Concrete Security Assessment of Isogeny-based Cryptography with the new Isogeny-Path algorithm

Maher Mamah

Department of Combinatorics & Optimization, University of Waterloo, ON, Canada

Abstract. Very recently, Wesolowski (*ePrint* 2026/1486) proposed a 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 constitutes an asymptotic improvement over the previous best-known complexity of $p^{1/2}\log^{O(1)}(p)$, its concrete impact on the security of isogeny-based cryptographic schemes, particularly SQIsign, remains unclear due to the superpolynomial overhead hidden in the $p^{o(1)}$ factor and the algorithm's exponential memory requirement. In this work, we assess the concrete 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, neither the optimized full-list attack nor its vOW variants outperform the previous state-of-the-art low-memory algorithm for computing supersingular endomorphism rings. We further study quantum claw-finding improvements. While Grover search can essentially remove 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. Overall, our results show that the asymptotic $p^{1/3+o(1)}$ improvement does not directly translate into a comparable reduction in concrete security.

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 [9] and SQIsign [15], 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 [27,12]. Since EndRing has been studied since Kohel's foundational work [20], recent cryptanalytic advances have increasingly focused on improving algorithms for EndRing. A particularly

relevant low-memory algorithm is that of Eisenträger et al. [13], 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 [23] 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 [1, Section 6], using a OneEnd solver for EndRing incurs a multiplicative running-time overhead of roughly $2^{94}\log^{13}(p)$.

Delfs and Galbraith [11] 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 [16], building on the GHS approach, develop such a low-storage method; their practical parameterization stores $O(n^{1/4})$ distinguished points, where n is the size of the underlying group-action space, while retaining square-root collision-search asymptotics.

Very recently, Wesolowski [28] 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 [13], 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 $(p/2)^{1/3}$ [2]. 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 [5] and CSIDH [8], 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 both time and memory constraints and provide concrete estimates for 256- and 512-bit characteristics.
– We study the classical time–memory tradeoff in detail. In particular, we instantiate the van Oorschot–Wiener collision search [22] without explicitly storing the smooth-isogeny list, introduce a deterministic indexing of smooth isogenies, and analyze two implementations depending on whether modular polynomials are stored or recomputed on demand. Our concrete estimates identify the regimes in which vOW improves over the full-list attack and those in which the latter becomes preferable.
– We compare these optimized attacks with the previous low-memory $O(p^{1/2}\log^2 p)$-time EndRing algorithm of Eisenträger et al. [13]. Over the practical memory ranges considered, neither the full-list nor the vOW variants outperform this benchmark. In particular, for 256- and 512-bit characteristics, the resulting classical attack costs remain above the nominal 2^{128} and 2^{256} security targets, respectively.
– Finally, we investigate quantum claw-finding improvements. Adapting the cost model of Jaques and Schanck [18], we analyze both Grover search and Tani's quantum walk in terms of gates, depth, and coherent quantum memory. We show that Tani's apparent $O(N^{2/3})$ advantage is only a query-complexity statement: once the quantum data structure is accounted for, it becomes a substantial time–memory tradeoff. Grover, on the other hand, essentially removes the large search-memory requirement but does not significantly improve the running-time dependence of the attack.

Since our objective is to assess the concrete impact of Wesolowski's EndRing algorithm, we focus our numerical comparison on algorithms that directly target EndRing and are closest in spirit to his approach, most notably the low-memory algorithm of Eisenträger et al. [13]. General isogeny-path algorithms such as Delfs–Galbraith also yield $p^{1/2+o(1)}$-time attacks and are relevant through the known reductions between Isogeny-path and EndRing problems. We do not include them in the concrete tables, since obtaining a directly comparable EndRing cost requires a separate concrete analysis, which we leave for future work.

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.

2 Wesolowski's Algorithm

In what follows we assume familiarity with Wesolowski's setup [28] and refer the reader to section 2 therein for the relevant background.
Wesolowski's algorithm [28] 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 [2] 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 $\phi = \eta \circ \psi$ [28, 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 [4, 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 [13]. Under certain heuristics, this approach is more efficient than the rigorous approach outlined in [23].

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 non-trivial 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

Our main goal in this section is to argue that the $o(1)$-term in Equation (2) is essentially negligible at cryptographic parameters relevant to our applications. Let ρ denote the Dickman–de Bruijn function. For asymptotically large u, the Dickman function satisfies

$$\rho(u) = u^{-u(1+o(1))}$$

A more refined asymptotic expansion [7] is

$$\rho(u) = \exp\left(-u\left(\log u + \log\log u - 1 + \frac{\log\log u - 1}{\log u} + O\left(\frac{(\log\log u)^2}{(\log u)^2}\right)\right)\right). \tag{3}$$

For integers of size approximately $(p/2)^{1/3}$, define $u = \frac{\log(p/2)}{3\log B}$. Under the random-integer heuristic introduced by Wesolowski [28],

$$\Pr[D_E \text{ is } B\text{-smooth}] \approx \rho(u).$$

Although the second-order term gives a sharper asymptotic description as $u \to \infty$, it does not necessarily yield a more accurate numerical approximation at the moderate values of u arising from cryptographic parameters.

Wesolowski [28] tested this heuristic experimentally by sampling random maximal orders in $B_{p,\infty}$ and recording the largest prime factor of the corresponding minimum isogeny degree. For the SQIsign-I prime $p = 5 \cdot 2^{248} - 1$, the empirical distribution obtained from 100 000 samples closely follows the Dickman prediction. We observed the same behavior in 20 000 additional samples present in Figure 1, covering approximately the range

$$1.01 \leq u \leq 4.58.$$

Over this interval, direct numerical evaluation gives

$$1 \lesssim \frac{\rho(u)}{u^{-u}} \lesssim 1.313.$$

Thus, replacing $\rho(u)$ by u^{-u} underestimates the smoothness probability and

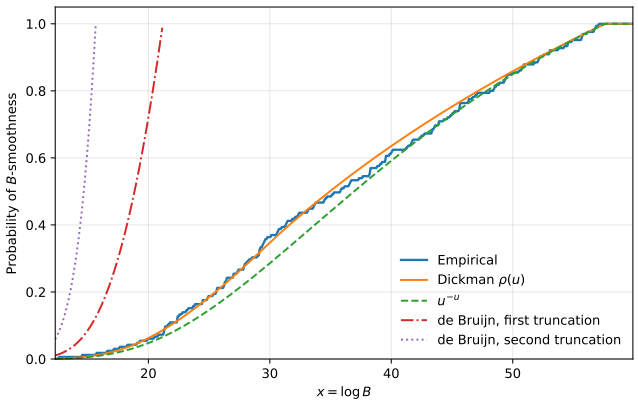

Fig. 1. Cumulative distribution of the logarithm of the largest prime factor dividing the degree of the smallest isogeny $E \to E^{(p)}$ over 20,000 samples compared to the theoretical prediction u^{-u} along with first and second order de Bruijn expressions

Statistic	D_E	$D_E/(p/2)^{1/3}$	$\log_2 D_E$
Minimum	$2.38 \cdot 10^{23}$	0.0228	77.6
Average	$3.95 \cdot 10^{24}$	0.379	81.71
Median	$3.89 \cdot 10^{24}$	0.374	81.69
Maximum	$8.71 \cdot 10^{24}$	0.836	82.85

Table 1. Summary statistics for the observed minimum isogeny degrees D_E for the 256-SQIsign-I prime where $\log_2((p/2)^{1/3}) \approx 83.11$

therefore conservatively overestimates the expected search cost by at most a factor of 1.313 over this range.

