Leveled Isogeny Problems with Hints

Subham Das[1], Riccardo Invernizzi[2], Péter Kutas[3,4], and Jonas Meers[5]

[1] CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
subham.das@cispa.de
[2] COSIC, KU Leuven, Belgium
riccardo.invernizzi@esat.kuleuven.be
[3] Eötvös Loránd University, Hungary
kutasp@gmail.com
[4] University of Birmingham, United Kingdom
[5] Ruhr University Bochum, Germany
research@meers.org

Abstract. We define and analyze the Leveled Isogeny Problem with Hints (LIPH), which is a generalization of the Isogeny Problem with Level Structure first introduced by De Feo, Fuoutsa and Panny at EUROCRYPT'24. In a LIPH instance we are tasked to recover a secret isogeny φ given masked torsion point images $\Gamma \cdot (\varphi(P), \varphi(Q))^\top$ for some (P, Q) of order N and unknown $\Gamma \in \mathrm{GL}_2(N)$. Additionally, we are provided a *hint* on Γ, revealing some bits of its entries. Instances of LIPH occur naturally in the case of modern isogeny-based key exchanges that use masked torsion points as part of their public key, when additionally some parts of the masking matrix Γ are revealed due to, for instance, a side-channel attack.
We provide efficient algorithms that solve various instances of LIPH, leading to efficient *partial key recovery attacks* in practice. More specifically, we present Coppersmith-type attacks that are able to recover an M-SIDH/POKÉ secret key given 50% (resp. 86%) of the most-significant bits of an entry of Γ, and a FESTA secret key given the 67% of the most-significant bits of Γ. In the case of FESTA we also present a tailored combinatorial attack running in subexponential time $O(2^{\sqrt{n}})$ when 50% of the bits of Γ leak at random.

1 Introduction

Introduced in 2011 by De Feo and Jao, the Supersingular Isogeny Diffie-Hellman (SIDH) [18] was one of the most compact post-quantum key exchanges and the flagship of isogeny-based cryptography at the time. On a high level, in SIDH two parties Alice and Bob choose independently a secret isogeny $\varphi_A : E_0 \to E_A$ and $\varphi_B : E_0 \to E_B$ between supersingular elliptic curves and the codomains are subsequently exchanged. By applying their secret isogeny on the curve of the respective other party, the goal is to compute a shared curve E_{AB} only known to Alice and Bob. However, due to the non-commutative nature of isogenies, SIDH requires to publish additional *torsion point information*. Concretely, besides the

curve E_A Alice needs to publish the image points $(\varphi_A(P), \varphi_A(Q))$ for some basis $(P, Q) \in E_0[N]$ of large order N (and similarly for Bob).

Although it was theorized for a long time that these additional torsion points might enable attacks on SIDH, the protocol withstood (almost) all cryptanalytic efforts for over a decade. Finally, in 2022 Castryck and Decru [6] were able to exploit the provided torsion point information in a devastating attack, which was subsequently improved by Maino, Martindale, Panny, Pope, Wesolowski [20] and Robert [29]. The attack builds upon *higher-dimensional* isogenies between Abelian varieties and requires the knowledge of the degree $\deg \varphi_A$ as well as torsion point images $(\varphi_A(P), \varphi_A(Q))$ of order at least $N \geq \sqrt{\deg \varphi_A}$. All these quantities are readily available in SIDH as either system parameters or in form of the public key.

Following the attacks and the emergence of constructive use-cases of the HD-technique, many variants of isogeny-based key exchanges were developed that resist the SIDH attacks [14,25,4,3]. They all make use of the fact that the commutativity of SIDH does not require the *exact* torsion point images $(\varphi_A(P), \varphi_A(Q))$. Instead, publishing the torsion point images *up to subgroup* is sufficient. For instance, the M-SIDH protocol [14] uses public keys of the form

$$(E_A, [\alpha]\varphi_A(P), [\alpha]\varphi_A(Q)), \quad \alpha \in \mathbb{Z}_N^\times$$

where the scalar α is commonly referred to as *masking scalar*. As mentioned before, such a scalar does not influence the correctness of the key exchange as $\langle P + Q\rangle = \langle [\alpha]P + [\alpha]Q\rangle$ for any two points $(P, Q) \in E[N]$, which is the only property required for an SIDH-style key exchange. However, the SIDH attack is not applicable anymore as the secret isogeny now effectively has much larger degree $\alpha^2 \deg \varphi_A$, violating the condition $N \geq \sqrt{\alpha^2 \deg \varphi_A}$ since $\alpha \in \mathbb{Z}_N^\times$.

A slightly different idea comes from the FESTA [4] family (QFESTA [25] and POKÉ [3]). Instead of building a Diffie-Hellman-like key exchange one can use the SIDH attacks in a trapdoor setting as prioneered in the now broken scheme Séta [11]. Here the underlying problem is differen from M-SIDH as the isogeny images are masked by two different scalars α and β. Furthermore, in the case of POKÉ, the degree $\deg \varphi_A$ additionally remains hidden. The security of these protocols thus relies on the assumption that it is hard to recover the secret isogeny φ_A given (some variant of) masked torsion point information.

To assess the hardness of these types of assumptions, De Feo, Fouotsa and Panny [12] analyzed isogeny problems with *level structure* – a notion first introduced by Arpin [1]. Here, a level structure of level N is defined as a basis $E[N]$ up to transformation by some subgroup of $\mathrm{GL}_2(N)$. The corresponding isogeny problem is then essentially the task of recovering a secret isogeny $\varphi : E \to E'$ given the tuple

$$(E, E', P, Q, P', Q'), \quad (P', Q') = \Gamma \cdot (\varphi(P), \varphi(Q))^\top$$

for some basis $(P, Q) \in E[N]$ and unknown $\Gamma \in \mathrm{GL}_2(N)$. Evidently, the isogeny problem with level structure exactly models the task of recovering the secret

isogeny from an M-SIDH or (Q)FESTA public key, whereas in the case of POKÉ an additional hardness arises since the degree of the isogeny is unknown.

Although easy instances of the isogeny problem with level structure exist, the authors in [12] conclude that instances arising from public keys of masked SIDH variants remain cryptographically hard to solve.

1.1 Our Contributions

In this paper, we study a variant of the isogeny problem with level structure where we are given an additional *hint* on the matrix Γ. This models the situation where part of the secret key leaks during key generation due to, for instance, bad randomness or side-channel attacks. Our work thus broadly falls into the category of *partial key exposure attacks*, which have been conducted on other post-quantum protocols before [13,32,19].

To this end, we define the so-called *Leveled Isogeny Problem with Hints (LIPH)*, which can informally be defined as follows:

$$\textit{Given } (E, E', P, Q, P', Q') \textit{ with } (P', Q') = \Gamma \cdot (\varphi(P), \varphi(Q))^\top \textit{ as well as some bits of } \Gamma\textit{, recover the secret isogeny } \varphi.$$

Depending on the size and type of the hint, we are able to efficiently solve LIPH, leading to efficient attacks in practice. In particular, our attacks do not require a hint on φ. The attacks all have in common that a LIPH instance gives rise to a *pairing equation* of the form

$$\deg \varphi \cdot \det \Gamma - r \equiv 0 \bmod N$$

for some known $r \in \mathbb{Z}_N$. We then incorporate the hint on Γ to efficiently solve the pairing equation. Concretely, our attacks can be summarized as follows:

- In the case of M-SIDH, we can efficiently solve the arising LIPH instance if 50% of the most-significant bits of an entry in Γ leak. Note that M-SIDH uses $\Gamma = \left(\begin{smallmatrix} \alpha & \\ & \alpha \end{smallmatrix}\right)$, hence a hint on a single scalar yields a hint on all scalars in Γ. The attack is based on the automated variant of the famous Coppersmith method [9] introduced in [23] and recovers the secret isogeny within at most a couple of hours, depending on the security level.
- In the case of (a compressed, four-dimensional variant of) POKÉ we require knowledge of roughly 86% of the most-significant bits of an entry in Γ. Here the matrix is again $\Gamma = \left(\begin{smallmatrix} \alpha & \\ & \alpha \end{smallmatrix}\right)$, however the attack is made more difficult by the fact that the degree of the isogeny φ is secret. Fortunately, in the four-dimensional variant $\deg \varphi$ is small compared to the order of the provided (masked) torsion points, hence still allowing for an efficient attack. The attack is again based on the automated variant of Coppersmith's method, running in a few hours at most.
- In the case of FESTA, a first attack based on the automated Coppersmith method requireds knowledge of the 67% most-significant bits of Γ. Here $\Gamma = \left(\begin{smallmatrix} \alpha & \\ & \beta \end{smallmatrix}\right)$ is a diagonal matrix with two distinct entries satisfying $\alpha\beta \equiv 1 \bmod N$. The attack again requires at most a couple of hours to fully recover the secret isogeny.

– A second attack, again in the FESTA setting, requires 50% of randomly distributed bits of α and β. This attack is purely combinatorial, leverages only the fact that $\alpha\beta = 1 \bmod N$ and runs in subexponential complexity $O(2^{\sqrt{b}})$ where $b = \log n$. Similar techniques appeared for instance in [17,16] in the context of RSA factoring, leading to a polynomial-time algorithm but those attacks are not immediately applicable here. Furthermore, we provide a detailed statistical analysis of the attack by providing interesting connections to certain combinatorial sequences.
  Due to the fact that this attack only uses a very specific structure of FESTA, the techniques developed might provide applications outside of isogeny-based cryptography as well.

Finally, we provide proof-of-concept implementations for all our attacks in Sage, available at the link https://github.com/KULeuven-COSIC/liph.

1.2 Related Work

The literature on partial key exposure attacks on post-quantum secure protocols is quite extensive: in [13] the authors conduct attacks on BIKE, Rainbow an NTRU. Furthermore, in [32] similar attacks were developed for the UOV signature scheme. In [22] the authors solve the LWE problem with hints. Lastly, [19] proposes a partial key exposure attack on McEliece.

