}$ is*

$$\Pr[\mathsf{A}_{E,n}] \approx \sqrt{\frac{12n}{p}}.$$

This heuristic is also supported by the analysis of Eisenträger et al. [EHL+20]. In particular, for a fixed prime degree ℓ, they prove under GRH that the number of supersingular curves admitting an ℓ-isogeny to their Frobenius conjugate is

$$\Omega\left(\frac{\sqrt{\ell p}}{6(\ell+1)\log\log(\ell p)}\right).$$

Hence, their result is consistent with the $\sqrt{\ell/p}$ dependence predicted by Heuristic 1, up to lower-order logarithmic factor.

We now aggregate the contribution of all admissible degrees. Assuming that the events $\{\mathsf{A}_{E,n}\}_{n \in \mathcal{D}_{B,X}}$ follow a Poisson model, the expected number of successful degrees is

$$\mu_{B,X} = \sum_{n \in \mathcal{D}_{B,X}} \Pr[\mathsf{A}_{E,n}] \approx \sqrt{\frac{12}{p}} \sum_{n \in \mathcal{D}_{B,X}} \sqrt{n}.$$

This leads to the following heuristic estimate.

Heuristic 2. *Assuming Heuristic 1 and the Poisson model above, the success probability of the claw search is*

$$\Pr[\mathsf{A}_{E,X,B}] \approx 1 - \exp(-\mu_{B,X}).$$

Equivalently, defining the weighted smooth-degree mass

$$\Sigma_{B,X} = \sum_{n \in \mathcal{D}_{B,X}} \sqrt{n},$$

we obtain

$$\Pr[\mathsf{A}_{E,X,B}] \approx 1 - \exp\left(-\sqrt{\frac{12}{p}}\Sigma_{B,X}\right).$$

We implemented (an optimized version of) Wesolowski's algorithm together with an algorithm for computing the exact $\Sigma_{B,X}$ for moderate values of B and p. We measured the empirical success probability for different parameter choices and compared the results against both our heuristic estimate and the approximation u^{-u}. Figure 1 presents the empirical data along with the corresponding perfect-fit line, while the exact numerical values are provided in Table 5 in Appendix A. The data show that the proposed heuristic closely matches the empirical probability of finding a Frobenius claw. It also shows that the empirical probability of success is substantially larger than the smoothness probability u^{-u}.

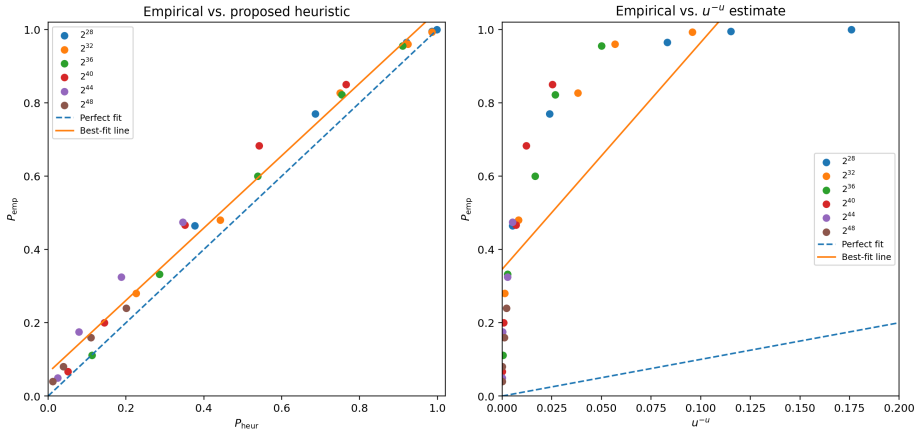

Fig. 1. Comparison between the empirical success probability, the proposed heuristic prediction from Heuristic 2, and the crude estimate $\Pr[D_E \text{ is } B\text{-smooth}] \approx u^{-u}$.

For large parameters, estimating $\Sigma_{B,X}$ requires understanding the distribution of integers that can be represented as products of two B-smooth integers of size at most X. This is closely related to the smooth multiplication table problem studied by Mehdizadeh [Meh17], which investigates the density and structure of sets of the form $\mathcal{S}(X,B)\mathcal{S}(X,B)$.

In particular, the main difficulty is determining the proportion of B-smooth integers up to X^2 that admit a balanced factorization into two factors of size at most X. Mehdizadeh's results [Meh17, Theorem 1.1, 1.2] provide asymptotic estimates for this density in certain parameter regimes. However, the assumptions underlying these asymptotics do not directly apply to our setting: they require $B = o(\log X)$, whereas the parameter choices arising in our application can be as large as

$$B \sim \exp\left(\frac{1}{3}\sqrt{\log p}\right) = \Omega(\log X).$$

Consequently, we do not rely on an asymptotic expression for $\Sigma_{B,X}$, but instead derive computable upper and lower bounds.

Let

$$W(Y,B) = \sum_{\substack{n \le Y \\ P^+(n) \le B}} \sqrt{n}$$

denote the weighted B-smooth counting function. Since every element of $\mathcal{D}_{B,X}$ is B-smooth and bounded by X^2, we have

$$\Sigma_{B,X} \le W(X^2, B).$$

Conversely, every B-smooth integer $n \le X^2/B$ admits a factorization $n = ab$ with $a, b \le X$: greedily assigning prime factors to one factor gives a divisor a satisfying $n/X \le a \le X$, since the remaining prime factors are at most B. Hence,

$$W(X^2/B, B) \le \Sigma_{B,X} \le W(X^2, B).$$

Combining these bounds with the above heuristic, noting that $1 - \exp(-\mu)$ is non-decreasing gives the following corollary.

Corollary 1. *Assuming Heuristic 2, the probability of finding a claw satisfies*

$$1 - \exp\left(-\sqrt{\frac{12}{p}} W(X^2/B, B)\right) \lesssim \Pr[\mathsf{A}_{E,X,B}] \lesssim 1 - \exp\left(-\sqrt{\frac{12}{p}} W(X^2, B)\right).$$

3.2 Estimating weighted smooth sums

To evaluate the bounds on $\Sigma_{B,X}$, we need to estimate the weighted smooth sum

$$W(Y,B) = \sum_{\substack{n \le Y \\ P^+(n) \le B}} \sqrt{n}.$$

This is a weighted analogue of the classical friable counting function $\Psi(Y,B)$, which counts integers up to Y whose prime factors are at most B.

We use the saddle-point method of Hildebrand and Tenenbaum [HT86]. Let $\alpha = \alpha(Y,B)$ be the unique solution of

$$\sum_{\ell \le B} \frac{\log \ell}{\ell^\alpha - 1} = \log Y,$$

where the sum is over primes $\ell \le B$. Define

$$\zeta(\alpha, B) = \prod_{\ell \le B} (1 - \ell^{-\alpha})^{-1} \quad \text{and} \quad \phi_2(\alpha, B) = \sum_{\ell \le B} \frac{\ell^\alpha (\log \ell)^2}{(\ell^\alpha - 1)^2}. \tag{3}$$

The saddle-point estimate of Hildebrand and Tenenbaum [HT86, Theorem 1] gives

$$\Psi(Y,B) \approx \frac{Y^\alpha \zeta(\alpha, B)}{\alpha \sqrt{2\pi \phi_2(\alpha, B)}}.$$

The following lemma extends this estimate to the weighted sum.

Lemma 1. *Under the saddle-point approximation above,*

$$W(Y,B) \approx \frac{\alpha}{\alpha+\frac{1}{2}}\sqrt{Y}\Psi(Y,B) \approx \frac{Y^{\alpha+\frac{1}{2}}\zeta(\alpha,B)}{(\alpha+\frac{1}{2})\sqrt{2\pi\phi_2(\alpha,B)}}.$$

Proof. By partial summation,

$$W(Y,B) = \sqrt{Y}\Psi(Y,B) - \frac{1}{2}\int_1^Y t^{-1/2}\Psi(t,B)\,dt.$$

The local form of the saddle-point approximation gives $\Psi(t,B) \approx \Psi(Y,B)(t/Y)^{\alpha}$. Therefore,

$$\int_1^Y t^{-1/2}\Psi(t,B)\,dt \approx \frac{\sqrt{Y}\Psi(Y,B)}{\alpha+\frac{1}{2}}.$$

Substituting into the partial summation formula yields

$$W(Y,B) \approx \left(1-\frac{1}{2\alpha+1}\right)\sqrt{Y}\Psi(Y,B) = \frac{\alpha}{\alpha+\frac{1}{2}}\sqrt{Y}\Psi(Y,B).$$

The second expression follows by substituting the Hildebrand–Tenenbaum estimate for $\Psi(Y,B)$.

We use this approximation to numerically evaluate the upper and lower bounds on $\Sigma_{B,X}$ and obtain concrete estimates for the claw success probability. For the cryptographic parameter ranges considered, particularly since B will not be relatively large, $\zeta(\alpha,B)$, $\phi_2(\alpha,B)$, and consequently $W(Y,B)$ can be computed quite accurately in practice.

4 Running-time Analysis of the Algorithm

As noted in Section 2, given an input curve E, we first apply a sufficiently long random walk so that the resulting curve E' is computationally indistinguishable from uniform. Following [BCC+23, Lemma 14], a walk length of $n = 4\log p$ suffices.

Starting from E' and its Frobenius conjugate $E'^{(p)}$, we then enumerate B-smooth isogenies of degree at most $B^{1/2}(p/2)^{1/6}$ until the two searches collide. If no collision is found, we rerandomize the input instance and repeat the procedure. As discussed in Section 3, the expected number of repetitions of this meet-in-the-middle attack is approximately $\Pr[\mathsf{A}_{E,X,B}]^{-1}$ and an interval for the number of rerandomizations can be constructed using Corollary 1 and Lemma 1; this gives a time range for our attack.

It therefore remains to analyze the cost of enumerating the required B-smooth isogenies, corresponding to Algorithm 1 of [Wes26] and then combine the results. The main results of this section are Theorem 1 and Corollary 2. However, due to the difficulty of obtaining a closed formula for $\Pr[\mathsf{A}_{E,X,B}]$, we choose to keep it implicit in the theorems' statements and refer the reader to the discussion after Corollary 2 for security estimates.