The discrepancy with truncated de Bruijn approximations is most visible in the smooth tail. At $u \approx 4.577$, corresponding to the smallest smoothness bound observed in our experiment, we obtain

$$\rho(u) \approx 1.117 \cdot 10^{-3}, \qquad u^{-u} \approx 9.469 \cdot 10^{-4},$$
$$\rho_{\mathrm{dB},1}(u) \approx 1.350 \cdot 10^{-2}, \quad \rho_{\mathrm{dB},2}(u) \approx 7.747 \cdot 10^{-2},$$

where $\rho_{\mathrm{dB},1}$ and $\rho_{\mathrm{dB},2}$ denote the first- and second-order truncated de Bruijn expressions. Hence, u^{-u} differs from the numerical Dickman value by only a factor of approximately 1.18, whereas the second-order truncation differs by a factor of approximately 69.

This discrepancy does not contradict the heuristic that the minimum degree of an isogeny from E to $E^{(p)}$ has the smoothness behavior of a random integer of comparable size. Rather, it shows that the values of u arising at the SQIsign parameters are too small for finite truncations of the asymptotic de Bruijn expansion Equation (3) to provide accurate concrete estimates. Consequently, the second-order truncation is not numerically reliable over the parameter range considered here. Indeed, the largest value in our experiment is $u \approx 4.577$, while the smoothest sample in Wesolowski's 100 000-sample experiment corresponds only to $u \approx 6.10$. Even for a 512-bit prime and the extreme choice $B = 2$, one has only $u \approx 170$. Thus, asymptotic expansions alone should not be used for concrete estimates without numerical validation.

We therefore adopt the following heuristic throughout the paper.

Heuristic 1. *For cryptographic parameter ranges, with p* 256- *to* 512*-bit prime and* $u = \frac{\log(p/2)}{3 \log B}$*, we estimate*

$$\Pr[D_E \textit{ is } B\textit{-smooth}] \approx \rho(u) \approx u^{-u}.$$

We note that, across all 20,000 executions, our numerical experiments consistently found the smallest-degree smooth isogeny to the Frobenius conjugate to attain values near the upper-bound scale $(p/2)^{1/3}$ as noted in Table 1, suggesting that this bound may in fact be near-optimal.
A source code for generating this data can be found here:

`https://github.com/MaherMamah7/Isogeny-to-Frobenius-Curve`

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 [4, 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 $\rho(u)^{-1} \approx u^u$, where $u = \frac{\log(p/2)}{3 \log B}$. It therefore remains to analyze the cost of enumerating the required B-smooth isogenies, corresponding to Algorithm 1 of [28] and then combine the results. The main results of this section are Theorem 1 and Corollary 1.

4.1 Listing B-smooth Isogenies

Following the suggestion in [28, 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 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 1 (GRH). *Algorithm 1 is correct and runs in expected time*

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

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.

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** $\mathcal{L}$

Proof. Algorithm 1 is essentially [28, Algorithm 1], with two modifications. First, for every prime $\ell \leq B$, we compute $\Phi_\ell(U, V) \bmod p$ 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 [6, Theorem 1], the modular polynomial $\Phi_\ell(U, V) \bmod p$, for a fixed prime ℓ, can be computed in time

$$O\big(\ell^3 + \ell^2 \log p\big),$$

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

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

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 [17, Corollary 10.8] computes

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

in $O\left(\ell\, \mathsf{M}(m_{\ell,i} + \ell) \log(m_{\ell,i} + \ell)\right)$ operations in $\mathbb{F}_{p^2}$.

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

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

operations in $\mathbb{F}_{p^2}$, where we used

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

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 [17, 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

$$\begin{aligned} T_{\text{roots}} &= O\left(\sum_{\ell,i} m_{\ell,i} \mathsf{M}(\ell) \log \ell \log(\ell p)\right) \\ &\le O\left(\frac{\mathsf{M}(B)}{B} \log B \log(Bp) \sum_{\ell,i} \ell m_{\ell,i}\right) \\ &= O\left(N \frac{\mathsf{M}(B)}{B} \log B \log(Bp)\right) \\ &= O\left(B^{-1} N \mathsf{M}(B) \log^2 p)\right) \end{aligned}$$

operations in $\mathbb{F}_{p^2}$.

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 [6, Theorem 1]. Thus the size of the local memory needed is $O(B^2 \log p)$.

Lemma 2. *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

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

Algorithm 2 Finding a smooth isogeny to the Frobenius conjugate

Require: A supersingular elliptic curve $E/\mathbb{F}_{p^2}$ and a smoothness bound $2 \leq B < p$.
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 $\perp$

bits. Finally, since

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

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 3 ([28, Lemma 3.4]). *Suppose that the smallest-degree isogeny $E \to E^{(p)}$ is B-smooth. Then Algorithm 1 returns a separable isogeny $\varphi : E \to E^{(p)}$.*

Next, we combine the numerical results from Section 3 with Lemma 1 to obtain an explicit running-time bound for Algorithm 2. We first recall the standard number-theoretic notation for smooth integers.

Definition 1. *For $X, B \geq 1$, let $\Psi(X, B) := \#\{d \leq X : P^+(d) \leq B\}$, where $P^+(d)$ denotes the largest prime divisor of d, with $P^+(1) = 1$.*

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

$$T = O\Bigg(u^u (B^4 + B^3 \log p) + p^{1/3} B (\log B)^{1/2} (\log p)^{5/2} \big(\log\log p + \log B \log\log B\big) \cdot \exp\left(\frac{\log(p/2)}{6 \log B} \log\left(\frac{2 \log(p/2)}{3 \log B} \right) \right) \Bigg).$$

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 1, one trial succeeds with probability approximately u^{-u}, and hence the expected number of trials is u^u. By Lemma 1, the listing cost of one trial is

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

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, [28, 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 $\Psi(X, B) \approx X\rho(v) \approx Xv^{-v}$, we further obtain $N = O(X^2 v^{-v} \log X)$. Thus, after accounting for the expected u^u trials, the preprocessing contributes $O(u^u(B^4 + B^3 \log p))$, while batching and root extraction together contribute

$$O(u^u X^2 v^{-v} \log X \ \log^2 p\,(\log\log p + \mathsf{M}(B)/B))$$

.