Regarding isogeny-based schemes, the first leakage attack on SIDH was conducted in [15]. In [23] this attack was extended to CSIDH [8]. However, in both attacks the authors assume partial knowledge on the *shared key* instead of the *secret key*.

Acknowledgments. The authors thank Wouter Castryck for suggesting a closed form of Equation (6), and the organizers of the Leuven Isogeny Days where this project was started. Jonas Meers was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2092 CASA – 390781972. Péter Kutas is partly supported by EPSRC through grant number EP/V011324/1. Péter Kutas is supported by the Hungarian Ministry of Innovation and Technology NRDI Office within the framework of the Quantum Information National Laboratory Program. Kutas is also supported by the grant "EXCELLENCE-151343". Subham Das was supported by the Government of Hungary through the Stipendium Hungaricum scholarship 2024/25. Riccardo Invernizzi is supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement ISOCRYPT – No. 101020788), by the Research Council KU Leuven grant C14/24/099, by CyberSecurity Research Flanders with reference number VOEWICS02, and by Research Foundation - Flanders (FWO) under a PhD Fellowship fundamental research (project number 1138925N). Together with Kutas, they are also supported by the CELSA alliance through the MaCro project.

2 Preliminaries

2.1 Elliptic Curves, Pairings and Isogenies

Let q be a prime power and E be an elliptic curve defined over $\mathbb{F}_q$. We denote the point at infinity with ∞_E. For an extension field $K \supseteq \mathbb{F}_q$ we denote the set of K-rational points by $E(K)$. For an integer n we denote the multiplication-by-n map by $[n]$. Its kernel is the *n-torsion subgroup* $E[n] = \{P \in E : [n]P = \infty_E\}$ and we call an elliptic curve *supersingular* if $E[p] = \{\infty_E\}$. The set of all supersingular elliptic curves is denoted by $\mathcal{E}\ell\ell(\mathbb{F}_q)$.

An isogeny is a morphism $\varphi : E \to E'$ between elliptic curves E, E' such that $\varphi(\infty_E) = \infty_{E'}$. The degree of φ is its degree as a morphism and we call φ *separable* if $\gcd(p, \deg\varphi) = 1$. In this work, we will only consider separable isogenies. We call two elliptic curves *isogenous* if there exists an isogeny between them. An isogeny is an isomorphism of elliptic curves if it has an inverse (which may be defined over the algebraic closure of $\mathbb{F}_q$). In that case, the inverse is again an isogeny. Isomorphic curves have the same *j-invariant*, which is a simple algebraic expression in the coefficients of the curve equation. Thus, one can check whether two elliptic curves are isomorphic by comparing their j-invariant.

An isogeny from E to itself is called *endomorphism*. The set $\mathrm{End}(E)$ of endomorphisms of E (defined over the algebraic closure of the base field) forms a ring under addition and composition and it is thus called the *endomorphism ring*. The multiplication-by-n map $[n]$ is always an endomorphism of E, hence $\mathbb{Z}$ always embeds in $\mathrm{End}(E)$. Any isogeny $\varphi : E \to E'$ is also a group homomorphism from E to E' with finite kernel. In the case where φ is separable we have $\deg\varphi = |\ker\varphi|$. Conversely, any finite subgroup $G \subset E$ corresponds to a separable isogeny $\varphi : E \to E'$ with kernel $\ker\varphi = G$, where φ and E' are unique up to post-composition with an isomorphism. Since E' is essentially uniquely determined by $\ker\varphi$, we will write $E' = E/G$. One can compute φ and E/G via Vélu's formula [33], which can be evaluated in polynomial time in the size of the kernel.

For an elliptic curve E defined over a field $\mathbb{K}$, the *Weil pairing* is a map $e_n : E[n] \times E[n] \to \mu_n$ that takes pairs of n-torsion points to n-th roots of unity [34]. It can be efficiently computed via the Miller function [24] and satisfies the following properties:

- *Bilinearity*: $e_n(P+Q, R) = e_n(P, R)e_n(Q, R)$ and similarly $e_n(P, Q+R) = e_n(P, Q)e_n(P, R)$
- *Alternating*: $e_n(P, P) = 1$ and $e_n(P, Q) = e_n(Q, P)^{-1}$
- *Non-degeneracy*: If $P \neq \infty$ then $e_n(P, Q) \neq 1$ for some $Q \in E[n]$
- *Compatibility*: $e_{mn}(P, Q) = e_n([m]P, Q)$ for all $P \in E[mn]$ and $Q \in E[n]$
- *Galois-equivariant*: $e_n(P^\sigma, Q^\sigma) = e_n(P, Q)^\sigma$ for all $\sigma \in \mathrm{Gal}(\overline{\mathbb{K}}/\mathbb{K})$
- *Isogenies*: $e_n(\varphi(P), \varphi(Q)) = e_n(P, Q)^{\deg\varphi}$ for all $\varphi \in \mathrm{Hom}(E, E')$
- *Surjectivity*: for each $P \in E[n]$ we have $\{e_n(P, Q) : Q \in E[n]\} = \mu_r$, where $r := \mathrm{ord}(P)$

Recently many different ways of representing an isogeny have emerged, leading to the following unified definition.

Definition 2.1 (Efficient Isogeny Representation [31]). *Let $\mathcal{V}, \mathcal{E}$ be two algorithms. Furthermore, let $\varphi : E \to E'$ be an isogeny of degree d defined over $\mathbb{F}_q$. An* efficient representation *of φ (with respect to $\mathcal{V}$ and $\mathcal{E}$) is a bit string $D_\varphi \in \{0,1\}^*$ of length $\mathcal{O}(\operatorname{poly}\log(dq))$ such that:*

- *$\mathcal{V}(E, E', d, D_\varphi)$ returns in time $\mathcal{O}(\operatorname{poly}\log(dq))$ whether D_φ is a valid encoding of a d-isogeny between E and E', and*
- *$\mathcal{E}(E, E', d, D_\varphi, P)$ returns in time $\mathcal{O}(\operatorname{poly}(k\log(dq)))$ the image $\varphi(P)$ of a point $P \in E(\mathbb{F}_{q^k})$.*

2.2 Polynomials

Let $x_1, \ldots, x_k$ be symbolic variables. A *monomial* is a product $x_1^{i_1} \cdot \ldots \cdot x_k^{i_k}$, where $i_1, \ldots, i_k \in \mathbb{N}$. In particular, a product of the form $c \cdot x_1^{i_1} \cdot \ldots \cdot x_k^{i_k}$, where $c \neq 1$, is not a monomial. Let $f(x_1, \ldots, x_k) = \sum_{i_1,\ldots,i_k \in \mathbb{N}} \alpha_{i_1,\ldots,i_k} \cdot x_1^{i_1} \cdot \ldots \cdot x_k^{i_k}$ be a polynomial with coefficients $\alpha_{i_1,\ldots,i_k} \in \mathbb{Z}$. We say that $x_1^{i_1} \cdot \ldots \cdot x_k^{i_k}$ is a *monomial of f*, if $\alpha_{i_1,\ldots,i_k} \neq 0$. If all monomials of f are elements of some set $\mathcal{M}$, then we say that f *is defined over* $\mathcal{M}$. We denote by $\deg(f)$ the *total degree* of f, i.e.,

$$\deg(f) := \max_{\alpha_{i_1,\ldots,i_k} \neq 0} (i_1 + \ldots + i_k).$$

The degree of some finite set of polynomials $\mathcal{F} \subseteq \mathbb{Z}[x_1, \ldots, x_k]$ is defined as

$$\deg(\mathcal{F}) := \max_{f \in \mathcal{F}} \deg(f).$$

Definition 2.2. *For a set of polynomials $\mathcal{F} \subset \mathbb{Z}[x_1, \ldots, x_k]$, we define the set of its* integer roots *as*

$$Z_{\mathbb{Z}}(\mathcal{F}) := \left\{ r = (r_1, \ldots, r_k) \in \mathbb{Z}^k \mid \forall f \in \mathcal{F} : f(r) = 0 \right\}.$$

Similarly, for parameters $N, X_1, \ldots, X_k \in \mathbb{N}$, we define the corresponding set of its small modular roots *as*

$$Z_{N,X_1,\ldots,X_k}(\mathcal{F}) := \left\{ r = (r_1, \ldots, r_k) \in \mathbb{Z}^k \middle| \begin{array}{l} \forall f \in \mathcal{F} : f(r) \equiv 0 \mod N, \\ \forall j : |r_j| \leq X_j \end{array} \right\}.$$

For a finite set $\mathcal{F} = \{f_1, \ldots, f_n\}$, we may abuse notation and write

$$Z_{\mathbb{Z}}(f_1, \ldots, f_n) := Z_{\mathbb{Z}}(\mathcal{F}),$$
$$Z_{N,X_1,\ldots,X_k}(f_1, \ldots, f_n) := Z_{N,X_1,\ldots,X_k}(\mathcal{F}).$$

Definition 2.3. *Let $\mathcal{M}$ be a set of monomials. A* monomial order (on $\mathcal{M}$) *is a total order $\prec$ on $\mathcal{M}$, that satisfies the following two properties:*

1. *For every $\lambda \in \mathcal{M}$, it holds that $1 \prec \lambda$.*
2. *If $\lambda_1 \prec \lambda_2$, then $\lambda \cdot \lambda_1 \prec \lambda \cdot \lambda_2$ for every monomial $\lambda \in \mathcal{M}$.*

We only use the *lexicographic monomial order* $\prec_{\mathsf{lex}}$. The *leading monomial* of a polynomial f (with respect to some monomial order $\prec$) is the unique monomial λ of f, which satisfies $\lambda' \prec \lambda$ for every monomial λ' of f. The coefficient of the leading monomial is called *leading coefficient*. If the monomial order is clear from the context, we denote by $\mathsf{LM}(f)$ and $\mathsf{LC}(f)$ the leading monomial and the leading coefficient of f, respectively. If $\mathsf{LC}(f) = 1$, then we say that f is *monic*.

