ON THE COST OF COMPUTING ISOGENIES BETWEEN SUPERSINGULAR ELLIPTIC CURVES

GORA ADJ, DANIEL CERVANTES, JESÚS-JAVIER CHI, ALFRED MENEZES, AND FRANCISCO RODRÍGUEZ-HENRÍQUEZ

Abstract. The security of the Jao-De Feo Supersingular Isogeny Diffie-Hellman (SIDH) key agreement scheme is based on the intractability of the Computational Supersingular Isogeny (CSSI) problem — computing $\mathbb{F}_{p^2}$-rational isogenies of degrees 2^e and 3^e between certain supersingular elliptic curves defined over $\mathbb{F}_{p^2}$. The classical meet-in-the-middle attack on CSSI has an expected running time of $O(p^{1/4})$, but also has $O(p^{1/4})$ storage requirements. In this paper, we demonstrate that the van Oorschot-Wiener collision finding algorithm has a lower cost (but higher running time) for solving CSSI, and thus should be used instead of the meet-in-the-middle attack to assess the security of SIDH against classical attacks. The smaller parameter p brings significantly improved performance for SIDH.

1. Introduction

The Supersingular Isogeny Diffie-Hellman (SIDH) key agreement scheme was proposed by Jao and De Feo [11] (see also [6]). It is one of 69 candidates being considered by the U.S. government's National Institute of Standards and Technology (NIST) for inclusion in a forthcoming standard for quantum-safe cryptography [10]. The security of SIDH is based on the difficult of the Computational Supersingular Isogeny (CSSI) problem, which was first defined by Charles, Goren and Lauter [3] in their paper that introduced an isogeny-based hash function. The CSSI problem is also the basis for the security of isogeny-based signature schemes [8, 27] and an undeniable signature scheme [12].

Let p be a prime, let ℓ be a small prime (e.g., $\ell \in \{2, 3\}$), and let E and E' be two supersingular elliptic curves defined over $\mathbb{F}_{p^2}$ for which a (separable) degree-ℓ^e isogeny $\phi : E \to E'$ defined over $\mathbb{F}_{p^2}$ exists. The CSSI problem is that of constructing such an isogeny. In [6], the CSSI problem is assessed as having a complexity of $O(p^{1/4})$ and $O(p^{1/6})$ against classical and quantum attacks [22], respectively. The classical attack is a meet-in-the-middle attack which has time complexity $O(p^{1/4})$ and space complexity $O(p^{1/4})$. We observe that the van Oorschot-Wiener collision finding (classical) algorithm [15, 16] can be employed to construct ϕ. Whereas the time complexity of the van Oorschot-Wiener algorithm is higher than that of the meet-in-the-middle attack, its space requirements are smaller. Our cost analysis of these two CSSI attacks leads to the conclusion that, despite its higher running time, the parallel collision finding CSSI attack has a lower cost than the meet-in-the-middle attack, and thus should be used to assess the security of SIDH against (known) classical attacks. Our work is inspired by Bernstein's paper [1] which presents convincing arguments that the $O(n^{1/2})$-time van Oorschot-Wiener algorithm for

Date: April 3, 2018.

finding collisions for an n-bit hash function is less costly than the $O(n^{1/3})$-time quantum algorithm of Brassard, Høyer and Tapp [2].

The remainder of this paper is organized as follows. The CSSI problem and relevant mathematics background are presented in §2. In §3 and §4, we report on our implementation of the meet-in-the-middle and parallel collision search methods for solving CSSI. Our implementations confirm that the heuristic analysis of these CSSI attacks accurately predicts their performance in practice. Our cost models and cost comparisons are presented in §5. Finally, in §6 we make some concluding remarks.

2. Computational Supersingular Isogeny Problem

2.1. **Mathematical prerequisites.** Let $p = \ell_A^{e_A}\ell_B^{e_B} - 1$ be a prime[1], where ℓ_A and ℓ_B are distinct small primes and $\ell_A^{e_A} \approx \ell_B^{e_B} \approx p^{1/2}$. Let E be a (supersingular) elliptic curve defined over $\mathbb{F}_{p^2}$ with $\#E(\mathbb{F}_{p^2}) = (p+1)^2$. Then $E(\mathbb{F}_{p^2}) \cong \mathbb{Z}_{p+1} \oplus \mathbb{Z}_{p+1}$, whence the torsion groups $E[\ell_A^{e_A}]$ and $E[\ell_B^{e_B}]$ are contained in $E(\mathbb{F}_{p^2})$.