Since $X^2 = B \cdot (p/2)^{1/3}$ and $v = (u+1)/2$, we obtain

$$\frac{u^u}{v^v} = O\left(\sqrt{\frac{\log B}{\log p}}\,\exp\left(\frac{\log(p/2)}{6\log B}\log\left(\frac{2\log(p/2)}{3\log B}\right)\right)\right).$$

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 1, Lemma 2, and Equation (4) we have

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

Now

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

Using $X^2 = B(p/2)^{1/3}$ and $\log X = \Theta(\log p)$ 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 1 (GRH). *Under the same hypothesis of Theorem 1 and letting* $B = \exp(\frac{1}{3}\sqrt{\log p})$*, Algorithm 2 runs in time*

$$T = O\left(p^{1/3}\log^{13/4}(p)\log\log p\cdot\exp\left(\frac{1}{4}\sqrt{\log p}\log\log p + \left(\frac{1}{3}+\frac{\ln 2}{2}\right)\sqrt{\log p}\right)\right)$$

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. Table 2 reports the resulting time–space tradeoff for 256- and 512-bit primes p.

u	B	$\log_2 T$	$\log_2 S$
8	1 579	136.39	100.61
7	4 522	135.55	103.89
6.269	**12 076**	**135.34**	**106.55**
6	18 390	135.37	107.60
5.5	44 904	135.62	109.69

u	B	$\log_2 T$	$\log_2 S$
10	134 136	237.40	190.58
9	498 048	236.70	194.39
8.326	**1 440 395**	**236.55**	**197.18**
8	2 567 026	236.59	198.61
7.5	6 866 346	236.82	200.92

Table 2. Time–space tradeoff for B, for $\log_2 p = 256$ (left) and 512 (right). Bold rows minimize the running time.

The data show that, even after optimizing B for running time, Wesolowski's estimates substantially underestimate the concrete cost captured by Theorem 1. In particular, under his optimistic cost model and an optimized choice of B, Wesolowski estimates $T_{256} \geq 2^{106.5}$ and $S_{256} \geq 2^{92.5}$ for the 256-bit case and $T_{512} \geq 2^{204.2}$ and $S_{512} = 2^{181.3}$ for the 512-bit case. In contrast, our refined analysis gives approximately $T_{256} = 2^{135.34}$ and $S_{256} = 2^{106.55}$ and $T_{512} = 2^{236.55}$ with $S_{512} = 2^{197.18}$. More generally, the results in table 2 indicate that the attack remains far from practical at the parameter sizes relevant to NIST security levels

	$\log_2 p = 256$		$\log_2 p = 512$	
Space bound	B	$\log_2 T$	B	$\log_2 T$
2^{40}	6.92	200.23	7.19	404.59
2^{50}	8.98	187.65	8.01	391.90
2^{60}	12.75	175.31	9.05	379.24
2^{70}	20.93	163.40	10.40	366.59
2^{80}	44.36	152.31	12.20	353.99

Table 3. Memory-constrained time–space tradeoff for $\log_2 p = 256$ and $\log_2 p = 512$.

I and V, primarily because of its prohibitive memory requirements. We stress, however, that for the 512-bit case the optimized running-time estimate is below 2^{256}, so the distinction between nominal bit security and practical feasibility is important.

Optimizing for space In practice, adversaries are often more constrained by memory than by running time. We therefore analyze the running time of the attack under different memory budgets. In particular, we consider memory-constrained adversaries with between 2^{40} and 2^{80} bits of storage [1]. For each memory budget, we optimize B subject to the corresponding space constraint; Table 3 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 approximately

$$T_{256} = 2^{152.31} \qquad \text{and} \qquad T_{512} = 2^{353.99}$$

operations for 256- and 512-bit primes, respectively.

Furthermore, we note that a smaller value of $X = B^{1/2}(p/2)^{1/6}$ would yield an exponentially faster algorithm, provided we are fortunate enough that the desired isogeny $\varphi : E \to E^{(p)}$ from the curve to its Frobenius map is significantly smaller than $(p/2)^{1/3}$. However, based on the data in Section 3, the degree $\deg(\varphi)$ appears to essentially attain $\sim (p/2)^{1/3}$. Determining whether this is always the case remains an interesting direction for future research.

5 The van Oorschot–Wiener technique

The analysis above considers a meet-in-the-middle procedure that explicitly constructs and stores the entire set of smooth isogenies $\mathcal{L}(E, X, B)$. This yields the lowest running time among the variants considered so far, but requires memory essentially proportional to the size of the search space. Wesolowski observes that

[1] For scale, in 2025 Google reported that many Colossus filesystems have multiple exabytes of storage, including two exceeding 10 EB each, for a combined capacity of more than 20 EB, or approximately $2^{67.1}$ bits.

the matching step can instead be viewed as a claw-finding problem and proposes the classical van Oorschot–Wiener (vOW) time–memory tradeoff [28,22]. This technique has previously been studied in considerable detail in the context of supersingular-isogeny cryptanalysis by Costello et al. [10]. Our analysis shows that vOW provides a meaningful advantage in severely memory-constrained regimes, but does not improve the running time once sufficient memory is available, at which point Algorithm 2 becomes preferable. The main results in this section is Theorem 2 and Theorem 3 and their concrete security estimates.

The generic vOW algorithm. Let $\mathcal{S}$ be a finite search space of size N, and let $h : \mathcal{S} \to \mathcal{Y}$ be an expensive function for which a particular collision solves the underlying problem. More precisely, a pair of distinct elements $x, y \in \mathcal{S}$ satisfying $h(x) = h(y)$ is called a collision of h. In the applications considered here, not every such collision is necessarily useful; a collision satisfying the additional problem-specific condition is referred to as a *golden collision*. The objective of the vOW algorithm is to find this golden collision without storing all N values of h.

For each function version ν, vOW composes h with a pseudorandom map $g_\nu : \mathcal{Y} \to \mathcal{S}$, obtaining the iteration function $f_\nu = g_\nu \circ h$. Starting from a uniformly sampled $x_0 \in \mathcal{S}$, one generates a trail $x_{i+1} = f_\nu(x_i)$. A proportion θ of the elements of $\mathcal{S}$ are designated as *distinguished points*, according to an efficiently testable predicate. Distinguishedness has no relation to whether a point participates in the golden collision; it is merely a device for compressing the memory required to detect merging trails.

A trail is continued until a distinguished point x_d is reached, at which point only the triple (x_0, x_d, d), consisting of its starting point, distinguished endpoint, and length, is stored. If a later trail terminates at the same distinguished point, the two trails must have merged somewhere before reaching that endpoint. They are therefore retraced in order to locate distinct predecessors $x \neq y$ such that

$$f_\nu(x) = f_\nu(y).$$

Such a collision of f_ν is not automatically a collision of h: it may have been introduced solely by the pseudorandom compression g_ν. Consequently, after locating x and y, one explicitly evaluates $h(x)$ and $h(y)$. If $h(x) \neq h(y)$, the collision is discarded. If $h(x) = h(y)$, one has found an actual collision of h, which is then checked against the problem-specific condition defining the golden collision.

Since the probability of detecting a fixed golden collision may be poor for a particular random function f_ν, the function is periodically changed. Following the heuristic optimization of [10], one runs a function version until approximately βw distinguished points have been produced, where w is the number of distinguished-point records that can be stored. Their experiments give $\beta = 10$, so a function version is changed after approximately $10w$ distinguished points have been generated.

van Oorschot and Wiener [22] claim that a precise closed formula for the runtime is difficult to achieve and rely on experimental data to determine several

constants. However, generally speaking, if w distinguished-point records can be stored, the generic analysis gives an expected number of step-function evaluations of order

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

where κ_{vOW} represents the constant overhead of the distinguished-point method. Following the practical analysis of [10], one may take $\kappa_{\mathrm{vOW}} \approx 2.5$ as an optimistic concrete estimate. The value should not be interpreted as an exact constant: the runtime depends on many factors: the distinguished-point density, collision checking, function changes, and the finite-size behavior of the random walks. In particular, [10] uses a proportion $\theta = \alpha\sqrt{w/N}$, with the experimentally optimized value $\alpha \approx 2.25$, and changes the function version after approximately $10w$ distinguished points have been produced.