2.3 (Automated) Coppersmith's Method

Coppersmith's method is a lattice-based algorithm to find small roots of a system of polynomial equations [9], where the polynomials can either be defined over the integers or a finite ring $\mathbb{Z}_N$. While Coppersmith's method was only able to handle uni- and bivariate polynomials at first, it can be extended to the multivariate case as well [5,10]. In this paper, we only consider the modular, multivariate variant where the polynomial system is given by $f_1, \ldots, f_n \in \mathbb{Z}_N[x_1, \ldots, x_k]$.

On a high level, Coppersmith's method transforms the polynomial system defined over $\mathbb{Z}_N[x_1, \ldots, x_k]$ into a polynomial system defined over $\mathbb{Z}[x_1, \ldots, x_k]$. To this end, Coppersmith's method uses lattice techniques to construct a set of polynomials $h_1, \ldots, h_k \in \mathbb{Z}[x_1, \ldots, x_k]$ such that all small modular roots of the f_i are *integer* roots of the h_i. The integer roots of the h_i can then be recovered from the Gröbner-basis of the ideal $\mathfrak{a} = (h_1, \ldots, h_k) \subseteq \overline{\mathbb{Q}}[x_1, \ldots, x_k]$. Unfortunately, this step only works when the ideal is zero-dimensional. Therefore, the multivariate variants of Coppersmith's method rely on the following heuristic.

Heuristic 2.4 (Coppersmith Heuristic)**.** *The polynomials obtained from Coppersmith's method generate an ideal of a zero-dimensional variety.*

To facilitate the process of using and optimizing Coppersmith's method, the authors of [23] propose an automated variant that replaces the lattice theory with combinatorial constraints. Contrary to previous works, the authors do not use an ad-hoc approach to construct the polynomials $h_1, \ldots, h_k$. Instead, their approach relies on choosing a set $\mathcal{M}$ of monomials from which the h_i are derived.

Definition 2.5. *Let $\mathcal{M}$ be a finite set of monomials, and let $\prec$ be a monomial order on $\mathcal{M}$. A set of polynomials $\mathcal{F}$ is called* $(\mathcal{M}, \prec)$-suitable*, if:*

1. *Every $f \in \mathcal{F}$ is defined over $\mathcal{M}$.*
2. *For every monomial $\lambda \in \mathcal{M}$ there is a unique polynomial $f \in \mathcal{F}$ with leading monomial λ (with respect to $\prec$).*

If $\mathcal{F}$ is $(\mathcal{M}, \prec)$-suitable and $\lambda \in \mathcal{M}$, then we denote by $\mathcal{F}[\lambda]$ the unique polynomial $f \in \mathcal{F}$ with leading monomial λ.

In their framework, Coppersmith's method can now be reformulated as follows.

Theorem 2.6 (Coppersmith's Method). *Suppose we are given a modulus $N \in \mathbb{N}$, polynomials $f_1, \ldots, f_n \in \mathbb{Z}_N[x_1, \ldots, x_k]$ and bounds $0 \leq X_1, \ldots, X_k \leq N$, where $k = \mathcal{O}(1)$. Furthermore, suppose we are given an integer $m \in \mathbb{N}$, a set of monomials $\mathcal{M}$, a monomial order $\prec$ on $\mathcal{M}$, and an $(\mathcal{M}, \prec)$-suitable set of polynomials $\mathcal{F} \subseteq \mathbb{Z}_{N^m}[x_1, \ldots, x_k]$ with*

$$Z_{N,X_1,\ldots,X_k}(f_1, \ldots, f_n) \subseteq Z_{N^m,X_1,\ldots,X_k}(\mathcal{F}). \tag{1}$$

If the conditions

$$\prod_{\lambda \in \mathcal{M}} |\mathsf{LC}(\mathcal{F}[\lambda])| \leq \frac{N^{(m_i-k)|\mathcal{M}|}}{\prod_{\lambda \in \mathcal{M}} \lambda(X_1, \ldots, X_k)}, \tag{2}$$

Algorithm 1: Constructing an optimal set $\mathcal{F}$.

Input: Set of monomials $\mathcal{M}$, monomial order $\prec$ on $\mathcal{M}$, monic polynomials $f_1, \ldots, f_n$, and integer $m \in \mathbb{N}$.

Output: $(\mathcal{M}, \prec)$-suitable set $\mathcal{F}$, satisfying Equation (1), and minimizing the left hand side in Equation (2).

```
F := ∅
for λ ∈ M do
    Enumerate all polynomials f_[λ,i_1,...,i_n] as in Equation (3) such that
      LM(f)^{i_1} · ... · LM(f)^{i_n} divides λ and f_[λ,i_1,...,i_n] is defined over M.
    Among all such f_[λ,i_1,...,i_n] pick one that maximizes i_1 + ... + i_n and
      include it in F.
end
return F
```

$\log(N) \geq |\mathcal{M}| \geq m$ *and* $|\mathcal{M}| \geq k$ *hold, then we can compute all*

$$r \in Z_{N,X_1,\ldots,X_k}(f_1, \ldots, f_n)$$

in time polynomial in $\deg(\mathcal{F}) \cdot \log(N)$*, under Heuristic 2.4 for* $k > 1$*.*

Given a set of monomials $\mathcal{M}$, constructing an *optimal* $(\mathcal{M}, \prec)$-suitable set $\mathcal{F}$ satisfying all the conditions in Theorem 2.6 can be done automatically via Algorithm 1 and so-called *shift polynomials* of the form

$$f_{[\lambda,i_1,\ldots,i_n]} := \frac{\lambda \cdot f_1^{i_1} \cdot \ldots \cdot f_n^{i_n} \cdot M^{m-(i_1+\ldots+i_n)}}{\mathsf{LM}(f_1)^{i_1} \cdot \ldots \cdot \mathsf{LM}(f_n)^{i_n}}. \tag{3}$$

Furthermore, for any set $f_1, \ldots, f_n$ of modular polynomial the authors in [23] provide a natural choice for $\mathcal{M}$, thus fully automating the entire process. Their choice for $\mathcal{M}$ might however not always yield optimal results.

Lastly, for any set $f_1, \ldots, f_n$ of modular polynomials the framework of [23] can be used to automatically derive asymptotic bounds $X_1, \ldots, X_k$ for which Coppersmith's method is successful. To this end, the authors define a sequence $\mathcal{M}_1 \subset \mathcal{M}_2 \subset \ldots$ of sets of monomials and compute the corresponding (optimal) $(\mathcal{M}_i, \prec)$-suitable $\mathcal{F}_i$. For each pair $(\mathcal{M}_i, \mathcal{F}_i)$ they then evaluate Equation (2) and use polynomial interpolation to derive asymptotic bounds on $X_1, \ldots, X_k$. The derivation relies on the following heuristic, which seems to always hold in practice.

Heuristic 2.7. *Let* $f_1, \ldots, f_n \in \mathbb{Z}[x_1, \ldots, x_k]$*, let* $\prec$ *be a monomial order on* $x_1, \ldots, x_k$*, and define*

$$\mathcal{M}_i := \left\{ \lambda \mid \lambda \text{ is a monomial of } f_1^{j_1} \cdot \ldots \cdot f_n^{j_n}, 0 \leq j_1, \ldots, j_n \leq i \right\}$$
$$m_i := i \cdot n,$$

for $i \in \mathbb{N}$*. Then there exists a polynomial* $p(m)$ *of degree* $k+1$*, such that for any set* $\mathcal{F}_i$*, that is obtained from Algorithm 1 on input* $(\mathcal{M}_i, \prec, (f_1, \ldots, f_n), m_i)$*, it*

holds that

$$\prod_{\lambda \in \mathcal{M}_i} |\mathsf{LC}(\mathcal{F}_i[\lambda])| = N^{p(m_i)}.$$

3 The Leveled Isogeny Problem with Hints

Let $\varphi : E \to E'$ be an isogeny of (potentially non-smooth) degree. Using higher-dimensional isogeny techniques [31], it is possible to represent (and subsequently evaluate) the isogeny φ via the tuple $(P, Q, \varphi(P), \varphi(Q))$ for some accessible basis (P, Q) of $E[N]$ with $N \geq \sqrt{\deg(\varphi)}$ smooth.

Since revealing such information (e.g. as part of a public key in a key exchange) proves to be fatal, protocols usually mask the torsion point information by multiplying $(\varphi(P), \varphi(Q))$ with a secret matrix Γ. Notably, the shape of Γ varies across schemes, and the security of each scheme crucially relies on the secrecy of Γ.

In this work we analyze the situation in which partial information about Γ is leaked, which we refer to as *hints*.

Definition 3.1 (Binary Hint). *Let $x \in \mathbb{N}$ be an n-bit integer and $x[i]$ its i-th bit.*[6] *A* k-bit hint for x *is a set $\mathbb{H}_{\mathcal{J}}(x) \coloneqq \{x[j] : j \in \mathcal{J}\}$ where $\mathcal{J} \subseteq \{1, \ldots, n\}$ is an index set of cardinality k.*

Definition 3.1 extends entry-wise to tuples and matrices. To simplify notation we may drop the index $\mathcal{J}$ and simply write $\mathbb{H}(x)$. Furthermore, we define the following explicit *types* of hints:

- *Most Significant Bits*: $\mathbb{M}_k(x) \coloneqq \mathbb{H}(x)$ with $\mathcal{J} = \{1, \ldots, k\}$.
- *Least Significant Bits*: $\mathbb{L}_k(x) \coloneqq \mathbb{H}(x)$ with $\mathcal{J} = \{n - k + 1, \ldots, n\}$.
- *Random Bits*: $\mathbb{R}_k(x) \coloneqq \mathbb{H}(x)$ with random $\mathcal{J} \subseteq \{1, \ldots, n\}$ of size k.
- *Continuous Block*: $\mathbb{C}_{s,k}(x) \coloneqq \mathbb{H}(x)$ with $\mathcal{J} = \{s + i \bmod n : 0 \leq i < k\}$.