4.1 Listing B-smooth Isogenies

Following the suggestion in [Wes26, Footnote 2], we populate the list of B-smooth isogenies using batched evaluations of modular polynomials. This allows us to replace the multiplicative superpolynomial factor $B^{O(1)}$ by some additive term in the running time.

For a fixed prime $\ell \leq B$, we compute the modular polynomial $\Phi_\ell(X, Y)$ mod p and collect the j-invariants of all intermediate curves that admit an ℓ-isogeny extension and evaluate $\Phi_\ell(j, Y)$ simultaneously, using multi-point evaluation, at these points, rather than computing each specialization independently. The resulting polynomials are then split over $\mathbb{F}_{p^2}$, and their roots yield the corresponding ℓ-isogenous neighbors. This allows the cost of constructing and specializing Φ_ℓ to be amortized across the entire batch, while root extraction and table insertion remain part of the per-entry cost. We regard our list $\mathcal{L}$ as an indexed array where isogenies are represented by a sequence of curves rather than torsion points. One can then reconstruct the isogenies quite easily using Velu's formula. For our list, if its t-th entry represents an isogeny path $\psi_t : E \to E_t$, we write

$$\mathcal{L}[t] = \big(j(E_t), \pi_t, d_t\big),$$

where $d_t = \deg(\psi_t)$ and π_t is the index of the entry representing the path obtained from ψ_t by removing its final prime-degree step. The identity entry is initialized as

$$\mathcal{L}[1] = \big(j(E), 0, 1\big),$$

where 0 denotes the null pointer.

Lemma 2 (GRH). *Algorithm 1 is correct and runs in expected time*

$$O\left(B^4 \log^2 B \log\log B + (N + B^3 \log X)\frac{M(N+B)}{N+B}\log(N+B) + N\frac{M(B)}{B}\log B \log(Bp)\right)$$

under GRH, where $N = \#\mathcal{L}(E, X, B)$ and $\mathsf{M}(n)$ denotes the cost of multiplying two polynomials of degree at most n. Its local memory consumption is $O\left(B^2 \log p\right)$ aside from the size of the output list.

Proof. Algorithm 1 is essentially [Wes26, Algorithm 1], with two modifications. First, for every prime $\ell \leq B$, we compute $\Phi_\ell(U, V)$ mod p once and reuse it for all curves processed at the corresponding ℓ-adic layers. Second, the specializations $\Phi_\ell(j(E'), V)$ for all curves E' in a given layer are computed simultaneously using fast multipoint evaluation, rather than separately.

These modifications do not change the output of the algorithm and correctness therefore follows analogously to Wesolowski's listing algorithm.

Let $N = \#L(E, X, B)$ be the number of list entries.

We now analyze the cost of the two modifications. Using the BLS algorithm [BLS12, Theorem 1], the modular polynomial $\Phi_\ell(U, V)$ mod p, for a fixed prime ℓ, can be computed in time

$$O\big(\ell^3 \log^3 \ell \log\log \ell\big),$$

Algorithm 1 Listing B-smooth isogenies with batched evaluation

Require: A supersingular elliptic curve $E/\mathbb{F}_{p^2}$ and parameters B, X
Ensure: An indexed list $\mathcal{L}$ of parent-pointer records representing $\mathcal{L}(E, X, B)$

1: $\mathcal{L}[1] \leftarrow \mathsf{Entry}(\mathrm{id}_E) \leftarrow (j(E), 0, 1)$
2: **for all** primes $\ell \leq B$ **do**
3: Compute $\Phi_\ell(U, V) \bmod p$
4: **for** $i = 1, \ldots, \lfloor \log_\ell X \rfloor$ **do**
5: $\mathcal{F}_{\ell,i} \leftarrow \{t \in \{1, \ldots, |\mathcal{L}|\} : v_\ell(d_t) = i - 1,\ \ell d_t \leq X\}$
6: $\{\mathcal{N}_\ell(t)\}_{t \in \mathcal{F}_{\ell,i}} \leftarrow \textsc{BatchEvaluate}_\ell\left(\Phi_\ell, \{j_t\}_{t \in \mathcal{F}_{\ell,i}}\right)$
7: **for all** $t \in \mathcal{F}_{\ell,i}$, where $\mathcal{L}[t] = \mathsf{Entry}(\psi_t) = (j_t, \pi_t, d_t)$ **do**
8: **for all** ℓ-isogeny edges $\eta : E_t \to E''$ represented in $\mathcal{N}_\ell(t)$ **do**
9: **if** η does not reverse the final edge of ψ_t **then**
10: $s \leftarrow |\mathcal{L}| + 1$
11: $\mathcal{L}[s] \leftarrow \mathsf{Entry}(\eta \circ \psi_t) \leftarrow \left(j(E''), t, \ell d_t\right)$
12: **end if**
13: **end for**
14: **end for**
15: **end for**
16: **end for**
17:
18: **return** $\mathcal{L}$

and space complexity $O(\ell^2 \log \ell p)$ assuming GRH. Summing over all primes $\ell \leq B < p$ gives

$$\sum_{\substack{\ell \leq B \\ \ell \text{ prime}}} O\big(\ell^3 \log^3 \ell \log\log \ell\big) = O\big(B^4 \log^2 B \log\log B\big).$$

This is an additive preprocessing cost, since each modular polynomial is computed only once.

It remains to analyze the batched specializations $\textsc{BatchEvaluate}_\ell$ in step 6. For a prime $\ell \leq B$ and $1 \leq i \leq \lfloor \log_\ell X \rfloor$, let $\mathcal{F}_{\ell,i}$ denote the frontier processed in the i-th ℓ-adic layer. More precisely,

$$\mathcal{F}_{\ell,i} = \left\{\psi : E \longrightarrow E' \in L : \deg(\psi) = d\ell^{i-1},\ P^+(d) < \ell,\ \ell \deg(\psi) \leq X\right\},$$

where $P^+(d)$ denotes the largest prime divisor of d, with $P^+(1) = 1$. We write

$$m_{\ell,i} = \#\mathcal{F}_{\ell,i}.$$

Thus, $m_{\ell,i}$ is the number of parent isogenies whose codomains are processed simultaneously in the (ℓ, i)-th batch.

For each layer (ℓ, i), let $m_{\ell,i}$ denote the number of parent isogenies processed in that layer and $n_{\ell,i}$ the number of new entries generated from them. Each parent produces either ℓ or $\ell+1$ admissible extensions, according to whether the backtracking edge is removed. Hence

$$n_{\ell,i} = \Theta(\ell m_{\ell,i}).$$

Moreover, every nonidentity entry of $L(E, X, B)$ is generated in exactly one layer. Therefore,

$$\sum_{\ell,i} n_{\ell,i} = N - 1, \qquad \text{and consequently} \qquad \sum_{\ell,i} \ell m_{\ell,i} = \Theta(N).$$

Fix now a nonempty layer (ℓ, i), and let $j_1, \ldots, j_{m_{\ell,i}}$ be the j-invariants of the codomains of its parent isogenies. Write

$$\Phi_\ell(U, V) = \sum_{k=0}^{\ell+1} f_{\ell,k}(U) V^k,$$

where every $f_{\ell,k}$ has degree at most $\ell + 1$.

Let $\mathsf{M}(n)$ denote the cost of multiplying two polynomials of degree less than n over $\mathbb{F}_{p^2}$, and assume that $\mathsf{M}(n)/n$ is nondecreasing. Standard product- and remainder-tree multipoint evaluation [vzGG13, Corollary 10.8] computes

$$\Phi_\ell(j_r, V), \qquad 1 \le r \le m_{\ell,i},$$

in $O\left(\ell\, \mathsf{M}(m_{\ell,i} + \ell) \log(m_{\ell,i} + \ell)\right)$.

Put $Q = N + B$. Since $m_{\ell,i} + \ell \le Q$, monotonicity of $\mathsf{M}(n)/n$ gives

$$\begin{aligned} \ell \mathsf{M}(m_{\ell,i} + \ell) &= \ell(m_{\ell,i} + \ell) \frac{\mathsf{M}(m_{\ell,i} + \ell)}{m_{\ell,i} + \ell} \\ &\le \left(\ell m_{\ell,i} + \ell^2\right) \frac{\mathsf{M}(Q)}{Q}. \end{aligned}$$

Summing over all primes and layers therefore yields

$$\begin{aligned} T_{\text{batch}} &= O\left(\frac{\mathsf{M}(Q)}{Q} \log Q \left(\sum_{\ell,i} \ell m_{\ell,i} + \sum_{\ell,i} \ell^2\right)\right) \\ &= O\left(\left(N + B^3 \log X\right) \frac{\mathsf{M}(N+B)}{N+B} \log(N+B)\right), \end{aligned}$$

where we used

$$\sum_{\ell,i} \ell^2 = \sum_{\substack{\ell \le B \\ \ell \text{ prime}}} \ell^2 \left\lfloor \log_\ell X \right\rfloor = O(B^3 \log X)$$

We next account for root extraction. For every parent isogeny $\psi : E \to E'$, the specialized polynomial $\Phi_\ell(j(E'), V)$ has degree $\ell + 1$ and splits over $\mathbb{F}_{p^2}$. Using fast randomized finite-field factorization [vzGG13, Corollary 14.16], its roots can be computed in expected time $O\big(\mathsf{M}(\ell) \log \ell \log \ell p\big)$ operations. Since the layer (ℓ, i) contains $m_{\ell,i}$ specialized polynomials, its total root-extraction cost is

$$O\left(m_{\ell,i} \mathsf{M}(\ell) \log \ell \log(\ell p)\right).$$

Summing over all primes and layers gives

$$\begin{aligned} T_{\text{roots}} &= O\left(\sum_{\ell,i} m_{\ell,i}\mathsf{M}(\ell)\log\ell\log(\ell p)\right) \\ &\le O\left(\frac{\mathsf{M}(B)}{B}\log B\log(Bp)\sum_{\ell,i}\ell m_{\ell,i}\right) \\ &= O\left(N\frac{\mathsf{M}(B)}{B}\log B\log(Bp)\right). \end{aligned}$$

For the local memory, only one modular polynomial is needed at a time, and $\Phi_\ell(U,V) \bmod p$ occupies $O(\ell^2\log\ell p)$ bits [BLS12, Theorem 1]. Thus the size of the local memory needed is $O(B^2\log p)$.