The advantage of vOW is therefore entirely a time–memory tradeoff. Reducing the number of stored elements from N to w increases the number of function evaluations by a factor essentially $\sqrt{N/w}$, but only w distinguished-point records need to be retained. The algorithm is also perfectly parallelizable, since the distinguished-point trails can be generated independently.

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

$$\mathcal{I}_{X,B} := \bigsqcup_{d \in \mathcal{S}(X,B)} \left(\{d\} \times \prod_{\ell_i^{e_i} \| d} \{0, \ldots, R_i - 1\} \right), \quad N = \#\mathcal{I}_{X,B} = \sum_{d \in \mathcal{S}(X,B)} \psi_{\mathrm{D}}(d). \tag{5}$$

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. Under Heuristic 1, a randomized curve admits the required B-smooth isogeny with probability approximately u^{-u}, where $u = \log(p/2)/(3 \log B)$. 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. The expected number of randomized instances is consequently u^u, up to a constant that we absorb into κ_{vOW}.

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, and $\mathsf{mode} \in \{\mathsf{stored}, \mathsf{direct}\}$.

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 (5)
if mode = stored **then**
    Precompute and store $\Phi_\ell(X, Y) \bmod p$ for all primes $\ell \le B$
end if
while true **do**
    Sample walk $\omega : E \to E_1$; define $h(\mathbf{r}) = \mathrm{canonical}(j(\mathrm{cod}(\mathsf{Decode}_{E_1,X,B}(\mathbf{r}))))$
    $\theta \leftarrow \alpha\sqrt{w/N}$; $Q \leftarrow \lceil \kappa_{\mathrm{vOW}}\sqrt{N^3/w} \rceil$; $q \leftarrow 0$, $\nu \leftarrow 0$
    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

Cost of the step function. The generic vOW complexity counts evaluations of $f_{\nu,E}$, but in our setting such an evaluation is substantially more expensive than a hash evaluation. The dominant operation is the evaluation of $h_E(r)$, which requires decoding r and computing the endpoint of the corresponding smooth isogeny. If the indexed degree is $d = \prod_i \ell_i$, the endpoint is obtained sequentially as

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

where Ω is the prime-divisor counting function. At the i-th step we must compute the roots of $\Phi_{\ell_i}(j(E_{i-1}), Y)$, remove the backtracking edge when necessary, and select the branch specified by the index.

This is precisely where the vOW implementation differs from the optimized batched enumeration underlying Theorem 1. In the latter, Algorithm 2 processes many curves requiring the same ℓ-extension simultaneously, allowing the corresponding specializations of Φ_ℓ to be evaluated in batches, with all non-backtracking roots contributing useful entries. Neither property is automatically available in a vOW walk. The next state $r' = g_\nu(h_E(r))$ is unknown until the current smooth isogeny has been fully evaluated, making each trail inherently adaptive. Moreover, an indexed state selects only one branch of each modular-polynomial specialization.

Independent vOW trails may still be synchronized and batched by ℓ, but the resulting amortization depends on the distribution of the indexed smooth paths and the available parallelism. We therefore do not assume this additional batching in the bounds below; any such optimization can only improve the stated estimates.

Accordingly, Algorithm 3 considers two computational modes: precomputing and storing all Φ_ℓ when sufficient memory is available, or computing each specialization $\Phi_\ell(j(E_r), Y)$ directly in memory-constrained settings.

Precomputed modular polynomials. The first approach keeps the modular polynomial preprocessing used by our optimized implementation of Wesolowski's algorithm. For every prime $\ell \leq B$, we compute $\Phi_\ell(X, Y) \bmod p$ using the BLS algorithm [6] once and retain it throughout the attack. This preprocessing is global: the polynomial depends only on p and ℓ, and hence can be reused across all vOW trails, function versions, and rerandomized curves.

Writing $\Phi_\ell(X, Y) = \sum_{k=0}^{\ell+1} c_{\ell,k}(X)Y^k$, a specialization at $j \in \mathbb{F}_{p^2}$ may be obtained by evaluating the $\ell+2$ coefficient polynomials $c_{\ell,k}$ at j. Without assuming cross-trail batching this requires $O(\ell^2)$ field operations. Using fast randomized finite-field factorization [17, Corollary 14.16], the expected cost of finding the required roots of the resulting degree-$(\ell+1)$ polynomial is $O\big(\mathsf{M}(\ell)\log\ell\log\ell p\big)$. Thus, the cost of specializing and root-extraction for one step is

$$C_{\text{store}}(B, p) := O(B^2 + \mathsf{M}(B)\log B \log p) \tag{6}$$

Since an indexed smooth isogeny contains at most $\log_2 X$ prime-degree steps, one vOW function evaluation costs at most $t_{\text{store}} = O(C_{\text{store}}(B, p)\log X)$.

The price is the storage of the modular-polynomial database. Since $\Phi_\ell \bmod p$ contains $O(\ell^2)$ coefficients, its total storage for all primes $\ell \leq B$ is $S_\Phi = O(B^3 \log p/\log B)$ bits. This cost is negligible compared with the original full smooth-isogeny table in the asymptotic regime considered above, but it becomes relevant when vOW is used with a very small collision table. For instance if $B = 17400$ and $\log_2 p = 256$, then $S_\Phi \approx 2^{46.4}$ bits or equivalently 12 TB. For higher values of B and p this storage requirement might become an obstacle for low-memory attacks and thus one can revert to Direct computation mode of algorithm 3.

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 1, 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 $\Phi_\ell \bmod p$, for primes $\ell \le B$, are precomputed and stored using BLS algorithm, Algorithm 3 has expected running time

$$T_{\mathrm{vOW}}^{\mathrm{store}} = O\Bigg(B^4 + B^3 \log p + \frac{p^{1/2}}{\sqrt{w}} (\log p)^{5/2} \Big[B^{7/2} + B^{5/2} (\log B)^2 \log\log B \log p \Big] \cdot \exp\left(u \log u - \frac{3(u+1)}{4} \log\left(\frac{u+1}{2} \right) \right) \Bigg).$$

where $u = \dfrac{\log(p/2)}{(3 \log B)}$. *Its space requirement is* $O\big(w \log N + B^3 \log p \log^{-1} B\big)$ *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)

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

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 [22] 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

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

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

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

Under Heuristic 1, a rerandomized curve admits the required B-smooth isogeny with probability approximately u^{-u}. Consequently, the vOW search must be repeated $O(u^u)$ times in expectation. Combining the above estimates, and using

$$X^3 = B^{3/2} \left(\frac{p}{2} \right)^{1/2},$$

gives time complexity

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

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.