We are now ready to state a generalization of the isogeny problem with level structure [12, Definition 1].

Definition 3.2 (Leveled Isogeny Problem with Hints (LIPH)). *Let $\varphi : E \to E'$ be an isogeny of (potentially unknown) degree between two supersingular curves defined over $\mathbb{F}_{p^2}$. Furthermore, let $(P, Q) \in E[N]$ be accessible torsion points and $(P', Q') = \Gamma \cdot (\varphi(P), \varphi(Q))^\top$ for some smooth $N \in \mathbb{N}$ and $\Gamma \in \mathrm{GL}(N, 2)$. Given (E, E', P, Q, P', Q') and a k-bit hint $\mathbb{H}(\Gamma)$, recover an efficient representation of φ.*

Remark 3.3. Instances of LIPH that appear in the literature are restricted to matrices Γ that are either *diagonal* or *circulant* since it is usually required that Γ commutes with another matrix.

[6] We use big-endian notation, i.e. $x[1]$ represents the MSB.

As mentioned earlier, by using the Weil pairing any instance of LIPH gives rise to a modular equation of the form

$$\deg(\varphi) \cdot \det(\Gamma) - r \equiv 0 \bmod N \tag{4}$$

for some known $r \in \mathbb{Z}_N$ and (potentially) unknown $\deg(\varphi)$ and Γ. This suggests the following strategy to solve LIPH: use the hint $\mathbb{H}(\Gamma)$ to solve Equation (4), recovering $\deg(\varphi)$ as well as the four entries of Γ. Once recovered, the LIPH instance is effectively turned into an unmasked SIDH instance and the isogeny φ can be recovered in polynomial time via the standard SIDH attack [7,21,30]. Note that, in the case where the LIPH instance arises from the public key of a key exchange, there is enough available torsion for the SIDH attacks due to the correctness of the key exchange.

The hardness of LIPH mainly depends on k, Γ and the type of $\mathbb{H}(\Gamma)$. Indeed, if no hint is provided, LIPH reduces to the generic isogeny problem, which is believed to be intractable unless we assume Γ to be of a specific shape (cf. [12, Figure 1]). On the other hand, if Γ is for instance the identity matrix or $\mathbb{H}(\Gamma)$ contains all the bits of Γ, then LIPH is already an unmasked SIDH problem. Furthermore, using the meet-in-the-middle strategy of [31] some instances of LIPH can also be solved efficiently for LSB hints $\mathbb{L}(\Gamma)$.

Proposition 3.4. *Let $N = 2^{2k}$, let $q < N$ be the (unknown) degree of the secret isogeny φ and let $\Gamma = \left(\begin{smallmatrix} a_1 & a_2 \\ a_3 & a_4 \end{smallmatrix}\right)$. There exist an efficient adversary against the Leveled Isogeny Problem with Hints where $\mathbb{L}_{4k}(\Gamma) = (\mathbb{L}_k(a_1), \ldots, \mathbb{L}_k(a_4))$.*

Proof. Upon receiving (E, E', P, Q, P', Q'), the adversary computes the points $(R, S) = ([2^k]P, [2^k]Q)$ and $(R', S') = ([2^k]P', [2^k]Q')$. We then have

$$(R, S) = \Gamma' \cdot (\varphi(R), \varphi(S))^\top, \qquad \Gamma' \equiv \Gamma \bmod 2^k$$

for known Γ' since $\mathbb{L}_k(\Gamma)$ contains the lower half of every entry in Γ. Furthermore, the adversary can recover $q \bmod 2^k$ via Equation (4), resulting in an unmasked instance of the SIDH-problem. The latter can be solved via the meet-in-the-middle strategy in [31, Appendix B], which requires torsion point images of order only $\sqrt{q} < \sqrt{N} = 2^k$ to represent φ. □

Proposition 3.4 shows that for modulus N a power of two, only the least-significant bits of Γ are required to solve LIPH, irrespective of the shape of Γ or the degree q (as long as it is sufficiently small). Therefore, any attempt at solving LIPH in this scenario only needs to recover sufficiently many LSBs of (the entries of) Γ without completely recovering every entry.

Unfortunately, this reasoning does not apply to an *odd* modulus as it is not possible to infer Γ' from $\mathbb{L}(\Gamma)$ such that $\Gamma' \equiv \Gamma \bmod N'$ for some divisor $N' \mid N$. Indeed, from $\mathbb{L}_{4k}(\Gamma)$ as defined above we can infer $a_i = a_i' \cdot 2^k + x$, but it is still not possible to know $a_i' \cdot 2^k + x \bmod N'$ as the modular reduction depends on a_i'. Note that this dependence is absent if N is a power of two and $N' \leq 2^k$ as $a_i' \cdot 2^k \equiv 0 \mod N'$. Therefore, an odd modulus N requires different techniques, which is a theme that continues in the next sections.

4 The Case $\Gamma = \left(\begin{smallmatrix} \alpha & \\ & \alpha \end{smallmatrix}\right)$ (a.k.a. M-SIDH, POKÉ)

In this section we consider the case where both torsion points are scaled by the same (unknown) scalar, which appears in the context of M-SIDH and (compressed, 4-dimensional variants of) POKÉ. Notably, the two schemes differ by the fact that in M-SIDH the degree q of the secret isogeny is known whereas in POKÉ it remains secret. Thus, in the case of M-SIDH Equation (4) contains one variable less.

4.1 Known Degree

To counteract the SIDH attacks, M-SIDH [14] tweaks the original SIDH key exchange by introducing a single masking scalar for both torsion points. Concretely, for a prime $p = ABf - 1$ with smooth coprime $A, B \in \mathbb{N}$, an M-SIDH public key is the tuple $(E', [\alpha]\varphi(P), [\alpha]\varphi(Q))$ where $\varphi : E_0 \to E'$ is the secret key isogeny of known degree, (P, Q) is a deterministic basis of $E_0[B]$ and $\alpha \in \mathbb{Z}_B^\times$ such that $\alpha^2 \equiv 1 \bmod B$. Hence, leakage on α together with the public key give rise to a LIPH instance where $\Gamma = \left(\begin{smallmatrix} \alpha & \\ & \alpha \end{smallmatrix}\right)$, which can be solved via the following, well-known theorem.

Theorem 4.1 (Coppersmith [9]). *Given a modulus N, a univariate monic polynomial $f(x)$ of constant degree δ and a bound $X \in \mathbb{N}$. If*

$$X < N^{1/\delta}$$

we can compute all $r \in Z_{N,X}(f)$ in time polynomial in $\log N$ and δ.

The next result immediately follows from Theorem 4.1.

Proposition 4.2. *Let $\varphi : E_0 \to E'$ be an isogeny of known degree and $\alpha \in \mathbb{Z}_B^\times$ as described above. There exists an efficient adversary against LIPH where $\Gamma = \left(\begin{smallmatrix} \alpha & \\ & \alpha \end{smallmatrix}\right)$ and $\mathbb{M}_k(\Gamma) = \mathbb{M}_k(\alpha)$ with $k = \lceil (\log B)/2 \rceil$.*

Proof. Combining the LIPH instance $(E_0, E', P, Q, [\alpha]\varphi(P), [\alpha]\varphi(Q))$ with Equation (4) yields the modular relation

$$\alpha^2 - 1 \equiv 0 \bmod B,$$

where $(P, Q) \in E_0[B]$. Using the MSB hint $\mathbb{M}_k(\alpha)$ with $k = (\log B)/2$ we can express the scalar as $\alpha = \alpha' + x$ for some known $\alpha' \geq \sqrt{B}$ and small unknown $x \in \mathbb{N}$, yielding the monic equation

$$(\alpha' + x)^2 - 1 \equiv 0 \bmod B.$$

Since $x < \sqrt{B}$ we can now recover the least-significant bits of α via Theorem 4.1, resulting in an unmasked instance of the SIDH problem. The isogeny φ can then be efficiently represented as explained in Section 3. □

4.2 Unknown Degree

POKÉ [2] combines rational and two-dimensional isogenies to build an efficient and compact key exchange. Here, two-dimensional isogenies are used to represent the secret isogeny. More specifically, for a prime $p = 2^a 3^b 5^c - 1$ and secret isogeny $\varphi : E_0 \to E'$ of unknown degree $q(2^a - q)$, the public key is the tuple

$$(E', [\alpha]\varphi(P), [\beta]\varphi(Q), [\gamma]\varphi(R), [\gamma]\varphi(S))$$

for deterministic bases (P, Q) of $E_0[2^a]$ and (R, S) of $E_0[3^b 5^c]$. Although the public key contains two potential LIPH instances with $\Gamma = \left(\begin{smallmatrix} \alpha & \\ & \beta \end{smallmatrix}\right)$ and $\Gamma = \left(\begin{smallmatrix} \gamma & \\ & \gamma \end{smallmatrix}\right)$, unfortunately both moduli 2^a ($\approx 2^\lambda$) and $3^b 5^c$ ($\approx 2^{7\lambda/3}$) are to small to allow for an efficient solution of Equation (4), even for very large hints (here λ refers to the security parameter of the scheme).

Instead, we will focus on an optimization technique proposed in [2, Section 4.4] where four-dimensional isogenies are used instead of the faster, but also less flexible two-dimensional counterparts. Subsequently, the degree of φ now is an unknown integer $q \in \mathbb{Z}_{2^a}$ and the public key is simply $(E', [\alpha]\varphi(P), [\alpha]\varphi(Q))$ for some deterministic basis (P, Q) of $E_0[2^a 3^b 5^c]$ and $\alpha \in \mathbb{Z}^\times_{2^a 3^b 5^c}$. Furthermore, $2^a = 2^{\lambda/2}$ is sufficient to represent the secret isogeny, leading to $N = 2^{17\lambda/6}$ and an overall more compact scheme.