Remark 3. We analyzed the running time of the algorithm component-wise: computing the Φ_ℓ's costs $B^4\log^2 B\log\log B$, batching costs $(N+B^3\log X)\frac{M(N+B)}{N+B}\log(N+B)$, and root extraction costs $N\frac{M(B)}{B}\log B\log(Bp)$. The corresponding plots and discussion are given in Appendix B.

Lemma 3. *Using the representation of Algorithm 1, each entry representing an isogeny $\psi \in \mathcal{L}$ can be encoded using*

$$s_{\text{entry}} = \lceil 2\log_2 p\rceil + \lceil\log_2(N+1)\rceil + \lceil\log_2 X\rceil$$

bits, where $N = \#L(E,X,B)$.

Proof. An isogeny $\psi_t : E \longrightarrow E'$ with $\deg(\psi) \le X$, in index t is represented by the record

$$\mathsf{Entry}(\psi) = \big(j(E'), \pi_t, \deg(\psi)\big),$$

where $d_t = \deg(\psi_t)$ and π_t is the index of the entry representing the path obtained from ψ_t by removing its final prime-degree step. Any j-invariant over $\mathbb{F}_{p^2}$ requires $\lceil 2\log_2 p\rceil$ bits.

The parent pointer takes one of N possible entry indices or the null value $\perp$, and hence requires

$$\lceil\log_2(N+1)\rceil$$

bits. Finally, since

$$1 \le \deg(\psi) \le \lfloor X\rfloor,$$

the value $\deg(\psi)-1$ can be encoded using $\lceil\log_2\lfloor X\rfloor\rceil$ bits. Summing the three contributions proves the result.

Lemma 4 ([Wes26, Lemma 3.4]). *Suppose that the smallest-degree isogeny $E \to E^{(p)}$ is B-smooth. Then Algorithm 1 returns a separable isogeny $\varphi : E \to E^{(p)}$.*

Algorithm 2 Finding a smooth isogeny to the Frobenius conjugate

Require: A supersingular elliptic curve $E/\mathbb{F}_{p^2}$ and a smoothness bound $2 \le B < p$.
Ensure: A smooth isogeny $E \to E^{(p)}$, or $\perp$.

$n \leftarrow \lceil 4 \log p \rceil$
Sample a non-backtracking walk $\omega : E \to E_1$ of length n in the 2-isogeny graph
$X \leftarrow B^{1/2}(p/2)^{1/6}$
$\mathcal{L} \leftarrow \mathcal{L}(E_1, X, B)$, the list of B-smooth cyclic isogenies $\psi : E_1 \to E'$ of degree at most X, computed using Algorithm 1
for $(E', \psi) \in \mathcal{L}$ **do**
    if $\mathcal{L}$ contains $(E'^{(p)}, \chi)$ with $\chi : E_1 \to E'^{(p)}$ **then**
        $\phi_1 \leftarrow \widehat{\chi}^{(p)} \circ \psi : E_1 \to E_1^{(p)}$
        return $\widehat{\omega}^{(p)} \circ \phi_1 \circ \omega : E \to E^{(p)}$
    end if
end for
return $\perp$

Next, we combine the numerical results from Section 3 with Lemma 2 to obtain a running-time bound for Algorithm 2. As noted before, we keep the term $\Pr[\mathsf{A}_{E,X,B}]$ implicit and analyze concrete sizes in the security analysis that follows Corollary 2.

Theorem 1 (GRH). *Let $E/\mathbb{F}_{p^2}$ be supersingular, let $2 \le B = p^{o(1)}$, and set $X = B^{1/2}(p/2)^{1/6}$. Let $u = \frac{\log(p/2)}{3 \log B}$. Assuming GRH, Heuristic 2 and fast polynomial multiplication $\mathsf{M}(n) = O(n \log n \log\log n)$, the expected running time of Algorithm 2 is*

$$T = O\Bigg(p^{1/3} B (\log B)^{1/2} (\log p)^{3/2} \Big(\log p \log\log p + (\log B)^2 \log\log B \Big) \cdot \Pr[\mathsf{A}_{E,X,B}]^{-1} \exp\left[-\frac{\log(p/2)}{6 \log B} \log\left(\frac{\log(p/2)}{6 \log B} \right) \right] \Bigg),$$

where $\Pr[\mathsf{A}_{E,X,B}]$ satisfies the bounds in Corollary 1. Its space complexity is

$$S = O\left(p^{1/3} B (\log B)^{1/2} (\log p)^{3/2} \exp\left(-\frac{\log(p/2)}{6 \log B} \log\left(\frac{\log(p/2)}{6 \log B} \right) \right) \right).$$

Proof. Set $v = \frac{\log X}{\log B} = \frac{u+1}{2}$. By Heuristic 2, one trial succeeds with probability approximately $\Pr[\mathsf{A}_{E,X,B}] \approx 1 - \exp(-\mu_{B,X})$, and hence the expected number of trials is $\Pr[\mathsf{A}_{E,X,B}]^{-1}$. By Lemma 2, the listing cost of one trial is

$$O\left(B^4 \log^2 B \log\log B + (N + B^3 \log X) \frac{M(N+B)}{N+B} \log(N+B) + N \frac{M(B)}{B} \log B \log(Bp) \right)$$

where $N = \#\mathcal{L}(E, X, B)$.

We first simplify the third term, namely the batched-evaluation term. Since $X = B^{1/2}(p/2)^{1/6} = p^{1/6+o(1)}$ and $B = p^{o(1)}$, we have $B = o(X)$. Moreover, the cyclic 2-power isogenies of degree below X already give $N = \Omega(X)$.

Hence $B^3 \log X = o(N)$ and $B = o(N)$, so the batched-evaluation term is $O(\mathsf{M}(N) \log N)$.

On the other hand, [Wes26, Lemma 3.2] gives

$$N = O\big(\Psi(X, B) X \log X\big) = O\big(X^2 v^{-v} \log X\big). \tag{4}$$

We have $N = O(X^2 \log X) = p^{O(1)}$, and therefore $\log N = O(\log p)$. Fast polynomial multiplication consequently gives

$$T_{\text{batch}} = O(N (\log p)^2 \log \log p).$$

Using the Dickman–de Bruijn estimate $\Psi(X, B) \approx X\rho(v) \approx X v^{-v}$, we further obtain $N = O(X^2 v^{-v} \log X)$. The preprocessing contributes $O(B^4 \log^2 B \log \log B)$ since $\Phi_\ell(U, V) \bmod p$ can be used in all instances of rerandomization, while batching and root extraction together contribute

$$O(\Pr[\mathsf{A}_{E,X,B}]^{-1} X^2 v^{-v} \log X \, (\log^2 p \log \log p + B^{-1} \mathsf{M}(B) \log B \log(Bp))$$

.

Since $X^2 = B \cdot (p/2)^{1/3}$, we have

$$\begin{aligned} v^{-v} &= \sqrt{\frac{2}{e}}\, u^{-1/2} \left(\frac{2}{u}\right)^{u/2} \left(1 + O\left(\frac{1}{u}\right)\right) \\ &= O\left(\sqrt{\frac{\log B}{\log p}} \exp\left(-\frac{\log(p/2)}{6 \log B} \log\left(\frac{\log(p/2)}{6 \log B}\right)\right)\right). \end{aligned} \tag{5}$$

Furthermore, $\log X = \Theta(\log p)$, and $\mathsf{M}(B)/B = O(\log B \log \log B)$. Substitution gives the claimed running-time bound.

For space, trials are executed sequentially, so the expected number of trials does not affect the peak local memory. Letting $N = \#L(E, X, B)$, and using Lemma 2, Lemma 3, and Equation (4) we have

$$S_{\text{attack}} = O\big((B^2 + N) \log p\big) = O(X^2 v^{-v} \log X \log p).$$

Using Equation (5) yields space complexity

$$O\left(p^{1/3} B (\log B)^{1/2} (\log p)^{3/2} \exp\left(-\frac{\log(p/2)}{6 \log B} \log\left(\frac{\log(p/2)}{6 \log B}\right)\right)\right),$$

as claimed.

Corollary 2 (GRH). *Under the same hypotheses of Theorem 1, and letting* $B = \exp(\frac{1}{3}\sqrt{\log p})$*, Algorithm 2 runs in time*

$$\begin{aligned} T = O\Bigg(& p^{1/3} (\log p)^{11/4} \log \log p \cdot \Pr[\mathsf{A}_{E,X,B}]^{-1} \\ & \cdot \exp\left[-\frac{1}{4}\sqrt{\log p} \log \log p + \left(\frac{1}{3} + \frac{\log 2}{2}\right)\sqrt{\log p}\right]\Bigg). \end{aligned}$$

and space

$$S = O\left(p^{1/3}\log^{7/4}(p)\cdot\exp\left(-\frac{1}{4}\sqrt{\log p}\log\log p + \left(\frac{1}{3}+\frac{\ln 2}{2}\right)\sqrt{\log p}\right)\right).$$

Optimizing for time. We now discuss the implications of Theorem 1 for the hardness of the isogeny-path problem, and consequently for the concrete security of SQIsign and related isogeny-based primitives. We note again that our running-time estimates measure the cost of finding such an isogeny from a curve to its conjugate and neglect lower-order polynomial overheads required to recover the full endomorphism ring. Following Wesolowski's approach, we first consider the choice of the smoothness bound B that minimizes the expected running time. Using Corollary 1 and Lemma 1, we aim to find a solution for

$$\min_B T$$

with no constraint on space S. We use standard discrete search minimization techniques to search for the optimum B. Table 1 reports the resulting time–space tradeoff for 256- and 512-bit primes p. In particular, with such values for B, $\zeta(\alpha, B)$ and $\phi_2(\alpha, B)$ in Equation (3) can be computed exactly.