Finally, since $v = (u+1)/2$,

$$u^u v^{-3v/2} = \exp\left(u \log u - \frac{3(u+1)}{4} \log\left(\frac{u+1}{2}\right)\right),$$

which yields the claimed running time.

For the space complexity, the w distinguished-point records require $O(w \log N)$ bits, in virtue of Lemma 2. 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.

Direct modular-polynomial evaluation. When storing all modular polynomials up to level B is undesirable, we instead compute $\Phi_\ell(j(E_i), Y)$ on demand at each ℓ-step. Unlike the stored variant, this replaces the one-time construction of Φ_ℓ by a recurring cost inside every evaluation of h_E, since the same level ℓ may be encountered at unrelated j-invariants along a vOW trail.

Algorithms for evaluating $\Phi_\ell(j, Y)$ without storing the full polynomial were developed by Sutherland [25]; more space-efficient variants appear in [24], although Sutherland's methods remain faster in practice. For simplicity, we use an on-demand BLS implementation [6], which has essentially the same complexity as [25]: for each ℓ-step, $\Phi_\ell \bmod p$ is constructed, specialized, and discarded. Including root finding, one local step costs

$$C_{\mathrm{dir}}(\ell, p) = O\big(\ell^3 (\log \ell)^3 \log\log \ell + \mathsf{M}(\ell) \log \ell \log(\ell p)\big). \tag{7}$$

Thus an indexed smooth isogeny can be evaluated in

$$t_{\mathrm{direct}} = O(C_{\mathrm{dir}}(B, p) \log X) \tag{8}$$

time. Crucially, this approach avoids the persistent B^3-size modular-polynomial database required by the stored mode.

Theorem 3 (vOW-direct). *Under the assumptions of Theorem 2, suppose that the required modular polynomial is computed on demand using the BLS algorithm and discarded after each local step. Assuming GRH and* $\mathsf{M}(B) = O(B \log B \log\log B)$, *Algorithm 3 has expected running time*

$$T_{\mathrm{vOW}}^{\mathrm{dir}} = O\Bigg(\frac{p^{1/2}}{\sqrt{w}} (\log p)^{5/2} \left[B^{9/2} \log B + B^{5/2} \log(Bp)\right] (\log B)^2 \log\log B \cdot \exp\left(u \log u - \frac{3(u+1)}{4} \log\left(\frac{u+1}{2}\right)\right)\Bigg),$$

where $u = \frac{\log(p/2)}{3 \log B}$. *Its local space requirement is* $O\big(w \log N + B^2\big((\log B)^2 + \log p\big)\big)$ *bits, up to lower-order indexing and working storage.*

Proof. The proof is analogous to that of Theorem 2, only the cost of evaluating an indexed smooth isogeny changes. Substituting t_{direct} of Equation (8) into vOW bound $O(u^u t_{\mathrm{direct}}\sqrt{N^3/w})$ and the bound on N (Equation (4)) yields the claimed running time. Space complexity easily follows from the complexity of BLS [6].

Remark on other alternatives. An alternative to modular-polynomial evaluation is to work explicitly with torsion points. Given a fixed basis $E[\ell] = \langle P, Q\rangle$, the $\ell + 1$ outgoing isogenies may be indexed by the cyclic subgroups $\langle Q\rangle$ and $\langle P + [a]Q\rangle$, for $a \in \mathbb{F}_\ell$. This avoids evaluating $\Phi_\ell(j, Y)$ and, in principle, provides the deterministic branch selection required by the vOW iteration.

There are, however, two drawbacks. First, one must compute a point of order ℓ and fix a deterministic encoding of the relevant ℓ-torsion so that the same branch index always reconstructs the same kernel. The method of [3, Section 4.4] constructs an ℓ-isogeny from such torsion data. However, the required torsion is generally defined over an extension of degree $O(\ell)$ [14], leading to a cost $C_{\mathrm{tors}}(\ell, p) = \widetilde{O}(\ell^2)$ base-field operations. For primes near the smoothness bound, this gives $C_{\mathrm{tors}}(\ell, p) = \widetilde{O}(B^2)$, which may appear more efficient than the direct-mode cost in Equation (7). However, the underlying arithmetic is performed in an extension field whose degree grows with B, and is therefore substantially more expensive in practice than arithmetic directly over $\mathbb{F}_{p^2}$.

By comparison, the modular-polynomial approach remains entirely over $\mathbb{F}_{p^2}$. With precomputed modular polynomials, a local ℓ-step costs

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

while also providing a natural deterministic enumeration of the outgoing edges. Thus, torsion-based evaluation may be advantageous when the required torsion is defined over a small extension, whereas for generic ℓ, and particularly for ℓ close to B, the large extension degree makes modular-polynomial evaluation more attractive in our setting. A detailed concrete analysis of this alternative is therefore left for future work.

5.2 Effect of vOW on concrete security.

Stored-mode case. For $\log_2 p = 256$, the optimal smoothness bound remains essentially constant at $B_{\mathrm{opt}} \approx 154.5$ throughout the memory range considered. Since the modular-polynomial database is small relative to the total budget, nearly all additional memory can be used for distinguished-point records. Hence

$$T_{\mathrm{vOW}}^{\mathrm{store}} \propto w^{-1/2} \approx S^{-1/2},$$

so every additional 10 bits of memory reduce the running-time exponent by approximately 5 bits.

Compared with the full-list attack, vOW is substantially faster in the low-memory regime, saving about 23 bits of running time at 2^{40} bits of space and

	$\log_2 p = 256$			$\log_2 p = 512$		
Space bound	B_{vOW}	$\log_2 T_{\mathrm{vOW}}^{\mathrm{store}}$	$\log_2 T_{\mathrm{full}}$	B_{vOW}	$\log_2 T_{\mathrm{vOW}}^{\mathrm{store}}$	$\log_2 T_{\mathrm{full}}$
2^{40}	154.48	**176.98**	200.23	1403.21	**327.58**	404.59
2^{50}	154.51	**171.98**	187.65	2002.20	**322.36**	391.90
2^{60}	154.51	**166.98**	175.31	2003.91	**317.36**	379.24
2^{70}	154.51	**161.98**	163.40	2003.91	**312.36**	366.59
2^{80}	154.51	156.98	**152.31**	2003.91	**307.36**	353.99

Table 4. Comparison between the time-optimized vOW-stored and full-list attacks under fixed space bounds. Bold entries indicate the smaller running time for each parameter set.

15 bits at 2^{50}. The advantage decreases with memory, and the two methods are essentially tied near 2^{70} bits. Beyond this point, Algorithm 2 becomes preferable, since it can use larger B and exploit batching and reuse of modular-polynomial evaluations, whereas vOW repeatedly reconstructs individual indexed smooth isogenies.

For $\log_2 p = 512$, the modular-polynomial database is non-negligible at 2^{40} bits, forcing a smaller optimal B. From 2^{50} bits onward, however, the optimum stabilizes near $B_{\mathrm{opt}} \approx 2 \times 10^3$, after which additional memory is again devoted mainly to distinguished-point records, yielding the same approximate $S^{-1/2}$ scaling.