In the following, we write (ℓ, e) to mean either (ℓ_A, e_A) or (ℓ_B, e_B). All isogenies ϕ considered in this paper are separable, whereby $\deg\phi = \#\mathrm{Ker}(\phi)$.

Let S be an order-ℓ^e subgroup of $E[\ell^e]$. Then there exists an isogeny $\phi : E \to E'$ (with both ϕ and E' defined over $\mathbb{F}_{p^2}$) with kernel S. The isogeny ϕ is unique up to isomorphism in the sense that if $\tilde{\phi} : E \to \tilde{E}$ is another isogeny defined over $\mathbb{F}_{p^2}$ with kernel S, then there exists an $\mathbb{F}_{p^2}$-isomorphism $\psi : E' \to \tilde{E}$ with $\tilde{\phi} = \psi \circ \phi$.

Given E and S, an isogeny ϕ with kernel S and the equation of E' can be computed using Vélu's formula [23]. The running time of Vélu's formula is polynomial in $\#S$ and $\log p$. Since $\#S \approx p^{1/2}$, a direct application of Vélu's formula does not yield a polynomial-time algorithm for computing ϕ and E'. However, since $\#S$ is a power of a small prime, one can compute ϕ and E' in time that is polynomial in $\log p$ by using Vélu's formula to compute a sequence of e degree-ℓ isogenies (see §2.2).

We will denote the elliptic curve that Vélu's formula yields by E/S and the (Vélu) isogeny by $\phi_S : E \to E/S$. As noted above, ϕ_S is unique up to isomorphism. Thus, for any fixed E, there is a one-to-one correspondence between order-ℓ^e subgroups of $E[\ell^e]$ and isogenies $\phi : E \to E'$ of order ℓ^e defined over $\mathbb{F}_{p^2}$. It follows that the number of order-ℓ^e isogenies $\phi : E \to E'$ is $\ell^e + \ell^{e-1} = (\ell+1)\ell^{e-1}$.

2.2. **Vélu's formula.** Vélu's formula (see Theorem 12.16 in [24]) can be used to compute degree-ℓ isogenies. We present Vélu's formula for $\ell = 2$ and $\ell = 3$.

Consider the elliptic curve $E/\mathbb{F}_{p^2} : Y^2 = X^3 + aX + b$, and let $P \in E(\mathbb{F}_{p^2})$ be a point of order two. Let $v = 3X_P^2 + a$, $a' = a - 5v$, $b' = b - 7vX_P$, and define the elliptic curve $E'/\mathbb{F}_{p^2} : Y^2 = X^3 + a'X + b'$. Then the map

$$(X, Y) \mapsto \left(\frac{v}{X - X_P} + X_P,\ Y - \frac{vY}{(X - X_P)^2}\right)$$

is a degree-2 isogeny from E to E' with kernel $\langle P \rangle$.

Let $P \in E(\mathbb{F}_{p^2})$ be a point of order three. Let $v = 3X_P^2 + a$, $u = 4Y_P^3$, $a' = a - 5v$, $b' = b - 7(u + vX_P)$, and define the elliptic curve $E'/\mathbb{F}_{p^2} : Y^2 = X^3 + a'X + b'$. Then the

[1]More generally, one can take $p = \ell_A^{e_A}\ell_B^{e_B}d \pm 1$ where d is a small cofactor.

map

$$(X,Y) \mapsto \left(\frac{v}{X-X_P} + \frac{u}{(X-X_P)^2} + X_P,\ Y\left(1 - \frac{v}{(X-X_P)^2} + \frac{2u}{(X-X_P)^3}\right)\right)$$

is a degree-3 isogeny from E to E' with kernel $\langle P \rangle$.

Suppose now that $R \in E(\mathbb{F}_{p^2})$ has order ℓ^e where $\ell \in \{2,3\}$ and $e \geq 1$. Then the isogeny $\phi : E \to E/\langle R \rangle$ can be efficiently computed as follows. Define $E_0 = E$ and $R_0 = R$. For $i = 0, 1, \ldots, e-1$, let $\phi_i : E_i \to E_{i+1}$ be the degree-ℓ isogeny obtained using Vélu's formula with kernel $\langle R_i \rangle$, and let $R_{i+1} = \phi_i(R_i)$. Then $\phi = \phi_{e-1} \circ \cdots \circ \phi_0$.