B	$\log_2(1/P)$	$\log_2 T$	$\log_2 S$
1 000	[16.23, 26.20]	[125.19, 135.16]	98.98
10 000	[1.49, 16.54]	[117.67, 132.72]	106.06
13 187	[0.51, 15.72]	[**117.41**, 132.63]	106.77
24 137	[0.00, 14.13]	[118.43, **132.56**]	108.25
100 000	[0.00, 11.16]	[121.72, 132.89]	111.44

B	$\log_2(1/P)$	$\log_2 T$	$\log_2 S$
100 000	[16.12, 35.62]	[217.01, 236.51]	189.65
1 000 000	[2.20, 26.82]	[209.78, 234.40]	196.25
1 560 271	[0.49, 25.49]	[**209.23**, 234.23]	197.38
4 992 737	[0.00, 22.46]	[211.60, **234.06**]	200.19
10 000 000	[0.00, 20.89]	[213.23, 234.12]	201.77

Table 1. Estimated time-intervals for $\log_2 p = 256$ (left) and 512 (right), where $P = \Pr[\mathsf{A}_E]$. Lower endpoints of the time intervals are bold. Bold B-values minimize these lower endpoints among the displayed choices.

Table 1 shows that refined analysis gives higher cost estimates than Wesolowski's quoted optimistic estimates [Wes26, section 4.1]. For the 256-bit case, the minimum lower time estimate is approximately $2^{117.41}$, with corresponding space $2^{106.77}$, compared with $2^{106.5}$ and $2^{92.5}$, respectively. For the 512-bit case, the corresponding estimates are $2^{209.23}$ and $2^{197.38}$, compared with $2^{204.2}$ and $2^{181.3}$. Thus, the refined optimistic time estimates increase by approximately 10.91 and 5.03 bits, respectively. The width of the reported time intervals reflects the remaining uncertainty in the success probability.

Although some estimates fall below the 128- and 256-bit security levels, they come at the cost of prohibitive memory requirements at both parameter sizes. We note that the reported values are heuristic cost estimates with hidden multiplicative constants set to one, rather than exact low-level implementation count.

	$\log p = 256$			$\log p = 512$		
k	$B_k^\star$	$\log_2 S(B_k^\star)$	$\log_2 T_k^\star$	$B_k^\star$	$\log_2 S(B_k^\star)$	$\log_2 T_k^\star$
50	–	–	infeasible	–	–	infeasible
60	2	56.2640	[151.19, 151.69]	–	–	infeasible
70	13	69.7542	[145.59, 147.74]	–	–	infeasible
80	67	79.8915	[138.15, 142.47]	–	–	infeasible

Table 2. Time-intervals with memory-constrained optimal parameters for $p \approx 2^{256}$ and $p \approx 2^{512}$. For each space budget $S \leq 2^k$, $B_k^\star$ minimizes the estimated running time over the feasible values of B.

Optimizing for space. In practice, adversaries are often more memory constrained than time-constrained. We therefore analyze the attack under memory budgets ranging from 2^{50} (roughly 141 TB) to 2^{80} bits of storage [1] . For each memory budget, we solve the optimization problem

$$\min_{B, S_B \leq 2^k} T(B),$$

for $k = 50, 60, 70, 80$. Table 2 summarizes the resulting time–space tradeoffs. Overall, the concrete estimates show that even nation-state adversaries with extremely large memory resources remain computationally constrained. For instance, with as much as 2^{80} bits of storage, the attack still requires at least $T_{256} = 2^{138.15}$ binary operations for a 256-bit prime.

We note three interesting points. First, directly using the space complexity term in Theorem 1 results in inaccurate concrete estimates. The reason is that, with very limited space, B is expected to be quite small, typically between 2 and 70. This makes the estimate for smooth integers $\Psi(X, B) \approx Xv^{-v}$ highly inaccurate. For instance, if $B = 2$ and $X = B^{1/2}(p/2)^{1/6} = \sqrt{2}(2^{255})^{1/6} \approx 2^{43}$, then v^{-v} is extraordinarily small at $B = 2$, roughly $2^{-233.31}$. Thus, $Xv^{-v} \approx 2^{-190}$, which does not make sense, whereas in fact $\Psi(X, 2) = 44$, corresponding to the 2-smooth integers $1, 2, 2^2 \ldots 2^{43}$.
Second, the case $S \approx 2^{50}$ is infeasible because even the smallest choice $B = 2$ would result in $\log_2(S(2)) \approx 56.26$, which is already above the available budget. Finally, for the 512-bit prime, all considered space-memory budgets are insufficient for the attack to take place. First, note that the space complexity is minimized at $B = 2$. Since $X = B^{1/2}(p/2)^{1/6} \approx 2^{85.66}$, we have $\Psi(X, 2) = 86$. Then our refined space formula $S = p^{1/3} B \log X \log p \frac{\Psi(X,B)}{X}$ gives $\log_2(S(2)) \approx 106$, which is already above our extreme budget 2^{80}.

[1] For scale, Gartner estimated global data-center storage capacity at roughly 1 zettabyte (ZB) in 2024, corresponding to approximately $2^{72.8}$ bits, with projections reaching 2.49 ZB by 2028. By comparison, 2^{80} bits of storage is approximately 151 ZB.

5 The van Oorschot–Wiener technique

The analysis above stores the entire set of smooth isogenies $\mathcal{L}(E, X, B)$, giving the lowest running time among the variants considered so far, but requiring memory proportional to the search space. Wesolowski instead formulates the matching step as a claw-finding problem and applies the van Oorschot–Wiener (vOW) time–memory tradeoff [Wes26,vOW99], previously studied for supersingular-isogeny cryptanalysis in [CLN+20]. We show that the plain vOW does not outperform Algorithm 2 except when high parallelization is considered. The main result is Theorem 2 and the security analysis that follows.

The generic vOW algorithm. Let $\mathcal{S}$ be a search space of size N, and let $h : \mathcal{S} \to \mathcal{Y}$ be an expensive function whose desired, or *golden*, collision solves the underlying problem. For each function version ν, vOW defines

$$f_\nu = g_\nu \circ h,$$

where $g_\nu : \mathcal{Y} \to \mathcal{S}$ is pseudorandom, and generates trails $x_{i+1} = f_\nu(x_i)$ from random starting points.

A fraction θ of $\mathcal{S}$ is designated as *distinguished points*. When a trail reaches a distinguished point x_d, only its start, endpoint, and length (x_0, x_d, d) are stored. If two trails reach the same endpoint, they are retraced to locate $x \neq y$ with $f_\nu(x) = f_\nu(y)$. Since this may be a collision introduced only by g_ν, one checks whether $h(x) = h(y)$ and, if so, whether it is the desired golden collision.

The function version is periodically changed to avoid poor random-walk instances. Following [CLN+20], one may use

$$\theta = \alpha\sqrt{\frac{w}{N}}, \qquad \alpha \approx 2.25,$$

and change the function after approximately $10w$ distinguished points, where w is the number of stored records.

The generic vOW analysis gives an expected number of step-function evaluations of order

$$\kappa_{\mathrm{vOW}}\sqrt{\frac{N^3}{w}},$$

with $\kappa_{\mathrm{vOW}} \approx 2.5$ as an optimistic concrete estimate from [CLN+20]. This constant is heuristic and depends on distinguished-point density, collision checking, function changes, and finite-size random-walk effects.

Thus, reducing storage from N to w increases the work by roughly $\sqrt{N/w}$, while requiring only w distinguished-point records. The trail generation is naturally parallelizable.

5.1 Instantiation of vOW to Wesolowski's algorithm

Indexing smooth isogenies. We now instantiate the generic construction in our setting. As before, set $X = B^{1/2}(p/2)^{1/6}$. Rather than constructing and

storing $\mathcal{L}(E, X, B)$, we represent its elements by compact combinatorial descriptions.

Let $d = \prod_{i=1}^{s} \ell_i^{e_i} \in \mathcal{S}(X, B)$, with $\ell_1 < \cdots < \ell_s$, and set $R_i = (\ell_i + 1)\ell_i^{e_i - 1}$. We define

$$\mathcal{I}_{X,B} := \bigsqcup_{d \in \mathcal{S}(X,B)} \left(\{d\} \times \prod_{\ell_i^{e_i} \| d} \{0, \ldots, R_i - 1\} \right) \tag{6}$$

Thus an element $\mathbf{r} = (d, r_{\ell_1}, \ldots, r_{\ell_s}) \in \mathcal{I}_{X,B}$ specifies both the smooth degree d and the local branch choices defining the corresponding isogeny. We fix a deterministic decoding map

$$\mathsf{Decode}_{E,X,B} : \mathcal{I}_{X,B} \to \mathcal{L}(E, X, B),\ \mathbf{r} \mapsto \psi_{\mathbf{r}},$$

which reconstructs this isogeny on demand.

For each prime-power component $\ell_i^{e_i}$, a cyclic $\ell_i^{e_i}$-isogeny is represented by a non-backtracking ℓ_i-isogeny path of length e_i. There are $\ell_i + 1$ choices for the first edge and ℓ_i choices thereafter, since the dual of the preceding edge is excluded, giving exactly R_i possible paths. The integer $r_{\ell_i} \in \{0, \ldots, R_i - 1\}$ encodes these successive branch choices. Namely, if $a_{i,0} \in \{0, \ldots, \ell_i\}$ and $a_{i,k} \in \{0, \ldots, \ell_i - 1\}$ for $1 \le k < e_i$, then

$$r_{\ell_i} = a_{i,0} + (\ell_i + 1) \sum_{k=1}^{e_i - 1} a_{i,k} \ell_i^{k-1}.$$

Conversely, $a_{i,0}$ is obtained by reducing r_{ℓ_i} modulo $\ell_i + 1$, while the remaining digits are recovered from the quotient in base ℓ_i. Hence r_{ℓ_i} is a mixed-radix encoding of the corresponding non-backtracking $\ell_i^{e_i}$-path.