Here the full-list algorithm remains strongly memory constrained throughout the 2^{40}–2^{80}-bit range. Consequently, vOW remains substantially faster throughout, improving the running-time exponent by about 77 bits at 2^{40} bits of memory and about 46 bits even at 2^{80}. Thus, unlike the 256-bit case, no crossover with the full-list method is observed for $\log_2 p = 512$ within the memory range considered.

These estimates use only the leading asymptotic terms, with hidden constants set to one; in practice, these constants are expected to be larger.

	$\log_2 p = 256$			$\log_2 p = 512$		
Space bound	B_{dir}	$\log_2 T_{\mathrm{vOW}}^{\mathrm{dir}}$	$\log_2 T_{\mathrm{full}}$	B_{dir}	$\log_2 T_{\mathrm{vOW}}^{\mathrm{dir}}$	$\log_2 T_{\mathrm{full}}$
2^{40}	48.36	**184.26**	200.23	343.57	**340.63**	404.59
2^{50}	48.36	**179.26**	187.65	343.58	**335.63**	391.90
2^{60}	48.36	**174.26**	175.31	343.58	**330.63**	379.24
2^{70}	48.36	169.26	**163.40**	343.58	**325.63**	366.59
2^{80}	48.36	164.26	**152.31**	343.58	**320.63**	353.99

Table 5. Time-optimized parameters for the vOW-direct variant under fixed space bounds.

Direct-mode case. For $\log_2 p = 256$, the direct variant is optimized near $B \approx 48$, while for $\log_2 p = 512$ the optimum is near $B \approx 344$. In both cases,

the working memory of a single BLS computation is negligible relative to the considered space bounds, so additional memory is devoted almost entirely to distinguished-point records, yielding the expected $S^{-1/2}$ scaling. The direct variant outperforms the full-list attack only in the lowest-memory regime for 256-bit p, but remains faster throughout the considered range for 512-bit p.

Whenever the modular-polynomial database fits in memory, however, the stored variant is faster. For $\log_2 p = 256$, this database is only about 2^{27} bits at the optimum, making direct mode largely unnecessary on a single machine. Its main relevance is in parallel vOW implementations with limited local memory: distinguished-point records can be shared globally, whereas the modular-polynomial database must be replicated or accessed remotely. With m workers, replication increases its effective storage from S_Φ to mS_Φ, while direct mode requires only temporary BLS working memory per worker.

This distinction becomes more important at larger parameters. For a 1024-bit characteristic, the time-optimal stored parameters occur near $B \approx 6 \times 10^4$, where the database already reaches roughly 2^{54} bits, or 2252 TB. Keeping such a database in fast local memory, especially under replication, is impractical. Thus, although slower for the 256- and 512-bit cases, direct mode remains a natural fallback for memory-constrained parallel implementations and larger parameters.

Comparison with previous EndRing algorithms. To assess the concrete security implications of the above tradeoffs, it is important to compare them with the previously known algorithms for computing supersingular endomorphism rings that are closest in spirit to our setting. Eisenträger et al. [13, Theorem 3.7] gave, under GRH and additional heuristics, an algorithm for computing $\mathrm{End}(E)$ in time $O\big(p^{1/2}(\log p)^2\big)$ using only polynomial space. More concretely, inspection of their algorithm shows that it stores only two paths P_1, P_2 from E to curves that are ℓ-isogenous to their Frobenius conjugates. Each path is represented by a sequence of curves, or equivalently j-invariants, and has length $\Theta\big(\log\big(p^{3/4}(\log\log p)^{1/2}\big)\big)$. Thus, for the parameter sizes considered here, its memory requirement is negligible compared with those of our attacks like Delfs-Galbraith [11,16].

More recently, Page and Wesolowski [23] gave an unconditional $\widetilde{O}(p^{1/2})$-time algorithm and established a polynomial-time equivalence between the OneEnd and EndRing problems. However, the corresponding reduction is not tight: as observed in [1, Section 6], using a OneEnd solver to solve EndRing incurs a multiplicative overhead of roughly $2^{94}\log^{13}(p)$ in the running time. We therefore use the explicit bound of [13] as a convenient concrete low-memory benchmark.

With the same convention used throughout our estimates, namely natural logarithms and hidden constants set to one, the EHLMP bound corresponds to approximately

$$2^{142.94} \qquad \text{and} \qquad 2^{272.94}$$

operations for 256- and 512-bit characteristics, respectively. Thus, throughout the memory range 2^{40}–2^{80} considered in Table 4, the previous $p^{1/2}$-time algorithm remains faster than both the full-list and vOW variants of the new attack.

This comparison is summarized by the following table, where the "new attack" column gives the smallest running time obtained from the full-list and vOW-stored variants within the space bounds considered above.

$\log_2 p$	Space range	Best new attack	EHLMP	Security target
256	2^{40}–2^{80}	$2^{152.31}$ (full)	$\mathbf{2^{142.94}}$	128 bits
512	2^{40}–2^{80}	$2^{307.36}$ (vOW)	$\mathbf{2^{272.94}}$	256 bits

Table 6. Comparison with the classical low-memory EndRing benchmark. The best new-attack values are taken over the space range considered in 4.

The conclusion is therefore somewhat striking. Although Wesolowski's algorithm improves the asymptotic exponent from $1/2$ to $1/3 + o(1)$, its concrete advantage is strongly memory-dependent. With 2^{80} bits of storage, the best estimates are $2^{152.31}$ operations for a 256-bit characteristic and $2^{307.36}$ for a 512-bit characteristic, still nearly 9 and 34.4 bits slower than EHLMP, respectively.

Moreover, EHLMP itself remains above the target security levels: $2^{142.94} > 2^{128}$ and $2^{272.94} > 2^{256}$. Thus, within the realistic memory range considered, Wesolowski's attack does not reduce the classical security below NIST Levels I and V; the best attack remains the polynomial-space $p^{1/2}$-time EndRing algorithm.

6 Quantum Improvements

Quantum claw formulation. For a fixed randomized curve E, consider two copies of the indexed smooth-isogeny space $\mathcal{I}_{X,B}$, with $N = \#\mathcal{I}_{X,B}$, and define

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

where $\psi_{\mathbf{r}} : E \to E_{\mathbf{r}}$ and $\psi_{\mathbf{s}} : E \to E_{\mathbf{s}}$ are the corresponding decoded smooth isogenies. A claw $F_E(\mathbf{r}) = G_E(\mathbf{s})$ gives a Frobenius-conjugate pair of codomains and hence, after computing the appropriate isomorphism ι, an isogeny $\widehat{\psi_{\mathbf{s}}}^{(p)} \circ \iota \circ \psi_{\mathbf{r}} : E \to E^{(p)}$. Thus the matching phase is a claw-finding problem on two domains of size N.

Following Jaques and Schanck [18], let E_G, E_D, E_W denote, respectively, the gate count, depth, and width of a reversible circuit evaluating F_E or G_E. These measure the total number of elementary gates, the longest sequential dependency chain, and the maximum number of simultaneously active qubits. In our setting, such an evaluation must coherently decode the index and reconstruct the corresponding smooth isogeny; the additional Frobenius operation in G_E is negligible by comparison.