Remark 1. (*cost of computing an ℓ^e-isogeny*) As shown in [6] (see also [11, 5]), an optimal strategy for computing a degree-ℓ^e isogeny requires about $\frac{e}{2}\log_2 e$ scalar multiplications by ℓ and $\frac{e}{2}\log_2 e$ degree-ℓ isogeny evaluations, as well as e constructions of degree-ℓ isogenous curves.

2.3. **SIDH.** In SIDH, the parameters ℓ_A, ℓ_B, e_A, e_B, p and E are fixed and public, as are bases $\{P_A, Q_A\}$ and $\{P_B, Q_B\}$ for the torsion groups $E[\ell_A^{e_A}]$ and $E[\ell_B^{e_B}]$.

In (unauthenticated) SIDH, Alice selects $m_A, n_A \in_R [0, \ell_A^{e_A} - 1]$, not both divisible by ℓ_A, and sets $R_A = m_A P_A + n_A Q_A$ and $A = \langle R_A \rangle$; note that A is an order-$\ell_A^{e_A}$ subgroup of $E[\ell_A^{e_A}]$. Alice then computes the isogeny $\phi_A : E \to E/A$ while keeping A and ϕ_A secret. She transmits

$$E/A, \quad \phi_A(P_B), \quad \phi_A(Q_B) \tag{1}$$

to Bob. Similarly, Bob selects $m_B, n_B \in_R [0, \ell_B^{e_B} - 1]$, not both divisible by ℓ_B, and sets $R_B = m_B P_B + n_B Q_B$ and $B = \langle R_B \rangle$. Bob then computes the isogeny $\phi_B : E \to E/B$. He keeps B and ϕ_B secret and transmits

$$E/B, \quad \phi_B(P_A), \quad \phi_B(Q_A) \tag{2}$$

to Alice. Thereafter, Alice computes $\phi_B(R_A) = m_A \phi_B(P_A) + n_A \phi_B(Q_A)$ and

$$(E/B)/\langle \phi_B(R_A) \rangle, \tag{3}$$

whereas Bob computes $\phi_A(R_B) = m_B \phi_A(P_B) + n_B \phi_A(Q_B)$ and

$$(E/A)/\langle \phi_A(R_B) \rangle. \tag{4}$$

The composition of isogenies

$$E \to E/A \to (E/A)/\langle \phi_A(R_B) \rangle$$

and

$$E \to E/B \to (E/B)/\langle \phi_B(R_A) \rangle$$

both have kernel $\langle R_A, R_B \rangle$. Hence the elliptic curves in (3) and (4) are isomorphic over $\mathbb{F}_{p^2}$, and Alice and Bob's shared secret k is the j-invariant of these curves.

Remark 2. (*SIDH vs. SIKE*) SIDH is an unauthenticated key agreement protocol. The NIST submission [10] specifies a variant of SIDH that is a key encapsulation mechanism (KEM) called SIKE (Supersingular Isogeny Key Encapsulation). In SIKE, Alice's long-term public key is $(E/A, \phi_A(P_B), \phi_A(Q_B))$. Bob sends Alice an ephemeral public key $(E/B, \phi_B(P_A), \phi_B(Q_A))$ where B is derived from Alice's public key and a random string, and then computes a session key from the j-invariant of the elliptic curve (4), the aforementioned random string, and the ephemeral public key. One technical difference between the

original SIDH specification in [11, 6] and the SIKE specification in [10] (and also the SIDH implementation in [4]) is that in the latter the secret R_A is of the form $P_A + n_A Q_A$ where n_A is selected (almost) uniformly at random from the interval $[0, \ell_A^{e_A} - 1]$ (and similarly for R_B). Thus, R_A is selected uniformly at random from a subset of size approximately ℓ^{e_A} of the set of all order-$\ell_A^{e_A}$ subgroups (which has cardinality $\ell_A^{e_A} + \ell_A^{e_A - 1}$).

2.4. **CSSI.** The challenge faced by a passive adversary is to compute k given the public parameters, E/A, E/B, $\phi_A(P_B)$, $\phi_A(Q_B)$, $\phi_B(P_A)$ and $\phi_B(Q_A)$. A necessary condition for hardness of this problem is the intractability of the Computational Supersingular Isogeny (CSSI) problem: Given the public parameters ℓ_A, ℓ_B, e_A, e_B, p, E, P_A, Q_A, P_B, Q_B, the elliptic curve E/A, and the auxiliary points $\phi_A(P_B)$ and $\phi_A(Q_B)$, compute the Vélu isogeny $\phi_A : E \to E/A$ (or, equivalently, determine a generator of A).