It is easy to see that, given a hint on α, the compressed public key gives rise to an instance of LIPH with $\Gamma = \left(\begin{smallmatrix} \alpha & \\ & \alpha \end{smallmatrix}\right)$ and unknown degree q. Contrary to the uncompressed variant, it turns out that such instances are efficiently solvable when given a sufficiently large MSB hint.

Theorem 4.3. *Let $\nu = \log N$ and $\varphi : E_0 \to E'$ be an isogeny of unknown degree $q < N^{6/17}$. Under Heuristics 2.4 and 2.7 there exists an efficient adversary against LIPH where $\Gamma = \left(\begin{smallmatrix} \alpha & \\ & \alpha \end{smallmatrix}\right)$ and $\mathbb{M}_k(\Gamma) = \mathbb{M}_k(\alpha)$ with $k = \lceil 29\nu/34 \rceil$.*

The proof of Theorem 4.3 relies on the following lemma, which is proven first.

Lemma 4.4. *Given a modulus N, a polynomial $f(x, y) = (x^2 + f_1 x + f_2) \cdot y + f_3$ for some constants $f_i \in \mathbb{N}$, bounds $X, Y \in \mathbb{N}$ and an arbitrarily small constant $\epsilon > 0$. If N is sufficiently large and*

$$XY \leq N^{1/2-\epsilon},$$

then under Heuristics 2.4 and 2.7 we can compute all $r \in Z_{N,X,Y}(f)$ in time polynomial in $\log(N)$.

Proof. We use the strategy outlined in Section 2.3. To this end, we define

$$\begin{aligned} \mathcal{M}_i &\coloneqq \{\lambda \mid \lambda \text{ is a monomial of } f^j, 0 \leq j \leq i\} \\ m_i &\coloneqq 2 \cdot i \end{aligned}$$

for $i \in \mathbb{N}$, which satisfy the conditions $\mathcal{M}_i \geq m_i$ and $\mathcal{M}_i \geq 2$ from Theorem 2.6. Furthermore, if N is sufficiently large then $\log(N) \geq |\mathcal{M}_i|$ is also satisfied.

We now have that

$$N^{(m-2)|\mathcal{M}_i|} = N^{p_{\mathcal{M}}(m_i)}, \qquad \prod_{\lambda \in \mathcal{M}_i} \lambda(X, Y) = X^{p_X(m_i)} \cdot Y^{p_Y(m_i)}$$

for some polynomials $p_{\mathcal{M}}$, p_X, p_Y due to the definition of $\mathcal{M}_i$. Similarly, under Heuristic 2.7 we have

$$\prod_{\lambda \in \mathcal{M}_i} |\mathsf{LC}(\mathcal{F}_i[\lambda])| = N^{p_{\mathcal{F}}(m_i)}$$

for some polynomial $p_{\mathcal{F}}$, where $\mathcal{F}_i$ is the output of Algorithm 1 on input $(\mathcal{M}_i, \prec_{\mathsf{lex}}, f, m_i)$. We ran Algorithm 1 for $i = 1, \ldots, 5$, obtaining the following values:

m_i	$p_{\mathcal{M}}(m_i)$	$p_{\mathcal{F}}(m_i)$	$p_X(m_i)$	$p_Y(m_i)$
2	0	7	3	3
4	18	31	13	13
6	64	82	34	34
8	150	170	70	70
10	288	305	125	125

Polynomial interpolation yields

$$p_{\mathcal{M}} = \frac{1}{4}m_i^3 + o(m_i^3), \quad p_{\mathcal{F}} = \frac{5}{24}m_i^3 + o(m_i^3), \quad p_X = p_Y = \frac{1}{12}m_i^3 + o(m_i^3)$$

and thus Equation (2) simplifies to

$$X^{1/12}Y^{1/12} \leq N^{1/4-5/24-\epsilon} = N^{1/24-\epsilon}$$

for some ϵ that vanishes when m_i increases. The result follows by raising both sides to the 12th power. □

Proof of Theorem 4.3. From Equation (4) it follows that the LIPH instance defines the equation

$$\alpha^2 q - r \equiv 0 \bmod N$$

for some known constant $r \in \mathbb{Z}_N$. Using the provided hint $\mathbb{M}_k(\alpha)$, an adversary knows the $k = 29\nu/34$ most-significant bits of α. Hence we may write $\alpha = \alpha' + x$ for some unknown $x \leq N^{5/34}$ and $q = y$ for some unknown $y \leq N^{6/17}$, yielding the equation

$$(\alpha' + x)^2 \cdot y - r \equiv 0 \bmod N$$

via substitution. Since $x \cdot y \leq N^{5/34+6/17} \leq N^{1/2}$, this modular equation can be solved in polynomial time via Lemma 4.4. Once α and q are recovered, the adversary easily obtains a representation for the secret isogeny φ as explained in Section 3. □

Theorem 4.3 connects back to POKÉ by observing that all required conditions are satisfied by the parameters of the four-dimensional POKÉ variant. In particular, $q \leq N^{6/17}$ due to the fact that $q \leq 2^\lambda$ and the parameter choice $N = 2^{17\lambda/6}$ made in [2, Section 4.4].

5 The Case $\Gamma = \left(\begin{smallmatrix} \alpha & \\ & \beta \end{smallmatrix}\right)$ (a.k.a. FESTA)

We now turn to the case where both torsion points are scaled by different scalars α, β, which appears in the FESTA trapdoor function [4]. In more detail, FESTA uses a prime $p = 2^b f - 1$, a secret isogeny $\varphi : E_0 \to E'$ of known degree and public keys of the form $(E', [\alpha]\varphi(P), [\beta]\varphi(Q))$ where (P, Q) is a deterministic basis of $E_0[2^b]$. Additionally, it is required that $\alpha\beta \equiv 1 \bmod 2^b$.

As before, a hint on Γ yields an instance of LIPH. Depending on the type of hint, however, the instance can either be solved by a dedicated combinatorial approach or generically by Coppersmith's method.

5.1 Random Hints

In this section we will approach the solution to the equation $\alpha\beta \equiv 1 \bmod 2^b$, assuming random bit leaks on both α and β. By random bit leaks we mean that each bit of α and β has the same probability q of being leaked to an attacker. This concretely rules out corner cases such as knowing the half most (or least) significant bits, or in general the knowledge of big blocks of consecutive bits, making Coppersmith-based approaches hard to adapt. Our attacks in this settings have complexity $O(2^{\sqrt{b}})$, which is subexponential and is asymptotically significantly faster than the best known key recovery attacks which are exponential.

In general, we can formulate the problem as follows. Let

$$\alpha = \sum_{i=0}^{b} \alpha_i 2^i, \quad \beta = \sum_{i=0}^{b} \beta_i 2^i$$

two numbers such that $\alpha\beta \equiv 1 \bmod 2^b$. As a leakage information, we receive a fixed portion of the coefficients $\alpha_i, \beta_i \in \{0, 1\}$. Let assume that $q = 1/2$, so we get half of the bits for both α and β.

A first naive approach is to try to substitute all possible combinations of 0 and 1 in all the unknown bits. Since we expect to still have b unknown bits ($b/2$ from α and $b/2$ from β), we will need to try 2^b combinations. Moreover, this method does not give us any insight on how many valid combinations we expect to find at the end, information that may be relevant if the solutions need to be checked afterwards.

A more structured approach consist in solving the equation mod 2^k for increasing values of k. Notice that $\alpha\beta \equiv 1 \bmod 2^b$ already implies $\alpha_0 = \beta_0 = 1$, so the pair $(1, 1)$ will be the only valid solution for $k = 1$. For $k = 2$, we will need to solve

$$(2\alpha_1 + 1)(2\beta_1 + 1) \equiv 1 \bmod 2^2$$

which may have one or two possible solutions, depending on whether the value of α_1 and β_1 is known or not. For each solution found, we can do the same reasoning for $k = 3$, and all the way up to $k = b$. More precisely, let Q_k be a list of pairs (x, y) such that $xy \equiv 1 \bmod 2^k$, and x and y match with the known

bits of α and β respectively up to 2^k. For each such pair (x, y), we look at the equation $\bmod 2^{k+1}$ where we obtain

$$(x + \alpha_k 2^k)(y + \beta_k 2^k) - 1 \equiv xy + 2^k(\alpha_k y + \beta_k x) - 1 \bmod 2^{k+1}.$$

But since (x, y) is a solution $\bmod 2^k$, the entire equation is divisible by 2^k, and we have to solve

$$\frac{xy - 1}{2^k} \equiv (\alpha_k + \beta_k) \bmod 2.$$

Here the left hand side is uniformly random on $\{0, 1\}$, and we are left with three possibilities:

- α_k, β_k are unknown; for a fixed pair (x, y) at step k, we then have two possible pairs at step $k+1$, given by the two pairs of α_k and β_k matching the required value;
- one among α_k, β_k is known; the left hand side fixes the value of the other coefficient, and from a solution we generate exactly a new one;
- both α_k and β_k are known; on average, half of the time the left hand side will match the right hand side, and we keep our solution; the other half of the times, the two sides will not match, and we will discard the solution.

In the first case, $|Q_{k+1}| = 2|Q_k|$, while in the second one $|Q_{k+1}| = |Q_k|$. In the third case, we expect on average $|Q_{k+1}| = |Q_k|/2$. While this behavior is only heuristic, especially for big size of Q_k it will be quite accurate. However we are always sure of the existence of at least one solution; so if $|Q_k| = 1$, and we are in the third case, we will still have $|Q_{k+1}| = 1$. We can hence make the following heuristic assumption.

Heuristic 5.1. *The known bits α_k of α and β_k of β are distributed uniformly at random, i.e. the three cases described above occur with the same probability. Moreover, if at a step k both α_k and β_k are known, the number of solutions is cut in half unless it is already one, i.e. $|Q_k| = \max\{|Q_{k-1}|/2, 1\}$.*

Statistical analysis Let $S_k = \mathbb{E}[|Q_k|]$ be the expected size of the list of solutions after k steps. Then the average case complexity of our algorithm is bounded by $O(bS_b)$, S_b being the number of queries we have to process at the last step.