To reconstruct the actual path, suppose that the current curve is F, the current prime is ℓ, and, when applicable, the preceding curve is F^-. We compute $\Phi_\ell(j(F), Y)$ and deterministically order its outgoing ℓ-isogeny edges according to a fixed canonical encoding of their codomain; say through increasing lexicographic order of j-invariants $j = a + bi = (a, b)$ with $\mathbb{F}_{p^2} = \mathbb{F}_p(i)$. For the first step of an ℓ^e-block, $a_{i,0} \in \{0, \ldots, \ell\}$ selects the corresponding edge. At each subsequent step, the dual edge returning to F^- is first removed, the remaining ℓ edges are canonically ordered, and $a_{i,k} \in \{0, \ldots, \ell - 1\}$ selects the next edge.

The prime-power blocks are evaluated in the fixed order $\ell_1 < \cdots < \ell_s$. Since these degrees are pairwise coprime, their composition has cyclic kernel of degree d. Conversely, every cyclic subgroup of order d decomposes uniquely into its ℓ_i-primary components, so this fixed ordering removes the ambiguity arising from permuting the prime-power factors. Consequently, the same tuple $\mathbf{r}$ always reconstructs the same $\psi_{\mathbf{r}} : E \to E_{\mathbf{r}}$.

Since the desired collision is between Frobenius-conjugate codomains, we identify j and j^p before applying the random map. Fixing an encoding $\mathbb{F}_{p^2} = \mathbb{F}_p(i)$, define

$$\mathrm{can}(j) := \min\{\mathrm{enc}(j), \mathrm{enc}(j^p)\}, \qquad h_E(\mathbf{r}) := \mathrm{can}(j(E_{\mathbf{r}})),$$

We then define

$$f_{\nu,E} = g_\nu \circ h_E, \qquad \text{where } g_\nu : \mathrm{Im}(h_E) \to \mathcal{I}_{X,B}$$

is modeled as a pseudorandom function. Pictorially,

$$\boxed{\mathbf{r} \xrightarrow{\text{decode}} (a_{i,k}) \xrightarrow{\Phi_{\ell_i}} \text{edges} \xrightarrow{a_{i,k}} \psi_{\mathbf{r}} \xrightarrow{\text{can}} h_E(\mathbf{r}) \xrightarrow{g_\nu} f_{\nu,E}(\mathbf{r})}.$$

A collision of $f_{\nu,E}$ need not be useful. When two colliding predecessors $r \neq s$ are recovered, we therefore recompute their actual endpoint invariants. If $j(E_r) = j(E_s)^p$, we compute an isomorphism $\iota : E_r \to E_s^{(p)}$, and obtain $\widehat{\psi_s}^{(p)} \circ \iota \circ \psi_r : E \longrightarrow E^{(p)}$. Collisions introduced solely by g_ν, as well as collisions satisfying $j(E_r) = j(E_s)$ rather than the required Frobenius relation, are discarded.

There is one additional difference from the generic claw-finding problem. A golden collision is not guaranteed to exist for every randomized curve E. We therefore run vOW for a fixed budget sufficient to find a golden collision with constant probability conditioned on its existence, and rerandomize the curve when this budget is exhausted.

Cost of the step function. The generic vOW complexity counts evaluations of $f_{\nu,E}$, but in our setting each evaluation requires computing $h_E(r)$, namely decoding r and evaluating the corresponding smooth isogeny. If its degree is $d = \prod_i \ell_i$, this amounts to a path

$$E = E_0 \xrightarrow{\ell_1} E_1 \xrightarrow{\ell_2} \cdots \xrightarrow{\ell_t} E_t, \qquad t = \Omega(d) \leq \log_2 X,$$

where $\Omega(\cdot)$ is the prime-divisor counting function and each step requires specializing $\Phi_{\ell_i}(j(E_{i-1}), Y)$, removing the backtracking edge when needed, and selecting the indexed branch.

Unlike the batched enumeration of Theorem 1, a vOW trail is adaptive: the next state $r' = g_\nu(h_E(r))$ is unknown until the current isogeny has been fully evaluated, and only one branch of each specialization is used. Hence the batching available to Algorithm 2 does not directly apply. Independent trails could still be batched by ℓ, but we do not assume such amortization; it can only improve the bounds below.

Precomputed modular polynomials. For each prime $\ell \leq B$, we compute $\Phi_\ell(X, Y) \bmod p$ using the BLS algorithm [BLS12] and reuse it across all vOW trails, function versions, and rerandomized curves.

Writing $\Phi_\ell(X, Y) = \sum_{k=0}^{\ell+1} c_{\ell,k}(X) Y^k$, specialization at $j \in \mathbb{F}_{p^2}$ costs $O(\ell^2)$ field operations without cross-trail batching. Fast randomized factorization [vzGG13, Corollary 14.16] gives expected root-extraction cost $O(\mathsf{M}(\ell) \log \ell \log(\ell p))$. Hence one prime-degree step costs

$$C_{\text{store}}(B, p) := O\big(B^2 + \mathsf{M}(B) \log B \log p\big), \tag{7}$$

Algorithm 3 vOW search for a smooth isogeny to the Frobenius conjugate

Require: A supersingular curve $E/\mathbb{F}_{p^2}$, smoothness bound B, memory for w records.
Ensure: A separable isogeny $\phi : E_1 \to E_1^{(p)}$ for curve E_1, and walk $\omega : E \to E_1$.

$X \leftarrow B^{1/2}(p/2)^{1/6}$; $\mathcal{I}_{X,B}$ as in Equation (6)
Precompute and store $\Phi_\ell(X, Y)$ mod p for all primes $\ell \le B$
while true **do**
  Sample walk $\omega : E \to E_1$; define $h(\mathbf{r}) = \text{canonical}(j(\text{cod}(\mathsf{Decode}_{E_1,X,B}(\mathbf{r}))))$
  $\theta \leftarrow \alpha\sqrt{w/N}$; $Q \leftarrow \lceil \kappa_{\text{vOW}}\sqrt{N^3/w} \rceil$; $q \leftarrow 0$, $\nu \leftarrow 0$
  while $q < Q$ **do**
    $f \leftarrow g_\nu \circ h$; initialize DP table of w slots; $c \leftarrow 0$
    while $c < \beta w$ **and** $q < Q$ **do**
      Sample $\mathbf{r}_0 \leftarrow \mathcal{I}_{X,B}$; $\mathbf{r} \leftarrow \mathbf{r}_0$, $d \leftarrow 0$
      while $\mathbf{r}$ not distinguished **and** $d < 20/\theta$ **and** $q < Q$ **do**
        $\mathbf{r} \leftarrow f(\mathbf{r})$; $d \leftarrow d + 1$; $q \leftarrow q + 1$
      end while
      if $\mathbf{r}$ is distinguished **then**
        $c \leftarrow c + 1$
        if the slot for $\mathbf{r}$ contains a trail ending at $\mathbf{r}$ **then**
          Retrace trails to get colliding predecessors $a \neq b$; compute ψ_a, ψ_b
          if $j(E_a) = j(E_b)^p$ **then**
            Compute $\iota : E_a \to E_b^{(p)}$; **return** $(\widehat{\psi_b}^{(p)} \circ \iota \circ \psi_a, \omega)$
          end if
        else
          Store (r_0, r, d) in the slot assigned to r, overwriting the previous record if necessary
        end if
      end if
    end while
    $\nu \leftarrow \nu + 1$
  end while
end while

and, since a smooth isogeny contains at most $\log_2 X$ such steps,

$$t_{\text{store}} = O(C_{\text{store}}(B, p) \log X) .$$

The modular-polynomial database requires $S_\Phi = O\left(\frac{B^3 \log p}{\log B}\right)$ bits. Although asymptotically smaller than the full smooth-isogeny table, this can dominate the memory of a small vOW collision table. For example, $B = 17400$ and $\log_2 p = 256$ give $S_\Phi \approx 2^{46.4}$ bits, or about 12 TB, which is a modest space requirement.

Theorem 2 (vOW-stored). *Let $E/\mathbb{F}_{p^2}$ be supersingular, let $X = B^{1/2}(p/2)^{1/6}$, and let $N = \#\mathcal{L}(E, X, B)$. Assume Heuristic 2, model the functions g_ν as random functions, and let w be the number of distinguished-point records available to the vOW algorithm. Assuming GRH and all modular polynomials Φ_ℓ mod p, for primes $\ell \le B$, are precomputed and stored using BLS algorithm, Algorithm 3*

has expected running time

$$T_{\mathrm{vOW}}^{\mathrm{store}} = O\left(B^4 \log^2 B \log\log B + \Pr[A_{E,X,B}]^{-1} \exp\left(-\frac{\log(p/2)}{4\log B} \log\left(\frac{\log(p/2)}{6\log B} \right) \right) \right.$$
$$\left. \times \frac{p^{1/2} B^{5/2}}{\sqrt{w}} (\log p)^{7/4} (\log B)^{3/4} \left[B + (\log B)^2 \log\log B \log p \right] \right).$$

Its space requirement is $O\left(w \log N + B^3 \log p \log^{-1} B\right)$ *bits, up to lower-order indexing and working storage.*

Proof. Let $v := \frac{\log X}{\log B} = \frac{1}{2} + \frac{\log(p/2)}{6\log B} = \frac{u+1}{2}$. Using Equation (4)

$$N = O\left(X^2 v^{-v} \log X\right).$$

Hence, conditioned on the existence of a golden collision, the generic vOW analysis requires

$$O\left(\kappa_{\mathrm{vOW}} \sqrt{\frac{N^3}{w}} \right) = O\left(\frac{\kappa_{\mathrm{vOW}}}{\sqrt{w}} X^3 v^{-3v/2} (\log X)^{3/2} \right)$$

evaluations of the iteration function. Numerical experiments [vOW99] recommend setting $\kappa_{\mathrm{vOW}} = 2.5$ and thus can be absorbed into the hidden constant.