An assumption one makes is that the auxiliary points $\phi_A(P_B)$ and $\phi_A(Q_B)$ are of no use in solving CSSI. Thus, we can simplify the statement of the CSSI problem to the following:

Problem 1 (CSSI)**.** Given the public parameters ℓ_A, ℓ_B, e_A, e_B, p, E, P_A, Q_A, and the elliptic curve E/A, compute an isogeny $\phi_A : E \to E/A$, or, equivalently, determine a generator of A.

3. Meet-in-the-Middle

For the sake of simplicity, we will suppose that e is even. We denote the number of order-$\ell^{e/2}$ subgroups of $E[\ell^e]$ by $N = (\ell + 1)\ell^{e/2-1} \approx p^{1/4}$.

Let $E_1 = E$ and $E_2 = E/A$. Let R denote the set of all j-invariants of elliptic curves that are isogenous to E_1; then $\#R \approx p/12$ [19]. Let R_1 denote the set of all j-invariants of elliptic curves over $\mathbb{F}_{p^2}$ that are $\ell^{e/2}$-isogenous to E_1. Since $\#R \gg N$, one expects that the number of pairs of distinct order-$\ell^{e/2}$ subgroups (A_1, A_2) of $E_1[\ell^e]$ with $j(E_1/A_1) = j(E_1/A_2)$ is very small. Thus, we shall assume for the sake of simplicity that $\#R_1 = N$. Similarly, we let R_2 denote the set of all j-invariants of elliptic curves that are $\ell^{e/2}$-isogenous to E_2, and assume that $\#R_2 = N$. Since E_1 is ℓ^e-isogenous to E_2, we know that $R_1 \cap R_2 \neq \emptyset$. Moreover, since $\#R_1 \ll \#R$ and $\#R_2 \ll \#R$, it is reasonable to assume that $\#(R_1 \cap R_2) = 1$; in other words, we assume that there is a unique degree-ℓ^e isogeny $\phi : E_1 \to E_2$.

3.1. **The attack.** The meet-in-the-middle attack on CSSI [6] (see Figure 1) proceeds by building a (sorted) table with entries $(j(E_1/A_1), A_1)$, where A_1 ranges over all order-$\ell^{e/2}$ subgroups of $E_1[\ell^e]$. Next, for each order-$\ell^{e/2}$ subgroup A_2 of $E_2[\ell^e]$, one computes E_2/A_2 and searches for $j(E_2/A_2)$ in the table. If $j(E_2/A_2) = j(E_1/A_1)$, then the composition of isogenies

$$\phi_{A_1} : E_1 \to E_1/A_1, \quad \psi : E_1/A_1 \to E_2/A_2, \quad \hat{\phi}_{A_2} : E_2/A_2 \to E_2,$$

where ψ is an $\mathbb{F}_{p^2}$-isomorphism and $\hat{\phi}$ denotes the dual of ϕ, is the desired degree-ℓ^e isogeny from E_1 to E_2.

The worst-case time complexity of the meet-in-the-middle attack is $T_1 = 2N$, where a unit of time is the time it takes to compute a degree-$\ell^{e/2}$ Vélu isogeny. The average-case time complexity is $1.5N$. The attack has space complexity N.