We define the family of random variables $\{J_k\}_k$ by $J_0 = 1$ and

$$\mathbb{P}(J_k = l | J_{k-1} = m) = \begin{cases} 1/2 & \text{if} \quad m = l \\ 1/4 & \text{if} \quad m = l + 1 \\ 1/4 & \text{if} \quad m = l - 1 \vee 0. \end{cases}$$

for $k \geq 1$.

Lemma 5.2. *For every k it holds $S_k = 2^{\mathbb{E}[J_k]}$.*

Proof. The statement is trivial for $k = 0$. At every step, J_k increases or decreases by one with the same probability of Q_k doubling or halving its size (subject to Heuristic 5.1). The two distributions are hence the same. □

We now want to estimate the expected value $\mathbb{E}(J_k)$. The most simple way to do it is to see J_k as the result of the following coin game: we start the game with 1 coin, and at each turn we throw a dice with 4 sides, a, b, c and d. Then

- if the outcome is a or b, we keep our coins unchanged
- if the outcome is c, we gain one coin
- if the outcome is d, we lose one coin unless we only have one coin; if we have one coin on d, nothing happens.

Let G_k be the set of all possible outcomes (i.e. the number of coins we have) after k steps. We have

$$G_0 = \{1\}$$
$$G_1 = \{1, 1, 1, 2\}$$
$$G_2 = \{1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 2, 1, 2, 2, 3\}$$

and so on.

Lemma 5.3. *Let $\sum G_k$ be the sum of all elements in G_k. Then $\sum G_k = 4^k \mathbb{E}[J_k]$.*

Proof. By frequentist approach, we can define $\mathbb{E}[J_k]$ as the sum of all possible outcomes for J_k, multiplied by their respective probabilities. If we see the event $J_{k+1} = J_k$ as two events, each one with probability $1/4$, then the possible outcomes for J_k are exactly the possible outcome of the coin game, i.e. the elements of G_k. The total number of outcomes (or equivalently, the number of elements of G_k) is 4^k. Hence $\mathbb{E}[J_k] = \sum G_k / 4^k$. □

Theorem 5.4. *Let $G_{k,n} = \#\{\text{copies of } \ n \in G_k\}$. Then*

$$G_{k,n} = \binom{2k+1}{k+n}. \tag{5}$$

Proof. We prove the statement by induction. The cases $k = 0, 1$ can be proven directly by looking at the sets above and computing

$$G_{0,1} = \binom{1}{1} = 1, \quad G_{1,1} = \binom{3}{2} = 2, \quad \binom{3}{3} = 1.$$

Notice that by definition of the problem it always holds $1 \leq n \leq k+1$. We can now assume that Equation (5) holds for $k-1$, and prove it for k. First we prove the statement for $n = 1$ and then for $n > 1$. For $n = 1$, one has that

$$G_{k,1} = 3 \cdot G_{k-1,1} + G_{k-1,2}.$$

By the induction hypotheses we have that

$$G_{k,1} = 3\binom{2k-1}{k} + \binom{2k-1}{k+1} = \frac{(2k-1)!(3 \cdot (k+1) + (k-1))}{(k+1)!(k-1)!} =$$

$$= \frac{(2k-1)!(4k+2)}{(k+1)!(k-1)!} = \frac{(2k-1)!(4k+2)}{(k+1)!(k-1)!} = \frac{4k+2}{k+1}\binom{2k-1}{k-1} = \binom{2k+1}{k+1}.$$

For $n > 1$ instead one has that

$$G_{k,n} = 2 \cdot G_{k-1,n} + G_{k-1,n-1} + G_{k-1,n+1}.$$

Again using the induction hypothesis we get

$$G_{k,n} = 2\binom{2k-1}{k+n-1} + \binom{2k-1}{k+n-2} + \binom{2k-1}{k+n} =$$

$$\frac{(2k-1)!(2(k+n)(k-n+1) + (k+n-1)(k+n) + (k-n)(k-n+1))}{(k+n)!(k-n+1)!} =$$

$$\frac{(2k-1)! \cdot 2k \cdot (2k+1)}{(k+n)!(k-n+1)!} = \binom{2k+1}{k+n}$$

□

We are now ready to compute $\sum G_k$.

Theorem 5.5. *It holds*

$$\sum G_k = \frac{1}{2}\left(4^k + \frac{(2k+1)!}{k!^2}\right). \tag{6}$$

Proof. For $k = 0$, $\sum G_k = 1$ and Equation (6) can be tested directly. We then proceed by induction. By looking at the coin game from before, we see that each value $n \neq 1$ in G_k produces the set $\{n-1, n, n, n+1\}$ in G_{k+1} for a total sum of $4n$. On the other hand, for each $n = 1$ in G_k we have the set $\{1, 1, 1, 2\}$ in G_{k+1} with a sum of 5. We thus have the recurrence relation

$$\sum G_{k+1} = 4\sum G_k + G_{k,1}$$

where we recall that $G_{k,1}$ is the amount of 1s in G_k. By using the induction hypothesis on $\sum G_k$ and Equation (5) for $G_{k,1}$ we have

$$\sum G_{k+1} = 4\sum G_k + G_{k,1} = \frac{4}{2}\left(4^k + \frac{(2k+1)!}{k!^2}\right) + \binom{2k+1}{k+1} =$$

$$= \frac{1}{2}\left(4^{k+1} + 4\frac{(2k+1)!}{k!^2} + 2\frac{(2k+1)!}{(k+1)!k!}\right) =$$

$$= \frac{1}{2}\left(4^{k+1} + \frac{(2k+1)!(4(k+1)^2 + 2(k+1))}{(k+1)!^2}\right) =$$

$$= \frac{1}{2}\left(4^{k+1} + \frac{(2k+3)!}{(k+1)!^2}\right)$$

where in the last step we used that $(2k+2)(2k+3) = 4(k+1)^2 + 2(k+1)$. □

Remark 5.6. As a byproduct of Theorem 5.5 we obtain the identity

$$\sum_{n=1}^{k+1} n\binom{2k+1}{k+n} = \sum G_k = \frac{1}{2}\left(4^k + \frac{(2k+1)!}{k!^2}\right).$$

Theorem 5.7. *Under Heuristic 5.1,* $S_k = 2^{O(\sqrt{k})}$.

Proof. By Lemma 5.2 and Lemma 5.3, it follows $S_k = 2^{\sum G_k/4^k}$. Applying the Stirling approximation formula [28] to Equation (6) we get

$$\frac{\sum G_k}{4^k} = \frac{1}{2\cdot 4^k}\left(4^k + \frac{(2k+1)!}{(k!)^2}\right) \approx$$

$$\frac{1}{2} + \frac{1}{2\cdot 4^k}\left(\frac{\sqrt{2\pi(2k+1)}\left(\frac{2k+1}{e}\right)^{2k+1}}{2\pi k\left(\frac{k}{e}\right)^{2k}}\right) \approx \tag{7}$$

$$\frac{1}{2} + \frac{2\sqrt{\pi k}(2k)^{2k+1}e^{2k}}{4^{k+1}\pi(ek)^{2k+1}} = \frac{1}{2} + \frac{1}{e\sqrt{\pi}}\sqrt{k} = O(\sqrt{k}).$$

□

Theorem 5.7 in particular implies the following:

Proposition 5.8. *Let* $\varphi : E_0 \to E'$ *be an isogeny of known degree and* $\alpha, \beta \in \mathbb{Z}_{2^b}$ *with* $\alpha\beta \equiv 1 \bmod 2^b$. *There exists an adversary against LIPH with* $\Gamma = \begin{pmatrix} \alpha & \\ & \beta \end{pmatrix}$, $\mathbb{R}_{2k}(\Gamma) = (\mathbb{R}_k(\alpha), \mathbb{R}_k(\beta))$ *and* $k = \lceil b/2 \rceil$ *that has time complexity* $\mathcal{O}(2^{\sqrt{b}})$.

In Figure 1 we can see the average size of Q_k after a 100 runs, for different values of k. Each dot correspond to a run, and the red line indicates the expected distribution of S_k.

Remark 5.9. This algorithm is completely parallelizable. At any step we can take Q_k and simply split it in the number of cores available.

A concrete example: practical key recovery on FESTA The subexponential complexity of this attack makes it very practical for a range of sizes that are very common in isogeny based cryptography. In particular, we can recover the secret key of a NIST Level 1 FESTA instance in a matter of seconds in sage, when given leakage of half of the bits of α, β. In this section we follow the notation from [4].