For an indexed smooth isogeny of degree $d < X$, the cost of its prime-degree steps is bounded by

$$O\left(\log X \left[B^2 + \mathsf{M}(B) \log B \log p\right]\right).$$

Using $\mathsf{M}(B) = O(B \log B \log\log B)$, one vOW function evaluation therefore costs

$$O\left(\log X \left[B^2 + B(\log B)^2 \log\log B \log p\right]\right).$$

Under Heuristic 2, a rerandomized curve admits the required B-smooth isogeny with probability $\Pr[\mathsf{A}_{E,X,B}]$ satisfying the bounds in Corollary 1. Combining the above estimates, and using $X^3 = B^{3/2} \left(\frac{p}{2}\right)^{1/2}$, gives time complexity

$$O\left(B^4 \log^2 B \log\log B + \frac{p^{1/2}}{\sqrt{w}} (\log p)^{5/2} \left[B^{7/2} + B^{5/2} (\log B)^2 \log\log B \log p \right] \Pr[A_{E,X,B}]^{-1} v^{-3v/2} \right)$$

where we used $\log X = O(\log p)$, and the $B^4 + B^3 \log p$ term accounts for the one-time construction of the modular-polynomial database. Using Equation (5), the running time follows.

For the space complexity, the w distinguished-point records require $O(w \log N)$ bits, in virtue of Lemma 3. Moreover, storing all modular polynomials $\Phi_\ell \bmod p$, for primes $\ell \le B$, requires $O\left(\log p \sum_{\substack{\ell \le B \\ \ell \text{ prime}}} \ell^2 \right) = O\left(\frac{B^3 \log p}{\log B} \right)$ bits. The remaining indexing and working storage is lower order.

$\log_2 p$	k	$B^\star_{\text{low}}$	$B^\star_{\text{high}}$	$\log_2 T^\star_k$
256	50	3	3	[157.52, 158.34]
256	60	3	3	[152.52, 153.34]
256	70	3	3	[151.11, 151.93]
256	80	≈ 2803	3	[**149.54**, **151.93**]
512	50	≈ 20443	3	[311.01, 313.36]
512	60	≈ 67993	3	[305.26, 308.36]
512	70	≈ 75797	3	[300.25, 303.36]
512	80	≈ 75797	3	[**295.25**, **298.36**]

Table 3. Running-time intervals for the stored-vOW variant. For each budget $S \le 2^k$, $B^\star_{\text{low}}$ minimizes the optimistic running-time bound obtained from the upper bound on $\Pr[A_{E,X,B}]$, while $B^\star_{\text{high}}$ minimizes the conservative bound obtained from the lower bound on $\Pr[A_{E,X,B}]$.

Remark on other alternatives. One may alternatively evaluate isogenies using explicit torsion points. Given a basis $E[\ell] = \langle P, Q\rangle$, the $\ell + 1$ outgoing isogenies can be indexed by $\langle Q\rangle$ and $\langle P + [a]Q\rangle$, $a \in \mathbb{F}_\ell$, providing deterministic branch selection without evaluating $\Phi_\ell(j, Y)$.

The drawback is that the required ℓ-torsion is generally defined over an extension of degree $O(\ell)$ [EPSV24]. Using the approach of [BBC+25, Section 4.4], this gives $C_{\text{tors}}(\ell, p) = \widetilde{O}(\ell^2)$ base-field operations, but over extension fields whose degree grows with ℓ. Thus, although asymptotically comparable to the precomputed modular-polynomial cost

$$C_{\text{store}}(\ell, p) = O\big(\ell^2 + \mathsf{M}(\ell)\log\ell\log p\big)\,,$$

field arithmetic becomes substantially more expensive in practice for ℓ near B.

The torsion approach is thus attractive over small extensions, while modular-polynomial evaluation is more natural for generic $\ell \sim B$; we leave a concrete comparison to future work.

5.2 Effect of vOW on concrete security.

Stored-mode case. Interestingly, under the considered memory budgets the stored-vOW variant does not outperform the full attack. For example, when $\log_2 p = 256$ and $S = 2^{80}$, the full attack has estimated running time $2^{[138.15,\,142.47]}$, whereas the stored-vOW variant requires $2^{[149.54,\,151.93]}$. The reason is that the vOW collision search introduces a factor of the form

$$\sqrt{\frac{N^3}{w}} = N\sqrt{\frac{N}{w}},$$

rather than a cost $O(C_{\text{list}}(B)N\Psi(X, B)/X)$ as in the full attack. Hence, relative to the full attack, vOW incurs an additional factor $\sqrt{N/w}$. Since $w \le N$ and the available memory also has to store the modular-polynomial data $B^3 \log p/\log B$, a budget of 2^{80} bits would result in $\sqrt{w} \lesssim 2^{40}$ and is not large enough to make

this penalty negligible. The value $B^{\star}_{\text{low}} \approx 2800$ for $\log_2 p = 256$ and $k = 80$ arises from the tradeoff between the inverse success probability and the cost of increasing B. Increasing B substantially improves $\Pr[A_{E,X,B}]$, thereby reducing the dominant factor $\Pr[A_{E,X,B}]^{-1}$. At the same time, however, it increases the precomputation terms $B^4 + B^3 \log p$, the direct-evaluation cost, and the storage term $B^3 \log p / \log B$. For the optimistic probability bound, the best balance under a 2^{80}-bit budget occurs around $B \approx 2800$. Beyond this point, the gain from improving the success probability is outweighed by the rapidly increasing polynomial cost in B. The conservative probability bound behaves differently and remains minimized at the smallest admissible value $B = 3$, which explains the relatively wide interval in the $k = 80$ row.

Since vOW is perfectly parallelizable, having m processors reduces running time to $T^{\text{store}}_{\text{vOW}}/m$. As we will see next this will play an important role in security analysis.

Comparison with previous EndRing algorithms. Here we assess the concrete implications of these tradeoffs for supersingular endomorphism-ring computation. Since the isogeny-path and endomorphism-ring problems are polynomial-time equivalent [Wes22,EHL+18], the Delfs–Galbraith algorithm provides the standard baseline. Its accelerated variant, SuperSolver [SCS22], was estimated to break SQIsign-I in roughly $58\sqrt{p}\,\mathsf{M}_p$ operations when $\#\mathrm{Cl}((1+\sqrt{-p})/2) \approx \sqrt{p}$, where M_p denotes the cost of one multiplication in $\mathbb{F}_p$.
Very recently, [PRSU26] presented a GPU implementation of the attack and introduced Hypersolver, which is argued to be roughly 6 times faster than Supersolver of [SCS22] at NIST-I security level. They also estimated the operation count of the algorithm and obtained experimental estimates suggesting that, for the SQIsign-I prime p, the algorithm has a dominant term of

$$\approx 3.67 \cdot p^{1/2}\mathsf{M}_p.$$

The authors then provided estimates for the number of gates required to break SQIsign, namely 2^{143} and 2^{272} gates for NIST-I and NIST-V, respectively. Incorporating the estimates in [PRSU26, Table 2] for SQIsign's primes, we obtain Table 4.

For a consistent comparison, we convert the running times of our attacks to the same gate count for $\mathbb{F}_p$-multiplication M_p adopted in [PRSU26] so that all entries in Table 4 are expressed in gates; we also use SQIsign's primes which are a bit smaller than 256 and 512 bits. We note that this causes a bit of a difference with Table 2 and Table 3. We refer the interested reader to Appendix C for a discussion on how these estimates are obtained.
The data show that Algorithm 2 is already competitive with HyperSolver and, with further optimizations, is expected to outperform the aforementioned algorithms, albeit only at the NIST Level I security setting, since the full attack becomes infeasible at 512-bit prime due to its prohibitive memory requirements. The vOW attack is also estimated to be slower than HyperSolver and SuperSolver at these parameters. However, in a high parallelization setting, the vOW variant

can outperform HyperSolver once sufficiently many processors are available: in the worst-case scenario more than $m = 2^{11.59} \approx 3080$ processors at NIST Level I, more than $m = 2^{16.74} \approx 109,000$ processors at NIST Level III, and more than $m = 2^{28.79} \approx 439$ million processors at NIST Level V, since the running time scales like $\sim T_{\mathrm{vOW}}^{\mathrm{stored}}/m$.
Our estimates therefore suggest that parameter growth should be guided by the concrete time–space tradeoff rather than solely by the asymptotic $p^{1/3}$ exponent. Unless one can eliminate the large memory requirement, it may not be necessary to increase $\log_2 p$ from approximately 2λ to 3λ; under realistic memory bounds, an increase of only a few tens of bits, together with a suitable safety margin, may suffice.

	NIST I	NIST III	NIST V
HyperSolver	2^{143}	2^{207}	2^{272}
SuperSolver	$2^{147.18}$	2^{211}	$2^{274.49}$
Full attack ($S \le 2^{80}$)	$2^{[138.78,\,143.51]}$	$2^{[220.1,\,220.58]}$	infeasible
Stored-vOW ($S \le 2^{80}$)	$2^{[151.36,\,154.59]}$	$2^{[222.93,\,223.74]}$	$2^{[297.35,\,301.71]}$

Table 4. Comparison of estimated gate complexities at NIST Levels I, III, and V, using the same M_p gate cost as [PRSU26] for $p = 5 \cdot 2^{248} - 1$, $6 \cdot 2^{376} - 1$, and $27 \cdot 2^{500} - 1$, respectively. HyperSolver values are from [PRSU26, Table 2]; SuperSolver estimates use the class-number estimator of [PRSU26]. For the full and stored-vOW attacks, $S \le 2^{80}$ bits.

6 Quantum Improvements

Quantum claw formulation. For a fixed randomized curve E, let $N = \#\mathcal{I}_{X,B}$ and define

$$F_E(\mathbf{r}) = j(E_{\mathbf{r}}), \qquad G_E(\mathbf{s}) = j(E_{\mathbf{s}})^p.$$

A claw $F_E(\mathbf{r}) = G_E(\mathbf{s})$ yields Frobenius-conjugate codomains and hence an isogeny $E \to E^{(p)}$. Thus the matching phase is a claw-finding problem on two domains of size N. Following Jaques and Schanck [JS19], let E_G, E_D, E_W denote the gate count, depth, and width of a reversible circuit evaluating F_E or G_E, dominated by coherent index decoding and smooth-isogeny evaluation.