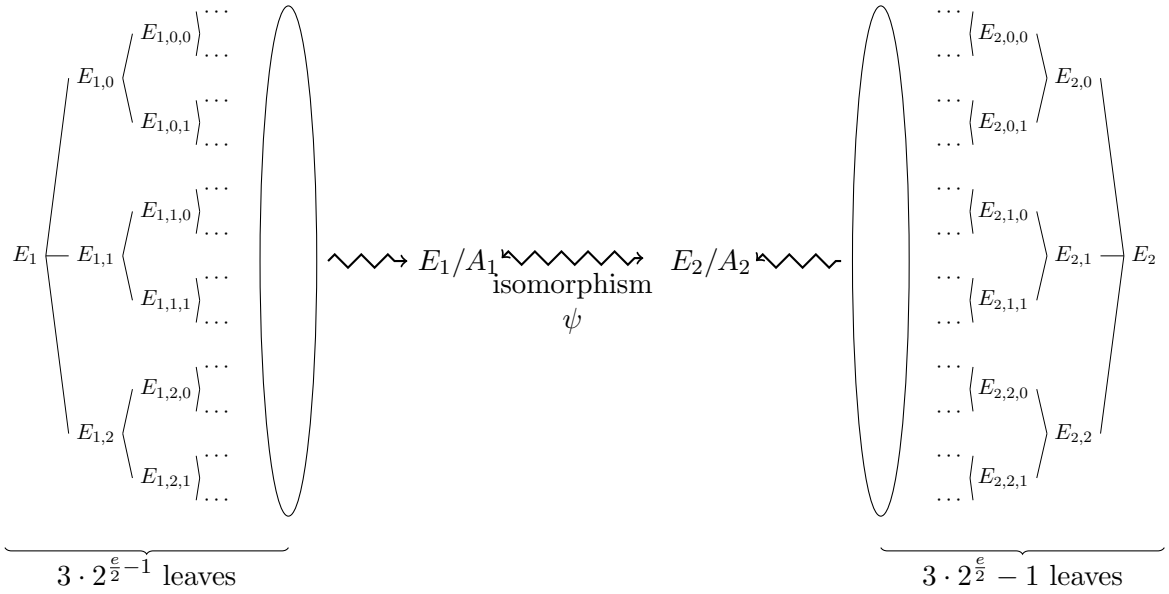

FIGURE 1. Meet-in-the-middle attack for degree-2 isogeny trees.

3.2. **Implementation report.** The meet-in-the-middle attack was implemented in C, compiled using gcc version 4.7.2, and executed on an Intel Xeon processor E5-2658 v2 server equipped with 20 physical cores and 256 GB of shared RAM memory. We used fopenmp for the parallelization.

For $p = 2^{e_A}3^{e_B}d - 1$, the elliptic curve $E/\mathbb{F}_p : Y^2 = X^3 + X$ has $\#E(\mathbb{F}_p) = p + 1$ and $\#E(\mathbb{F}_{p^2}) = (p+1)^2$. A point $P \in E(\mathbb{F}_{p^2})$ of order $2 \cdot 3 \cdot d$ was randomly selected, and the isogenous elliptic curve $E_1 = E/\langle P \rangle$ was computed. Then, a random order-2^{e_A} subgroup A of $E_1(\mathbb{F}_{p^2})$ was selected, and the isogenous elliptic curve $E_2 = E_1/A$ was computed. Our CSSI challenge was to find a generator of A given E_1 and E_2.

We used Jacobian coordinates for elliptic curve arithmetic, isogeny computations, and isogeny evaluations. The leaves of the E_1-rooted tree shown in Figure 1 were generated as follows. Let (P, Q) be a basis for $E_1[2^{e/2}]$. Then for each pair $(b, k) \in \{0, 1, 2\} \times \{0, 1, \ldots, 2^{e/2-1} - 1\}$, triples

$$\begin{aligned} \left(j(E_1/\langle P + (b2^{e/2-1} + k)Q\rangle),\ \ b,\ \ b2^{e/2-1} + k\right), \qquad & \text{for } b = 0, 1, \\ \left(j(E_1/\langle (2k)P + Q\rangle),\ \ b,\ \ k\right), \qquad & \text{for } b = 2, \end{aligned}$$

were computed and stored in 20 tables sorted by j-invariant (each of the 20 cores was responsible for generating a portion of the leaves). The 20 tables were stored in shared RAM memory.

Table 1 shows the time expended for finding 2^e-isogenies for $e \in \{32, 34, 36, 38, 40, 42, 44\}$ with the MITM attack. These experimental results confirm the accuracy of the attack's heuristic analysis.

						median		average	
e_A	e_B	d	expected time	number of runs	space	measured time	clock cycles	measured time	clock cycles
32	20	23	$2^{17.17}$	19	$2^{20.72}$	$2^{17.07}$	$2^{34.35}$	$2^{17.06}$	$2^{34.28}$
34	21	109	$2^{18.17}$	19	$2^{21.83}$	$2^{18.21}$	$2^{35.39}$	$2^{18.20}$	$2^{35.37}$
36	22	31	$2^{19.17}$	19	$2^{22.87}$	$2^{19.12}$	$2^{36.39}$	$2^{19.15}$	$2^{36.43}$
38	23	271	$2^{20.17}$	19	$2^{23.99}$	$2^{20.22}$	$2^{37.53}$	$2^{20.21}$	$2^{37.53}$
40	25	71	$2^{21.17}$	19	$2^{25.04}$	$2^{21.14}$	$2^{38.51}$	$2^{21.13}$	$2^{38.52}$
42	26	37	$2^{22.17}$	19	$2^{26.09}$	$2^{22.21}$	$2^{39.82}$	$2^{22.21}$	$2^{39.81}$
44	27	37	$2^{23.17}$	6	$2^{27.14}$	$2^{23.12}$	$2^{40.79}$	$2^{23.09}$	$2^{40.76}$

TABLE 1. Meet-in-the-middle attack for finding a 2^{e_A}-isogeny between two supersingular elliptic curves over $\mathbb{F}_{p^2}$ with $p = 2^{e_A} \cdot 3^{e_B} \cdot d - 1$. For each p, the listed number of CSSI instances were solved and the median and average of the results are reported. The running time columns give the expected number and the actual number of degree-$2^{e_A/2}$ isogeny computations. The space is measured in bytes.