Tani's claw-finding algorithm. Tani's quantum-walk algorithm [26] finds a claw between two balanced domains of size N using $O(N^{2/3})$ oracle queries. Since $N = p^{1/3+o(1)}$ in our setting, this suggests a $p^{2/9+o(1)}$-time inner search at the level of query complexity.

This estimate, however, ignores the cost of maintaining Tani's coherent quantum data structure. Following the analysis of Jaques and Schanck [18, Section 5.5], let m be the number of bits needed to encode an element in the list and let $R \leq N^{2/3}$ denote the size of the Johnson-graph data structure needed by Tani. Specializing the general bounds [18, section 5.5] to our balanced case gives

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

The first term in the gate count accounts for maintaining the quantum data structure and is absent from the query-complexity estimate.

Indeed, the query-optimal choice $R = N^{2/3}$ yields only $O(N^{2/3})$ function evaluations, but requires $G_{\mathrm{Tani}} = O(mN^{4/3} + E_G N^{2/3})$ gates and $W_{\mathrm{Tani}} = O(mN^{2/3} + E_W)$ qubits. Thus, when $m, E_G = N^{o(1)}$, the gate complexity is actually $N^{4/3+o(1)} = p^{4/9+o(1)}$, showing that the apparent $p^{2/9+o(1)}$ attack is not realistic in this cost model.

More generally, provided $E_G/m \leq N^{2/3}$, the gate count is minimized near $R = E_G/m$, giving $G_{\mathrm{Tani}} = O(N\sqrt{mE_G})$. Tani can therefore still provide a concrete improvement when oracle evaluation is expensive, but this should be viewed as a quantum time–memory tradeoff rather than an unconditional $N^{2/3}$ running-time improvement.

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 $\mathcal{I}^2_{X,B}$. Since this space has size N^2, Grover requires $O(N)$ iterations. In the notation above, the Jaques–Schanck [18] 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 therefore uses more oracle evaluations than query-optimal Tani, but avoids its large coherent quantum data structure. In the regime $m, E_G, E_D, E_W = N^{o(1)}$, both Grover and gate-optimized Tani have $N^{1+o(1)} = p^{1/3+o(1)}$ asymptotic cost, while Grover requires substantially less coherent memory. This mirrors the conclusion of Jaques and Schanck for SIKE, where the query-optimal parameterization of Tani's algorithm is not the cost-optimal one in realistic fault-tolerant models.

Remark 3. One might hope to apply Grover search to the outer rerandomization step and reduce the expected u^u repetitions to $u^{u/2}$. This would require, however,

an efficiently evaluable predicate

$$f(E') = \begin{cases} 1, & \text{if } E' \text{ admits the required } B\text{-smooth isogeny to } E'^{(p)}, \\ 0, & \text{otherwise.} \end{cases}$$

No such efficient predicate is known. Indeed, determining whether $f(E') = 1$ appears to require essentially solving the underlying problem: one must either find the smooth isogeny to the Frobenius conjugate or recover sufficient endomorphism-ring information to certify its existence. Thus, unlike the inner claw search, the outer rerandomization does not immediately admit a Grover speedup. We therefore retain the classical u^u rerandomization factor in our quantum analysis.

Parallelization and depth constraints. Grover search does not parallelize linearly: using m independent quantum processors reduces the depth by only a factor $\sqrt{m}$, and this tradeoff is optimal [29]. Thus, reducing a circuit of depth D_{Grover} to a target depth $D_{\max}$ requires

$$m \geq \left(\frac{D_{\text{Grover}}}{D_{\max}} \right)^2.$$

This is particularly relevant for fault-tolerant computation, where long serial computations are costly. NIST, for example, uses 2^{40} logical-gate layers as an indicative upper bound on the depth of a one-year quantum computation [21]. Hence, if $D_{\text{Grover}} \approx 2^d$, achieving depth 2^{40} requires roughly $2^{2(d-40)}$ parallel processors.

For Tani's algorithm, the parallel strategy of Jeffery et al. [19] offers a better tradeoff, but requires substantial communication during the checking step. A realistic analysis must therefore account for processor topology and communication costs, which we leave for future work.

Concrete quantum estimates. Following the concrete methodology of Jaques and Schanck [18], we assume that the classical and quantum gate counts of the underlying arithmetic are equal, which is favorable to the attacker. Writing $L = \log_2 p$, we charge $4L \log_2 L$ gates per dominant $\mathbb{F}_{p^2}$-operation and take

$$E_G = 4L \log_2 L \, C_{\text{eval}}^{\text{store}}(B, p), \qquad E_W = 2L, \qquad E_D = E_G / E_W,$$

where $C_{\text{eval}}^{\text{store}}(B, p)$ is the stored-mode evaluation cost derived in Equation (6). For Tani's algorithm we further take $m = 2L + \log_2 N$, sufficient to encode an indexed endpoint $(\mathbf{r}, F_E(\mathbf{r}))$, and use the gate-optimal parameter $R = E_G/m$. The outer rerandomization factor remains u^u.

Grover gives the clearest low-memory quantum attack. After optimizing B, it requires approximately $2^{157.7}$ and $2^{266.9}$ gates for 256- and 512-bit characteristics, respectively, with only about 2^9 and 2^{10} logical qubits. Thus, its main advantage

	$\log_2 p = 256$			$\log_2 p = 512$		
Method	B_{opt}	$\log_2 G$	$\log_2 W$	B_{opt}	$\log_2 G$	$\log_2 W$
Grover	841	157.72	9.00	21646	266.92	10.00
Tani	2540	141.45	44.98	88426	246.24	55.23

Table 7. Optimized quantum gate and width estimates for computing a smooth isogeny to the Frobenius conjugate.

over classical attacks is eliminating the large smooth-isogeny table rather than substantially reducing time.

Gate-optimized Tani lowers the costs to approximately $2^{141.5}$ and $2^{246.2}$ gates, but requires roughly 2^{45} and 2^{55} logical qubits. Hence, while Tani crosses the nominal 2^{256} Level-V threshold in our optimistic model, it does so only at an extreme coherent-memory cost. In contrast, Grover remains above 2^{128} and 2^{256}, respectively, with negligible quantum memory.

7 Conclusion

In this work, we assessed the concrete impact of Wesolowski's $p^{1/3+o(1)}$ algorithm and showed that its main practical bottleneck is the memory required by the smooth-isogeny search. Although vOW substantially reduces this requirement, the previous low-memory $p^{1/2}$-time EndRing algorithms remain preferable over the parameter ranges considered. A natural direction for future work is therefore to seek even lower-memory collision-search techniques, potentially of Pollard-ρ type. Whether such methods can be adapted efficiently is unclear since the smooth-isogeny search space lacks the explicit group structure available in classical group-action settings.

Acknowledgment. The author acknowledges support of the NSERC Alliance Consortia Quantum Grant "Accelerating the transition to quantum-resistant cryptography" (ALLRP 578463–2022), and the NSERC Discovery Grant program held by Professor David Jao.