Tani's claw-finding algorithm. Tani's algorithm [Tan09] finds a claw in $O(N^{2/3})$ oracle queries, suggesting $p^{2/9+o(1)}$ query complexity since $N = p^{1/3+o(1)}$. However, accounting for its coherent Johnson-graph data structure of size $R \le$

$N^{2/3}$, the Jaques–Schanck model [JS19, Section 5.5] gives

$$G_{\mathrm{Tani}}(R) = O\left(mN\sqrt{R} + E_G\frac{N}{\sqrt{R}}\right),$$
$$D_{\mathrm{Tani}}(R) = O\left((\log m + \log^2 R + E_D)\frac{N}{\sqrt{R}}\right), \quad W_{\mathrm{Tani}}(R) = O(Rm + E_W),$$

where m is the number of bits per stored element. The query-optimal choice $R = N^{2/3}$ gives

$$G_{\mathrm{Tani}} = O(mN^{4/3} + E_G N^{2/3}), \qquad W_{\mathrm{Tani}} = O(mN^{2/3} + E_W).$$

Hence, for $m, E_G = N^{o(1)}$, the gate complexity is $N^{4/3+o(1)} = p^{4/9+o(1)}$, rather than $p^{2/9+o(1)}$. More generally, when $E_G/m \leq N^{2/3}$, the gate-optimal choice $R \approx E_G/m$ yields $G_{\mathrm{Tani}} = O\big(N\sqrt{mE_G}\big)$.

Grover search. Alternatively, define $H(\mathbf{r}, \mathbf{s}) = 1$ precisely when $F_E(\mathbf{r}) = G_E(\mathbf{s})$ and apply Grover search over $\mathcal{I}_{X,B}^2$. Since this space has size N^2, Grover requires $O(N)$ iterations. In the notation above, the Jaques–Schanck [JS19] model gives

$$G_{\mathrm{Grover}} = O(E_G N), \qquad D_{\mathrm{Grover}} = O(E_D N), \qquad W_{\mathrm{Grover}} = O(E_W).$$

Grover uses more oracle evaluations than query-optimal Tani but avoids its large coherent data structure. When $m, E_G, E_D, E_W = N^{o(1)}$, both have $N^{1+o(1)} = p^{1/3+o(1)}$ asymptotic cost, while Grover requires far less coherent memory, consistent with Jaques and Schanck's conclusion that query-optimal Tani need not be cost-optimal in realistic fault-tolerant models.

Remark 4. The rerandomization factor $\Pr[\mathsf{A}_{E,X,B}]^{-1}$ does not appear to admit a direct Grover speedup, since testing whether a randomized curve has the required B-smooth isogeny to its Frobenius conjugate essentially requires solving the underlying problem. We therefore retain the classical factor $\Pr[\mathsf{A}_{E,X,B}]^{-1}$.

7 Implementation and Comparison

We implemented the optimized batched attack, Algorithm 2, and compared its empirical time and space costs with the bounds of Theorem 1. Since the largest prime characteristic considered is only about 44 bits, the fast-multiplication model assumed in Theorem 1 is not representative at these parameters: in practice, field multiplication has essentially constant cost and introduces noticeable fluctuations. We therefore compare field-operation counts, namely T/M_p, where T denotes the time complexity in Theorem 1. Figure 2 and Figure 3 show the resulting time and space comparisons.

Using $\Pr[\mathsf{A}_{E,X,B}] \approx 1 - e^{-\mu_{X,B}}$ from Heuristic 2, with exact $\mu_{X,B}$, the empirical time and space costs match theory within factors of about 1–2.

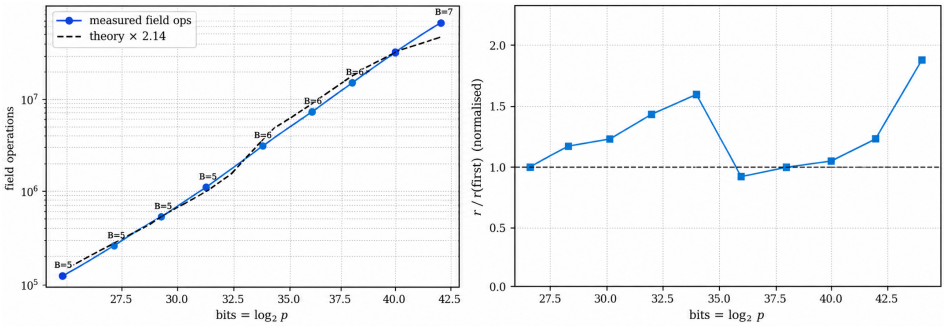

Fig. 2. Growth of theoretical vs actual number of field operations for Algorithm 2 (left) and the estimated hidden constant (right) at different prime sizes.

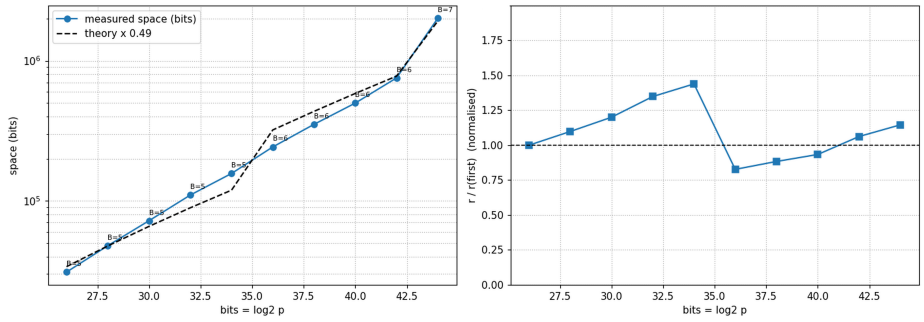

Fig. 3. Growth of theoretical vs actual space complexity for Algorithm 2 (left) and estimated hidden constant (right) at different prime sizes.

We also implemented our algorithm against the original implementation in [Wes26] and SuperSolver of [SCS22]. Figure 4 shows the corresponding performance. The batched Algorithm 2 seems to be 1.7–2× faster than the original attack. Moreover, even though Wesolowski's algorithm has asymptotic growth $p^{1/3}$, compared to the $p^{1/2}$ growth of Delfs–Galbraith, this advantage is masked at small parameters by the polylogarithmic factors in the time-complexity bound of Theorem 1. Consequently, for the parameter range considered here, with primes of at most 40 bits, SuperSolver outperforms our algorithm. We expect, however, that the asymptotic advantage should kick in for substantially larger primes, such as the 256-bit primes used at NIST Level I, where the difference in the exponents should dominate the lower-order logarithmic factors.

We did not compare Algorithm 2 with HyperSolver [PRSU26], as the available implementation is GPU-based. Source code for our algorithms can be found here:

https://github.com/MaherMamah7/One-third-p-Isogeny-Path-attack

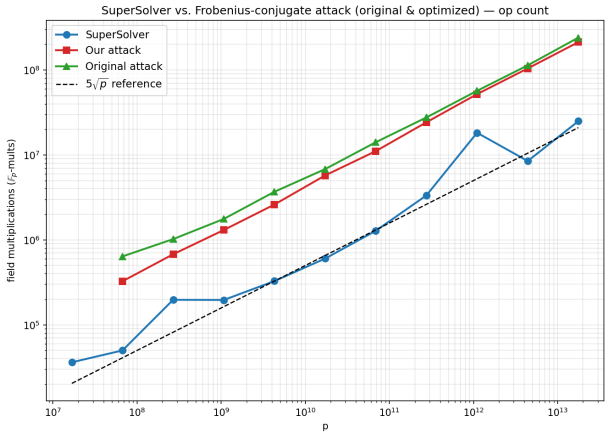

Fig. 4. Operation-count comparison between Algorithm 2 and SuperSolver of [SCS22].

Acknowledgment The author would like to thank Dr. David Jao for helpful editorial comments. The author acknowledges support of the NSERC Alliance Consortia Quantum Grant "Accelerating the transition to quantum-resistant cryptography" (ALLRP 578463–2022), the NSERC Discovery Grant program and William Tutte Post-Graduate scholarship.

References

ABF+25. Marius A. Aardal, Andrea Basso, Luca De Feo, Sikhar Patranabis, and Benjamin Wesolowski. A complete security proof of SQIsign. In *Advances in Cryptology – CRYPTO 2025*, volume 16005 of *Lecture Notes in Computer Science*, pages 190–222. Springer, 2025.

AOV26. Yves Aubry, Roger Oyono, and Christelle Vincent. Minimal degree of an isogeny between a supersingular elliptic curve and its conjugate. *arXiv preprint arXiv:2607.14624*, 2026. arXiv:2607.14624 [math.NT].

BBC+25. Andrea Basso, Giacomo Borin, Wouter Castryck, Maria Corte-Real Santos, Riccardo Invernizzi, Antonin Leroux, Luciano Maino, Frederik Vercauteren, and Benjamin Wesolowski. PRISM: Simple and compact identification and signatures from large prime degree isogenies. In *Public-Key Cryptography – PKC 2025*, pages 300–332. Springer, 2025.

BBD+24. Jeremy Booher, Ross Bowden, Javad Doliskani, Tako Boris Fouotsa, Steven D. Galbraith, Sabrina Kunzweiler, Simon-Philipp Merz, Christophe Petit, Benjamin Smith, Katherine E. Stange, Yan Bo Ti, Christelle Vincent, José Felipe Voloch, Charlotte Weitkämper, and Lukas Zobernig. Failing to hash into supersingular isogeny graphs. *The Computer Journal*, 67(8):2702–2719, 2024.

BCC+23. Andrea Basso, Giulio Codogni, Deirdre Connolly, Luca De Feo, Tako Boris Fouotsa, Guido Maria Lido, Travis Morrison, Lorenz Panny, Sikhar Patranabis, and Benjamin Wesolowski. Supersingular curves you can trust. In *Advances in Cryptology – EUROCRYPT 2023*, pages 405–437. Springer, 2023.

BKV19. Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. Cryptology ePrint Archive, Paper 2019/498, 2019.

BLS12. Reinier Bröker, Kristin Lauter, and Andrew V. Sutherland. Modular polynomials via isogeny volcanoes. *Mathematics of Computation*, 81(278):1201–1231, 2012.