4. Parallel collision search

4.1. **Van Oorschot-Wiener parallel collision search.** Let S be a finite set of cardinality M, and let $f : S \to S$ be an efficiently-computable function which we shall heuristically assume is a random function. The van Oorschot-Wiener (VW) method [16] finds a collision for f, i.e., a pair $x, x' \in S$ with $f(x) = f(x')$ and $x \neq x'$.

Define an element x of S to be *distinguished* if it has some easily-testable distinguishing property. Suppose that the proportion of elements of S that are distinguished is θ. For $i = 1, 2, \ldots$, the VW method repeatedly selects $x_{i,0} \in_R S$, and iteratively computes a sequence $x_{i,j} = f(x_{i,j-1})$ for $j = 1, 2, 3, \ldots$ until a distinguished element $x_{i,a}$ is encountered. In that event, the triple $(x_{i,a}, a, x_{i,0})$ is stored in a table sorted by first entry. If $x_{i,a}$ was already in the table, say $x_{i,a} = x_{i',b}$ with $i \neq i'$, then a collision has been *detected* (see Figure 2). The two colliding table entries $(x_{i,a}, a, x_{i,0})$, $(x_{i',b}, b, x_{i',0})$ can then be used to find a collision for f by iterating the longer sequence (say the ith sequence) beginning at $x_{i,0}$ until it is the same distance from $x_{i,a}$ as $x_{i',0}$ is from $x_{i',b}$, and then stepping both sequences in unison until they collide (see Figure 3).

By the birthday paradox, the expected time before a collision occurs is $\sqrt{\pi M/2}$, where a unit of time is an f evaluation. After a collision has occurred, the expected time before it is detected is $1/\theta$, and thereafter the expected time to find the collision is approximately $3/\theta$. Thus, the expected time complexity of the VW method is approximately $\sqrt{\pi M/2} + 4/\theta$. The expected storage complexity is $\theta\sqrt{\pi M/2}$. The parameter θ can be selected to control the storage requirements.

The collision detecting stage of the VW method can be effectively parallelized. Each of the available m processors computes its own sequences, and the distinguished elements

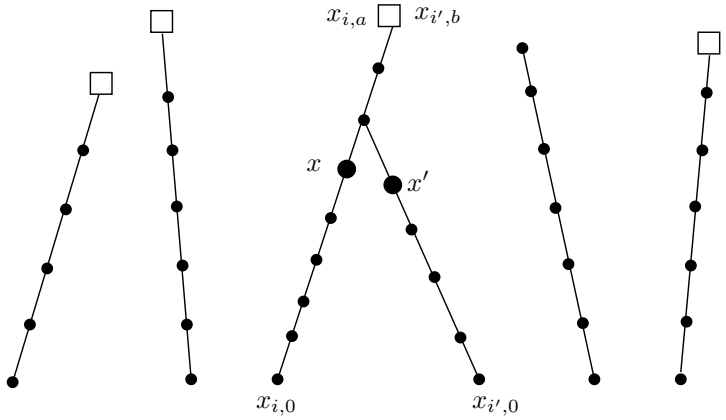

FIGURE 2. VW method: detecting a collision (x, x').

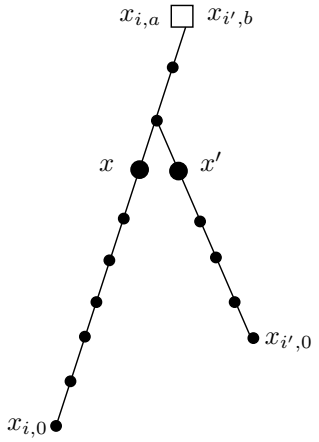

FIGURE 3. VW method: finding a collision (x, x').

are stored in shared memory. The expected time complexity of parallelized VW is then $\frac{1}{m}\sqrt{\pi M/2} + \frac{2.5}{\theta}$. The space complexity is $\theta\sqrt{\pi M/2}$.