References

1. Aardal, M.A., Basso, A., Feo, L.D., Patranabis, S., Wesolowski, B.: A complete security proof of SQIsign. In: Advances in Cryptology – CRYPTO 2025. Lecture Notes in Computer Science, vol. 16005, pp. 190–222. Springer (2025). https://doi.org/10.1007/978-3-032-01887-8_7
2. Aubry, Y., Oyono, R., Vincent, C.: Minimal degree of an isogeny between a supersingular elliptic curve and its conjugate. arXiv preprint arXiv:2607.14624 (2026), https://arxiv.org/abs/2607.14624, arXiv:2607.14624 [math.NT]

3. Basso, A., Borin, G., Castryck, W., Santos, M.C.R., Invernizzi, R., Leroux, A., Maino, L., Vercauteren, F., Wesolowski, B.: PRISM: Simple and compact identification and signatures from large prime degree isogenies. In: Public-Key Cryptography – PKC 2025. pp. 300–332. Springer (2025). https://doi.org/10.1007/978-3-031-91826-1_10
4. Basso, A., Codogni, G., Connolly, D., Feo, L.D., Fouotsa, T.B., Lido, G.M., Morrison, T., Panny, L., Patranabis, S., Wesolowski, B.: Supersingular curves you can trust. In: Advances in Cryptology – EUROCRYPT 2023. pp. 405–437. Springer (2023). https://doi.org/10.1007/978-3-031-30617-4_14
5. Beullens, W., Kleinjung, T., Vercauteren, F.: CSI-FiSh: Efficient isogeny based signatures through class group computations. Cryptology ePrint Archive, Paper 2019/498 (2019), https://eprint.iacr.org/2019/498
6. Bröker, R., Lauter, K., Sutherland, A.V.: Modular polynomials via isogeny volcanoes. Mathematics of Computation **81**(278), 1201–1231 (2012). https://doi.org/10.1090/S0025-5718-2011-02508-1
7. de Bruijn, N.G.: The asymptotic behaviour of a function occurring in the theory of primes. Journal of the Indian Mathematical Society **15**, 25–32 (1951)
8. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An efficient post-quantum commutative group action. In: Advances in Cryptology – ASIACRYPT 2018. pp. 395–427. Springer (2018)
9. Charles, D.X., Goren, E.Z., Lauter, K.E.: Cryptographic hash functions from expander graphs. Journal of Cryptology **22**(1), 93–113 (2009). https://doi.org/10.1007/s00145-007-9002-x
10. Costello, C., Longa, P., Naehrig, M., Renes, J., Virdia, F.: Improved classical cryptanalysis of SIKE in practice. In: Public-Key Cryptography – PKC 2020. Lecture Notes in Computer Science, vol. 12111, pp. 505–534. Springer (2020). https://doi.org/10.1007/978-3-030-45388-6_18
11. Delfs, C., Galbraith, S.D.: Computing isogenies between supersingular elliptic curves over $\mathbb{F}_p$. Designs, Codes and Cryptography **78**(2), 425–440 (2016). https://doi.org/10.1007/s10623-014-0010-1
12. Eisenträger, K., Hallgren, S., Lauter, K.E., Morrison, T., Petit, C.: Supersingular isogeny graphs and endomorphism rings: reductions and solutions. In: EUROCRYPT 2018 (2018). https://doi.org/10.1007/978-3-319-78372-7_11
13. Eisenträger, K., Hallgren, S., Leonardi, C., Morrison, T., Park, J.: Computing endomorphism rings of supersingular elliptic curves and connections to path-finding in isogeny graphs. In: Algorithmic Number Theory Symposium (ANTS XIV). The Open Book Series, vol. 4, pp. 215–232. Mathematical Sciences Publishers (2020)
14. Eriksen, J.K., Panny, L., Sotáková, J., Veroni, M.: Deuring for the people: Supersingular elliptic curves with prescribed endomorphism ring in general characteristic. In: LuCaNT: LMFDB, Computation, and Number Theory, Contemporary Mathematics, vol. 796, pp. 339–373. American Mathematical Society (2024)
15. Feo, L.D., Kohel, D., Leroux, A., Petit, C., Wesolowski, B.: SQISign: Compact post-quantum signatures from quaternions and isogenies. In: Advances in Cryptology – ASIACRYPT 2020. pp. 64–93. Springer (2020). https://doi.org/10.1007/978-3-030-64837-4_3
16. Galbraith, S.D., Stolbunov, A.: Improved algorithm for the isogeny problem for ordinary elliptic curves. arXiv preprint arXiv:1105.6331 (2011), arXiv:1105.6331 [math.NT]
17. von zur Gathen, J., Gerhard, J.: Modern Computer Algebra. Cambridge University Press, 3 edn. (2013)

18. Jaques, S., Schanck, J.M.: Quantum cryptanalysis in the RAM model: Claw-finding attacks on SIKE. In: Advances in Cryptology – CRYPTO 2019. Lecture Notes in Computer Science, vol. 11692, pp. 32–61. Springer (2019). https://doi.org/10.1007/978-3-030-26948-7_2
19. Jeffery, S., Magniez, F., de Wolf, R.: Optimal parallel quantum query algorithms. Algorithmica **79**(2), 509–529 (2017). https://doi.org/10.1007/s00453-016-0203-7
20. Kohel, D.: Endomorphism rings of elliptic curves over finite fields. Ph.D. thesis, University of California, Berkeley (1996)
21. National Institute of Standards and Technology: Submission requirements and evaluation criteria for the post-quantum cryptography standardization process. NIST Post-Quantum Cryptography Standardization Call for Proposals (Dec 2016), https://csrc.nist.gov/csrc/media/projects/post-quantum-cryptography/documents/call-for-proposals-final-dec-2016.pdf
22. van Oorschot, P.C., Wiener, M.J.: Parallel collision search with cryptanalytic applications. Journal of Cryptology **12**(1), 1–28 (1999). https://doi.org/10.1007/PL00003816
23. Page, A., Wesolowski, B.: The supersingular endomorphism ring and one endomorphism problems are equivalent. In: Advances in Cryptology – EUROCRYPT 2024. pp. 388–417. Springer (2024). https://doi.org/10.1007/978-3-031-58751-1_14
24. Santos, M.C.R., Eriksen, J.K., Leroux, A., Meyer, M., Panny, L.: Evaluation of modular polynomials from supersingular elliptic curves. In: LuCaNT: Databases, Algorithms, and Computational Number Theory. Contemporary Mathematics, vol. 840, pp. 311–338. American Mathematical Society (2026)
25. Sutherland, A.V.: On the evaluation of modular polynomials. In: Algorithmic Number Theory Symposium (ANTS X). The Open Book Series, vol. 1, pp. 531–555. Mathematical Sciences Publishers (2013)
26. Tani, S.: Claw finding algorithms using quantum walk. Theoretical Computer Science **410**(50), 5285–5297 (2009)
27. Wesolowski, B.: The supersingular isogeny path and endomorphism ring problems are equivalent. In: FOCS 2022. pp. 1100–1111. IEEE (2022). https://doi.org/10.1109/FOCS52979.2021.00109
28. Wesolowski, B.: The supersingular isogeny problem in time and memory $p^{1/3+o(1)}$. Cryptology ePrint Archive (2026/1486) (2026), report 2026/1486
29. Zalka, C.: Grover's quantum searching algorithm is optimal. Physical Review A **60**(4), 2746–2751 (1999). https://doi.org/10.1103/PhysRevA.60.2746