The secret key is an isogeny φ_A of degree $A = d_{A,1}d_{A_2}$, where

$$d_{A,1} = (59 \cdot 6299 \cdot 6719 \cdot 9181)^2$$

and

$$d_{A_2} = (3023 \cdot 3359 \cdot 4409 \cdot 5039 \cdot 19531 \cdot 22679 \cdot 41161)^2$$

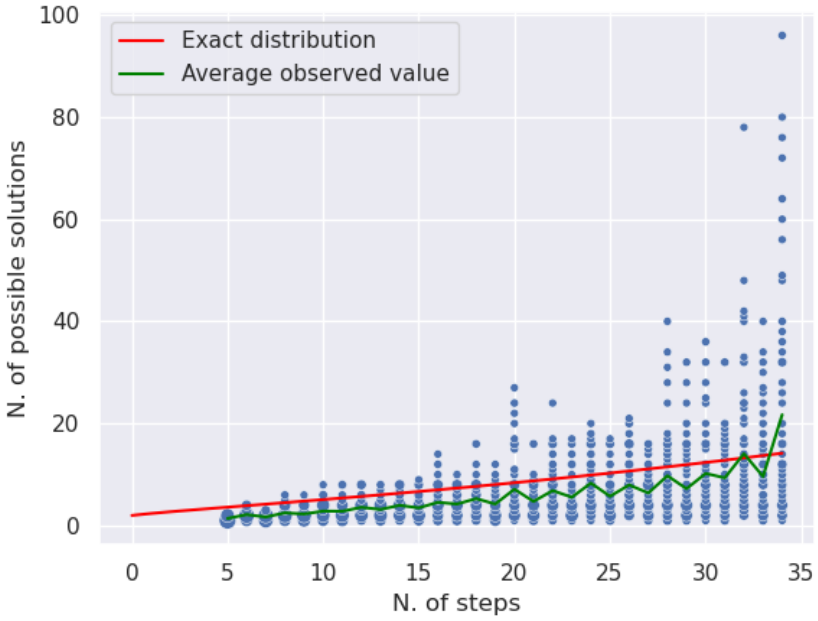

Fig. 1. Average size of Q_k for growing k

for a total of 274 bits. Let (P_b, Q_b) a basis of $E_0[2^b]$ with $b = 632$. We are also given $R_A = \alpha\varphi_A(P_b)$ and $S_A = \beta\varphi_B(Q_b)$. The first observation is that it is enough to recover $\alpha, \beta \bmod 2^c$ for any c such that $2^{2c} > 4d_A$. This is true because from R_A we can recover

$$2^{b-c}R_A = 2^{b-c}\alpha\varphi(P_b) = (\alpha \bmod 2^c)\varphi(2^{b-c}P_b)$$

and the same for S_A. Then the image of the 2^c torsion uniquely characterize φ_A, which can be recovered in polynomial time using higher dimensional isogenies. This would allow values of c of roughly 140 bits, for which the attack described in the previous section definitely applies. However, the need (in general) of 4 or 8 dimensional isogenies maks this approach not fully practical.

What we can do instead is to try to solve the equation

$$un_1 + vn_2 = 2^t, \tag{8}$$

where $n_1 n_2 = d_A$. As shown in, [27], if we can do that and we have access to a u-isogeny on E_0 and a v-isogeny on E_A (the public curve) we can actually recover φ_A given its interpolation data on the 2^t torsion. In this case, E_0 is the curve $y^2 = x^3 + 6x^2 + x$, of which we know the endomorphism ring (since it is 2-isogenous to the curve with $j = 1728$). This means that, using the method

from [26], we can compute isogenies of any degree u from E_0. We can hence focus on solving Equation (8) for v smooth, in order to compute a v-isogeny from E_A. A possible solution is given by

$$n_1 = 3481$$
$$n_2 = \texttt{0x160f1fdedc055ce30ae140a0eb15f28}$$
$$\texttt{192c31edede111b28fae7c2343612ef3aa9}$$
$$u = \texttt{0x8073145bb9bd729fa94bd34f9f6e}$$
$$\texttt{555ce7f2306415af1707e0dcd964923898b0d}$$
$$v = 4675$$
$$t = 273$$

where we also exploit the fact that we see the $4675 = 5^2 \cdot 11 \cdot 17$ torsion on E_A. This implies that we need to recover α and β mod 2^{273}, which is again well within the reach of the method described in the previous section.

Notice that in general the solution returned from the combinatorial attack will not be unique. However, only for the correct values of α, β the points βR_A and αS_A will interpolate the isogeny φ_A. In all other cases the corresponding isogeny chain will not land on a product of elliptic curves, giving us a distinguisher. With this method we can recover the secret φ_A. We do not recover the exact values of α, β directly (we only get them mod2^{273}) but they can be obtained by evaluating φ_A on P_b, Q_b.

We implemented the full attack in sage, showing that it can effectively recover a good portion of keys in 3 to 4 seconds on a laptop. For comparison, key generation takes 4 seconds on the same laptop. If the list of possible solutions to the equation $\alpha\beta = 1 \bmod 2^{273}$ grows, also the time taken to check all possible solutions grows. However, this step can be fully parallelized, and when run on 20 cores usually takes at most 5 minutes.

Polynomial time variants If we fix the maximum allowed size of Q, the algorithm described above runs in polynomial time in b. Fixing the maximal size of Q means discarding at each step all exceeding solutions. This implies that the correct solution has a certain probability of being discarded at every step. If the solutions are discarded at random, the probability that the correct solution is discarded grows sub-exponentially with n. Another way to interpret this is that while this combinatorial attack generates a subexponential number of possible paths, we can chose to follow only polynomially many of them.

The situation changes if we have a criterion to choose which solutions to discard. This is the case if the final solution has some easy to check property, e.g. if one among α and β is chosen to have small Hamming weight. In this case, we can compute the probability that at a certain step k there are more than S solution with a smaller Hamming weight that the correct one, assuming that bits of a random solution are uniformly 0 or 1 and bits of the correct one are 0 with the probability determined by the target Hamming weight. Notice that

for higher k it becomes easier to distinguish the correct solution. If we target a fixed success probability p, it is enough to fix S such that the probability of discarding the correct solution over the course of the algorithm stays below p to obtain a polynomial time algorithm.

Circulant matrices The FESTA trapdoor can be instantiated also using circulant matrices instead of diagonal ones (see [4, Sec. 3]). In that setting, what we obtain instead are two coefficients a, b such that $a^2 + b^2 = 1$ (or $a^2 - b^2 = 1$) modulo 2^k. Our combinatorial attack extends almost directly to this setting: we can write $a = \sum x_i 2^i$ and $b = \sum y_i 2^i$ and try to solve the equation symbolically modulo 2^i for increasing values of i. A small issue is that we are no longer able to recover all bits of a and b directly from that approach. For instance, the term x_{k-1} will only appear in the double product $2x_0(2^{k-1}x_{k-1})$, thus we can never see it modulo 2^k. If the first few bits of a are zero we lose information about lower coefficients x_i as well. However, this situation is very unlikely, and can occur for at most one of a and b, since one of the two must be odd. We implemented a proof of concept of this approach as well, and the results in practice are similar to the ones obtained for diagonal scaling. We leave a more detailed statistical analysis of this case as future work.

5.2 Continuous Hints

We now turn to the case where large, continuous blocks of bits leak. This scenario represents the worst case for the attack presented in the previous section as it cannot perform early rejections, leading to impractical running times. Therefore, we have to resort back to a more generic approach based on Coppersmith's method. Depending on the modulus N, our attack can either handle MSB hints $\mathbb{M}_k$ or continuous hints $\mathbb{C}_{s,k}$.

We first require the following lemma. Due to its similarities with Lemma 4.4 we do not provide a formal proof.

Lemma 5.10. *Given a modulus N, a polynomial $f(x, y) = (f_1+x)\cdot(f_2+y)+f_3$ for some constants $f_i \in \mathbb{N}$, bounds $X, Y \in \mathbb{N}$ and an arbitrarily small constant $\epsilon > 0$. If N is sufficiently large and*

$$XY \leq N^{2/3-\epsilon},$$

then under Heuristics 2.4 and 2.7 we can compute all $r \in Z_{N,X,Y}(f)$ in time polynomial in $\log(N)$.

We now get the following result, assuming MSB leakage and modulus a power of two (as it is the case in FESTA).

Proposition 5.11. *Let $\varphi : E_0 \to E'$ be an isogeny of known degree and $\alpha, \beta \in \mathbb{Z}_{2^b}$ with $\alpha\beta \equiv 1 \bmod 2^b$. There exists an efficient adversary against LIPH where $\Gamma = \begin{pmatrix} \alpha & \\ & \beta \end{pmatrix}$ and $\mathbb{M}_{2k}(\Gamma) = (\mathbb{M}_k(\alpha), \mathbb{M}_k(\beta))$ with $k = \lceil 2b/3 \rceil$.*

Proof. The proof relies on Lemma 5.10 and is conceptually identical to Theorem 4.3. □

We can improve Proposition 5.11 slightly in the case where the modulus 2^b is replaced by an odd modulus N. Indeed, in this case we can handle *continuous* hints as well since 2^e is invertible modulo N for any $e \in \mathbb{N}$. Note that such parameters were not officially proposed in [4].

Corollary 5.12. *Let $\varphi : E_0 \to E'$ be an isogeny of known degree and $\alpha, \beta \in \mathbb{Z}_N$ with $\alpha\beta \equiv 1 \bmod N$. There exists an efficient adversary against LIPH where $\Gamma = \left(\begin{smallmatrix} \alpha \\ \beta \end{smallmatrix}\right)$ and $\mathbb{C}_{s,2k}(\Gamma) = (\mathbb{C}_{s,k}(\alpha), \mathbb{C}_{s,k}(\beta))$ with $k = \lceil 2\log(N)/3 \rceil$ and $0 \leq s < \lceil \log(N) \rceil$.*

Proof. The proof is almost identical to Theorem 4.3, except for the fact that we express α and β as

$$\alpha = \alpha_1 \cdot 2^{e_1} + x \cdot 2^{e_0} + \alpha_0$$
$$\beta = \beta_1 \cdot 2^{e_1} + y \cdot 2^{e_0} + \beta_0$$

with $e_1 = \max\{\nu - s, \nu - (s + k \bmod \nu)\}$, $e_0 = \min\{\nu - s, \nu - (s + k \bmod \nu)\}$ and $\nu = \lceil \log N \rceil$. Equation (4) now yields

$$(\alpha_1 \cdot 2^{e_1} + x \cdot 2^{e_0} + \alpha_0) \cdot (\beta_1 \cdot 2^{e_1} + y \cdot 2^{e_0} + \beta_0) - 1 \equiv 0 \bmod N$$

and since 2^{e_0} is invertible modulo N, we can transform the above equation into a polynomial of the form $(f_1 + x) \cdot (f_2 + y) + f_3$. The result follows from Lemma 5.10. □

6 Experimental Results

We implemented the Coppersmith-type attacks on POKÉ and FESTA from Section 4 and Section 5.2 in Sage 10.5. We ran the attacks on a system with an Intel i7-1165G7 processor with 8 logical cores and 32GB of memory. For the lattice reduction we used `flatter`[7], leading to a significant increase in performance. Note that we do not provide performance results for the attack on M-SIDH as it performs almost identical to the attack on FESTA.