4.2. **Finding a golden collision.** Van Oorschot and Wiener estimate that a random function $f : S \to S$ is expected to have $M/2$ unordered collisions. Suppose that we seek a particular one of the collisions, called a *golden collision*; we assume that the golden collision can be efficiently recognized. Then, one can expect to have to find $M/4$ collisions until the golden collision is found. Thus one continues generating distinguished points and collisions until the golden collision is encountered. The expected time to find q collisions is only $\sqrt{q}$ times as much as that to find one collision. However, since not all collisions are equally likely and the golden collision might have a very low probability of detection (see [15]), it is necessary to change the version of f periodically.

Suppose that the available memory can store w triples $(x_{i,a}, a, x_{i,0})$. When a distinguished point $x_{i,a}$ is encountered, the triple $(x_{i,a}, a, x_{i,0})$ is stored in a memory cell determined by hashing $x_{i,a}$. If the memory cell was already occupied with a triple holding a distinguished point $x_{i',b} = x_{i,a}$, then the two triples are used to locate a collision.

Van Oorschot and Wiener proposed setting

$$\theta = \alpha\sqrt{w/M} \tag{5}$$

and using each version of f to produce βw distinguished points. Experimental data presented in [16] suggested that the total running time to find the golden collision is minimized by setting $\alpha = 2.25$ and $\beta = 10$. Then, for $2^{10} \leq w \leq M/2^{10}$, the expected running time to find the golden collisions when m processors are employed is slightly overestimated as

$$\frac{1}{m}(2.5\sqrt{M^3/w}). \tag{6}$$

4.3. **The attack.** Let $I = \{1, 2, \ldots, N\}$ and $S = \{1, 2\} \times I$. For $i = 1, 2$, let $\mathcal{A}_i$ denote the set of all order-$\ell^{e/2}$ subgroups of $E_i[\ell^e]$, define $f_i : \mathcal{A}_i \to R_i$ by $f_i(A_i) = j(E_i/A_i)$, and let $h_i : I \to \mathcal{A}_i$ be bijections. Let $g : R \to S$ be a random function. Finally, define $f : S \to S$ by

$$f : (i, x) \mapsto g(f_i(h_i(x))).$$

Then one can view f as a "random" function from S to S.

Recall that one expects there are unique order-$\ell^{e/2}$ subgroups A_1, A_2 of $E_1[\ell^e]$, $E_2[\ell^e]$, respectively, with $j(E_1/A_1) = j(E_2/A_2)$. Let $y_1 = h_1^{-1}(A_1)$ and $y_2 = h_2^{-1}(A_2)$. Then the collision for f that we seek is the golden collision $(1, y_1)$, $(2, y_2)$. Using m processors and w cells of memory, the VW method can be used to find this golden collision in expected time

$$\frac{1}{m}(2.5\sqrt{8N^3/w}) \;\approx\; 7.1p^{3/8}/(w^{1/2}m). \tag{7}$$

Remark 3. (*finding any collision vs. finding a golden collision*) The problem of finding a collision for a hash function $H : \{0,1\}^* \to \{0,1\}^n$ and the problem of computing discrete logarithms in a cyclic group $\mathcal{G}$ can be formulated as problems of finding a collision for a random function $f : S \to S$, where $\#S = 2^n$ for the first problem and $\#S = \#\mathcal{G}$ for the second problem (see [16]). For both formulations, *any* collision for f yields a solution to the original problem. Thus, letting $N = 2^n$ or $N = \#\mathcal{G}$, the problems can be solved using van Oorschot-Wiener collision search in time approximately

$$\frac{1}{m}N^{1/2}. \tag{8}$$

In contrast, the only formulation of CSSI as a collision search problem for $f : S \to S$ that we know requires one to find a *golden* collision for f. For this problem, the van Oorschot-Wiener algorithm has running time approximately

$$N^{3/2}/(w^{1/2}m). \tag{9}$$

4.4. **Implementation report.** The VW attack was implemented in C, compiled using gcc version 4.7.2, and executed on an Intel Xeon processor E5-2658 v2 server equipped with 20 physical cores and 256 GB of shared RAM memory. We used fopenmp for the parallelization and openssl's MD5 implementation. The CSSI challenges were the same as the ones in §3.2.