CGL09. Denis X. Charles, Eyal Z. Goren, and Kristin E. Lauter. Cryptographic hash functions from expander graphs. *Journal of Cryptology*, 22(1):93–113, 2009.

CLM+18. Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An efficient post-quantum commutative group action. In *Advances in Cryptology – ASIACRYPT 2018*, pages 395–427. Springer, 2018.

CLN+20. Craig Costello, Patrick Longa, Michael Naehrig, Joost Renes, and Fernando Virdia. Improved classical cryptanalysis of SIKE in practice. In *Public-Key Cryptography – PKC 2020*, volume 12111 of *Lecture Notes in Computer Science*, pages 505–534. Springer, 2020.

DG16. Christina Delfs and Steven D. Galbraith. Computing isogenies between supersingular elliptic curves over $\mathbb{F}_p$. *Designs, Codes and Cryptography*, 78(2):425–440, 2016.

EHL+18. Kirsten Eisenträger, Sean Hallgren, Kristin E. Lauter, Travis Morrison, and Christophe Petit. Supersingular isogeny graphs and endomorphism rings: reductions and solutions. In *EUROCRYPT 2018*, 2018.

EHL+20. Kirsten Eisenträger, Sean Hallgren, Chris Leonardi, Travis Morrison, and Jennifer Park. Computing endomorphism rings of supersingular elliptic curves and connections to path-finding in isogeny graphs. In *Algorithmic Number Theory Symposium (ANTS XIV)*, volume 4 of *The Open Book Series*, pages 215–232. Mathematical Sciences Publishers, 2020.

EPSV24. Jonathan Komada Eriksen, Lorenz Panny, Jana Sotáková, and Mattia Veroni. Deuring for the people: Supersingular elliptic curves with prescribed endomorphism ring in general characteristic. In *LuCaNT: LMFDB, Computation, and Number Theory*, volume 796 of *Contemporary Mathematics*, pages 339–373. American Mathematical Society, 2024.

FKL+20. Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. SQISign: Compact post-quantum signatures from quaternions and isogenies. In *Advances in Cryptology – ASIACRYPT 2020*, pages 64–93. Springer, 2020.

GS11. Steven D. Galbraith and Anton Stolbunov. Improved algorithm for the isogeny problem for ordinary elliptic curves. *arXiv preprint arXiv:1105.6331*, 2011. arXiv:1105.6331 [math.NT].

HT86. Adolf Hildebrand and Gérald Tenenbaum. On integers free of large prime factors. *Transactions of the American Mathematical Society*, 296(1):265–290, 1986.

JS19. Samuel Jaques and John M. Schanck. Quantum cryptanalysis in the RAM model: Claw-finding attacks on SIKE. In *Advances in Cryptology – CRYPTO 2019*, volume 11692 of *Lecture Notes in Computer Science*, pages 32–61. Springer, 2019.

Koh96. David Kohel. *Endomorphism rings of elliptic curves over finite fields*. PhD thesis, University of California, Berkeley, 1996.

Meh17. Marzieh Mehdizadeh. The multiplication table for smooth integers, 2017.

PRSU26. Lorenz Panny, Ryan Rueger, Alessandro Sferlazza, and Aleksei Udovenko. Hypersolver: Asymptotically and concretely accelerating the delfs–galbraith attack using isogeny ladders. In *Advances in Cryptology – ASIACRYPT 2026*, 2026. To appear.

PW24. Aurel Page and Benjamin Wesolowski. The supersingular endomorphism ring and one endomorphism problems are equivalent. In *Advances in Cryptology – EUROCRYPT 2024*, pages 388–417. Springer, 2024.

SCS22. Maria Corte-Real Santos, Craig Costello, and Jia Shi. Accelerating the delfs–galbraith algorithm with fast subfield root detection. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022*, pages 285–314, Cham, 2022. Springer Nature Switzerland.

Tan09. Seiichiro Tani. Claw finding algorithms using quantum walk. *Theoretical Computer Science*, 410(50):5285–5297, 2009.

vOW99. Paul C. van Oorschot and Michael J. Wiener. Parallel collision search with cryptanalytic applications. *Journal of Cryptology*, 12(1):1–28, 1999.

vzGG13. Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 3 edition, 2013.

Wes22. Benjamin Wesolowski. The supersingular isogeny path and endomorphism ring problems are equivalent. In *FOCS 2022*, pages 1100–1111. IEEE, 2022.

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

A Estimates for the Number of Rerandomizations

$\log_2 p$	B	X	u^{-u}	P_{heur}	P_{emp}
28	5	51	0.0052	0.376	0.465
28	7	60	0.0239	0.687	0.770
28	11	76	0.0831	0.921	0.965
28	13	82	0.1151	0.986	0.995
28	17	94	0.1759	0.999	1.000
32	5	81	0.0013	0.226	0.280
32	7	96	0.0083	0.442	0.480
32	11	120	0.0381	0.749	0.827
32	13	130	0.0568	0.925	0.960
32	17	149	0.0959	0.986	0.993
36	5	128	0.0003	0.113	0.111
36	7	151	0.0027	0.286	0.333
36	11	190	0.0166	0.538	0.600
36	13	206	0.0268	0.755	0.822
36	17	236	0.0501	0.910	0.956
40	5	203	0.0001	0.051	0.067
40	7	240	0.0008	0.144	0.200
40	11	301	0.0069	0.351	0.467
40	13	327	0.0121	0.542	0.683
40	17	374	0.0252	0.766	0.850
44	5	322	0.0000	0.025	0.050
44	7	381	0.0002	0.079	0.175
44	11	477	0.0028	0.188	0.325
44	13	519	0.0053	0.346	0.475
48	5	510	0.0000	0.012	0.040
48	7	604	0.0001	0.039	0.080
48	11	757	0.0011	0.110	0.160
48	13	823	0.0022	0.201	0.240

Table 5. Comparison of the classical estimate u^{-u}, the refined heuristic probability $P_{\text{heur}} = 1 - \exp(-\mu_{B,X})$, and the empirical success probability P_{emp} for $X = B^{1/2}(p/2)^{1/6}$.

B Analysis of Algorithm 1

We analyzed the running time of Algorithm 1 and report the results in Figure 5. The data indicate that SageMath uses PARI's implementation for computing the modular polynomials Φ_ℓ, which exhibits the same asymptotic growth as the BLS algorithm [BLS12]. Moreover, with N denoting the list size, the top-right plot verifies that the second and third terms in the running-time bound of Lemma 2 scale essentially linearly with N, up to lower-order logarithmic factors that are effectively constant at these small parameters. Finally, for the third

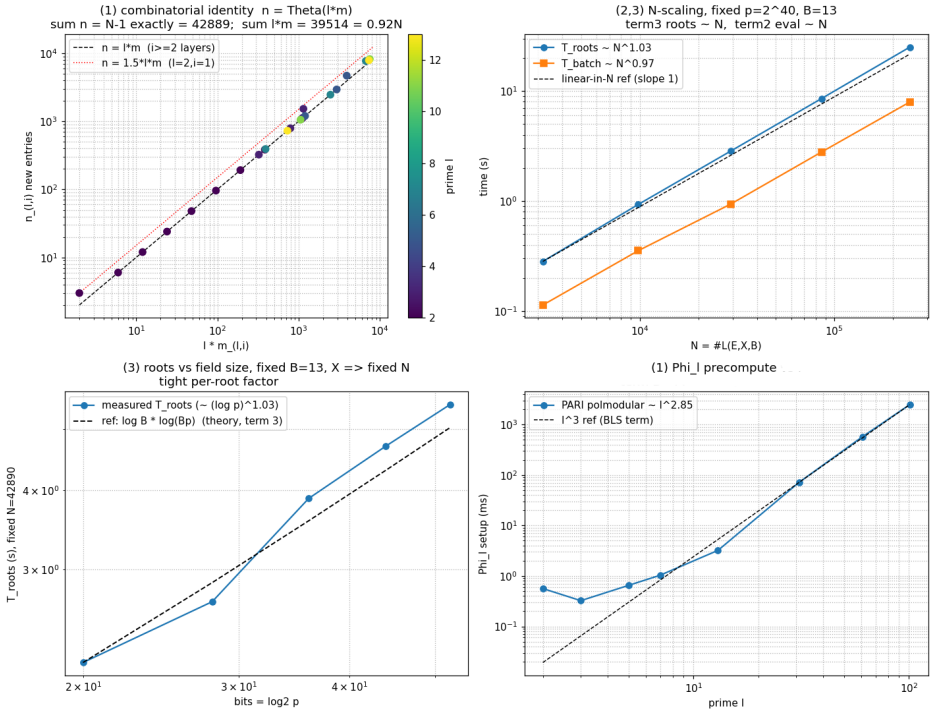

Fig. 5. Analysis of running time of each component of Algorithm 1.

term, corresponding to root extraction, the $\log B \log(Bp)$ dependence appears to be tight, as shown in the bottom-left plot.

C Estimates in Table 4

We briefly explain how the estimates in Table 4 were obtained. The time complexities in Theorem 1 and Theorem 2 are expressed in bit operations, since they inherit the arithmetic costs of the BLS algorithm [BLS12].

By contrast, Panny et al. [PRSU26] estimate the number of $\mathbb{F}_p$-multiplications and charge a cost of $2^{15.88}$ gates per multiplication. To place our estimates on the same scale, we normalize the bit-operation complexity T by the bit complexity M_p of one $\mathbb{F}_p$-multiplication,

$$\mathsf{M}_p \lesssim \log p \log\log p$$

thereby obtaining an $\mathbb{F}_p$-multiplication-equivalent operation count. This should be interpreted as a normalization for comparison rather than as an exact count of field multiplications, since the attack involves heterogeneous arithmetic subroutines.

This normalization is also supported empirically by Figure 2, where the theoretical field-operation estimates closely track the measured operation counts.

We then apply the same cost of $2^{15.88}$ gates per field multiplication as in [PRSU26]. Thus, the gate complexity reported for our attacks is estimated as

$$\frac{T}{\mathsf{M}_p} \times 2^{15.88}.$$