The results in Figure 2 show that we can get close to the asymptotic bounds stated in Lemmas 4.4 and 5.10. Furthermore, in every experiment Heuristic 2.7 was valid. Note that these runtimes only reflect the recovery of Γ and (potentially) $\deg \varphi$.

Acknowledgments. The authors thank Wouter Castryck for suggesting a closed form of Equation (6), and the organizers of the Leuven Isogeny Days where this project was started. Jonas Meers was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence

[7] https://github.com/keeganryan/flatter

Protocol	$\log N$	k (known bits)	m	Lattice Dimension	Runtime (seconds)
POKÉ	363	334 (92%)	10	36	2
POKÉ	363	327 (90%)	20	121	65
POKÉ	363	323 (89%)	30	256	720
POKÉ	726	668 (92%)	10	36	5
POKÉ	726	653 (90%)	20	121	145
POKÉ	726	646 (89%)	30	256	1830
POKÉ	1451	1335 (92%)	10	36	14
POKÉ	1451	1306 (90%)	20	121	400
POKÉ	1451	1276 (88%)	30	256	5331
FESTA	632	456 (72%)	10	36	3
FESTA	632	443 (70%)	20	121	78
FESTA	632	437 (69%)	30	256	1435
FESTA	992	715 (72%)	10	36	6
FESTA	992	695 (70%)	20	121	131
FESTA	992	685 (69%)	30	256	2480
FESTA	1472	1060 (72%)	10	36	12
FESTA	1472	1031 (70%)	20	121	236
FESTA	1472	1016 (69%)	30	256	4415

Fig. 2. Runtime for the Coppersmith-type attacks from Sections 4.2 and 5.2. Here N refers to the modulus of the corresponding pairing equation, which is either of the form $N = 2^a 3^b 5^c$ (POKE) or $N = 2^b$ (FESTA). The runtimes are averaged over 5 runs.

Strategy – EXC 2092 CASA – 390781972. Péter Kutas is partly supported by EPSRC through grant number EP/V011324/1. Péter Kutas is supported by the Hungarian Ministry of Innovation and Technology NRDI Office within the framework of the Quantum Information National Laboratory Program. Kutas is also supported by the grant "EXCELLENCE-151343". Subham Das was supported by the Government of Hungary through the Stipendium Hungaricum scholarship 2024/25. Riccardo Invernizzi is supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement ISOCRYPT – No. 101020788), by the Research Council KU Leuven grant C14/24/099, by CyberSecurity Research Flanders with reference number VOEWICS02, and by Research Foundation - Flanders (FWO) under a PhD Fellowship fundamental research (project number 1138925N). Together with Kutas, they are also supported by the CELSA alliance through the MaCro project.

References

1. Arpin, S.: Adding level structure to supersingular elliptic curve isogeny graphs. Journal de théorie des nombres de Bordeaux **36**(2), 405–443 (Nov 2024). https://doi.org/10.5802/jtnb.1283, http://dx.doi.org/10.5802/jtnb.1283
2. Basso, A., Maino, L.: POKÉ: A compact and efficient PKE from higher-dimensional isogenies. Cryptology ePrint Archive, Paper 2024/624 (2024), https://eprint.iacr.org/2024/624
3. Basso, A., Maino, L.: Poké: A compact and efficient pke from higher-dimensional isogenies. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 94–123. Springer (2025)
4. Basso, A., Maino, L., Pope, G.: FESTA: Fast encryption from supersingular torsion attacks. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part VII. LNCS, vol. 14444, pp. 98–126. Springer, Singapore (Dec 2023). https://doi.org/10.1007/978-981-99-8739-9_4
5. Bauer, A., Joux, A.: Toward a rigorous variation of Coppersmith's algorithm on three variables. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 361–378. Springer, Berlin, Heidelberg (May 2007). https://doi.org/10.1007/978-3-540-72540-4_21
6. Castryck, W., Decru, T.: An efficient key recovery attack on sidh. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 423–447. Springer (2023)
7. Castryck, W., Decru, T.: An efficient key recovery attack on SIDH. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 423–447. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_15
8. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An efficient post-quantum commutative group action. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part III. LNCS, vol. 11274, pp. 395–427. Springer, Cham (Dec 2018). https://doi.org/10.1007/978-3-030-03332-3_15
9. Coppersmith, D.: Finding a small root of a univariate modular equation. In: Maurer, U.M. (ed.) EUROCRYPT'96. LNCS, vol. 1070, pp. 155–165. Springer, Berlin, Heidelberg (May 1996). https://doi.org/10.1007/3-540-68339-9_14

10. Coron, J.S.: Finding small roots of bivariate integer polynomial equations revisited. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 492–505. Springer, Berlin, Heidelberg (May 2004). https://doi.org/10.1007/978-3-540-24676-3_29
11. De Feo, L., Delpech de Saint Guilhem, C., Fouotsa, T.B., Kutas, P., Leroux, A., Petit, C., Silva, J., Wesolowski, B.: Séta: Supersingular encryption from torsion attacks. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part IV. LNCS, vol. 13093, pp. 249–278. Springer, Cham (Dec 2021). https://doi.org/10.1007/978-3-030-92068-5_9
12. De Feo, L., Fouotsa, T.B., Panny, L.: Isogeny problems with level structure. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part VII. LNCS, vol. 14657, pp. 181–204. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58754-2_7
13. Esser, A., May, A., Verbel, J.A., Wen, W.: Partial key exposure attacks on BIKE, rainbow and NTRU. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part III. LNCS, vol. 13509, pp. 346–375. Springer, Cham (Aug 2022). https://doi.org/10.1007/978-3-031-15982-4_12
14. Fouotsa, T.B., Moriya, T., Petit, C.: M-SIDH and MD-SIDH: Countering SIDH attacks by masking information. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 282–309. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_10
15. Galbraith, S.D., Petit, C., Shani, B., Ti, Y.B.: On the security of supersingular isogeny cryptosystems. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part I. LNCS, vol. 10031, pp. 63–91. Springer, Berlin, Heidelberg (Dec 2016). https://doi.org/10.1007/978-3-662-53887-6_3
16. Henecka, W., May, A., Meurer, A.: Correcting errors in RSA private keys. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 351–369. Springer, Berlin, Heidelberg (Aug 2010). https://doi.org/10.1007/978-3-642-14623-7_19
17. Heninger, N., Shacham, H.: Reconstructing RSA private keys from random key bits. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 1–17. Springer, Berlin, Heidelberg (Aug 2009). https://doi.org/10.1007/978-3-642-03356-8_1
18. Jao, D., De Feo, L.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In: Post-Quantum Cryptography: 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29–December 2, 2011. Proceedings 4. pp. 19–34. Springer (2011)
19. Kirshanova, E., May, A.: Decoding McEliece with a hint - secret goppa key parts reveal everything. In: Galdi, C., Jarecki, S. (eds.) SCN 22. LNCS, vol. 13409, pp. 3–20. Springer, Cham (Sep 2022). https://doi.org/10.1007/978-3-031-14791-3_1
20. Maino, L., Martindale, C., Panny, L., Pope, G., Wesolowski, B.: A direct key recovery attack on sidh. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 448–471. Springer (2023)
21. Maino, L., Martindale, C., Panny, L., Pope, G., Wesolowski, B.: A direct key recovery attack on SIDH. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 448–471. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_16
22. May, A., Nowakowski, J.: Too many hints - when LLL breaks LWE. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part IV. LNCS, vol. 14441, pp. 106–137. Springer, Singapore (Dec 2023). https://doi.org/10.1007/978-981-99-8730-6_4
23. Meers, J., Nowakowski, J.: Solving the hidden number problem for CSIDH and CSURF via automated coppersmith. In: Guo, J., Steinfeld, R. (eds.) ASI-

ACRYPT 2023, Part IV. LNCS, vol. 14441, pp. 39–71. Springer, Singapore (Dec 2023). https://doi.org/10.1007/978-981-99-8730-6_2
24. Miller, V.S.: The Weil pairing, and its efficient calculation. Journal of Cryptology **17**(4), 235–261 (Sep 2004). https://doi.org/10.1007/s00145-004-0315-8
25. Nakagawa, K., Onuki, H.: QFESTA: Efficient algorithms and parameters for FESTA using quaternion algebras. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part V. LNCS, vol. 14924, pp. 75–106. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68388-6_4
26. Nakagawa, K., Onuki, H.: Qfesta: Efficient algorithms and parameters for festa using quaternion algebras. In: Annual International Cryptology Conference. pp. 75–106. Springer (2024)
27. Page, A., Robert, D.: Introducing clapoti (s): Evaluating the isogeny class group action in polynomial time. Cryptology ePrint Archive (2023)
28. Robbins, H.: A remark on stirling's formula. The American mathematical monthly **62**(1), 26–29 (1955)
29. Robert, D.: Breaking sidh in polynomial time. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 472–503. Springer (2023)
30. Robert, D.: Breaking SIDH in polynomial time. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 472–503. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_17
31. Robert, D.: On the efficient representation of isogenies (a survey). Cryptology ePrint Archive, Report 2024/1071 (2024), https://eprint.iacr.org/2024/1071
32. Seto, Y., Furue, H., Takayasu, A.: Partial key exposure attacks on UOV and its variants. Cryptology ePrint Archive, Report 2025/595 (2025), https://eprint.iacr.org/2025/595
33. Vélu, J.: Isogénies entre courbes elliptiques. Comptes-Rendus de l'Académie des Sciences **273**, 238–241 (1971)
34. Weil, A.: Sur les fonctions algébriques à corps de constantes fini. Les Comptes Rendus de l'Académie des Sciences (1940)