Let (P_1, Q_1), (P_2, Q_2) be bases for $E_1[2^{e/2}]$, $E_2[2^{e/2}]$, respectively. Noting that $N = 3 \cdot 2^{e/2-1}$, we identify the elements of $I = \{1, 2, \ldots, N\}$ with elements of $I_1 \times I_2$ where $I_1 = \{0, 1, 2\}$ and $I_2 = \{0, 1, \ldots, 2^{e/2-1} - 1\}$. The bijections $h_i : I_1 \times I_2 \to \mathcal{A}_i$ for $i = 1, 2$ are defined by

$$h_i : (b, k) \mapsto \begin{cases} P_i + (b2^{e/2-1} + k)Q_i, & \text{if } b = 0, 1, \\ (2k)P_i + Q_i, & \text{if } b = 2. \end{cases}$$

Let $S = \{1, 2\} \times I_1 \times I_2$. For $n \in \{0, 1\}^{64}$, we let $g_n : R \to S$ be the function computed using Algorithm 1. We then define the version $f_n : S \to S$ of f by $(i, x) \mapsto g_n(f_i(h_i(x)))$.

Algorithm 1 The "random" function g_n

Require: $n \in \{0, 1\}^{64}$ and $j \in \mathbb{F}_{p^2}$.
Ensure: Output $c \in \{1, 2\}$, $b \in I_1$, $k \in I_2$.
1: $counter := 0$;
2: $h := \text{MD5}(j, n, counter)$.
3: Let h' be the $e/2 + 2$ least significant bits of h, and parse h' as (k, c, b), where k, c, b have bitlengths $e/2 - 1$, 1, and 2, respectively.
4: **if** $b = 11$ **then**
5: $counter := counter + 1$;
6: Go to Step 2
7: **end if**
8: **return** $(c + 1, b, k)$;

Table 2 shows the time expended for finding 2^e-isogenies for $e \in \{32, 34, 36, 38, 40, 42\}$ with the VW attack. These experimental results partially confirm the accuracy of the VW attack's heuristic analysis. We are conducting additional experiments with larger values of e to gather more evidence that the VW attack performs as expected.

5. Comparisons

There are many factors that can affect the efficacy of an algorithm.

(1) *Time*: the worst-case or average-case number of basic arithmetic operations performed by the algorithm.
(2) *Space*: the amount of storage (RAM, hard disk, etc.) required.
(3) *Parallelizability*: the speedup achievable when running the algorithm on multiple processors. Ideally, the speedup is by a factor equal to the number of processors, and the processors do not need to communicate with each other; if this is the case then the parallelization is said to be *perfect*[2].

[2]If the processors share the same storage space, then frequent storage accesses might decrease the parallelizability of the algorithm.

							median			average		
e_A	e_B	d	w	r	expected time	number of runs	# f_n's	measured time	clock cycles	# f_n's	measured time	clock cycles
32	20	23	2^9	3	$2^{23.20}$	19	193	$2^{23.07}$	$2^{40.53}$	466	$2^{24.35}$	$2^{41.80}$
34	21	109	2^9	4	$2^{24.70}$	19	372	$2^{25.01}$	$2^{42.56}$	432	$2^{25.23}$	$2^{42.78}$
36	22	31	2^{10}	4	$2^{25.70}$	19	423	$2^{26.20}$	$2^{43.84}$	530	$2^{26.52}$	$2^{44.17}$
38	23	271	2^{11}	4	$2^{26.70}$	19	750	$2^{28.02}$	$2^{45.66}$	1186	$2^{28.68}$	$2^{46.32}$
40	25	71	2^{11}	4	$2^{28.20}$	19	416	$2^{27.15}$	$2^{44.87}$	1639	$2^{29.13}$	$2^{46.85}$
42	26	37	2^{12}	4	$2^{29.20}$	19	1188	$2^{29.66}$	$2^{47.57}$	2750	$2^{30.87}$	$2^{48.78}$

TABLE 2. Van Oorschot-Wiener parallel collision search for finding a 2^{e_A}-isogeny between two supersingular elliptic curves over $\mathbb{F}_{p^2}$ with $p = 2^{e_A} \cdot 3^{e_B} \cdot d - 1$. For each p, the listed number of CSSI instances were solved and the median and average of the results are reported. The parameter r in the fifth column indicates the number of most significant bits of the parameter k that must be zero in order for an element of S to be considered distinguished. The $\#f_n$'s column indicates how many random functions f_n were tested until the golden collision was found. The expected and measured times list number of degree-$2^{e_A/2}$ isogeny computations.

(4) *Communication costs*: the time taken for communication between processors, and the memory access time for retrieving data from large storage devices. Memory access time can be a dominant cost factor when using extremely large storage devices [1].
(5) *Custom-designed devices*: the possible speedups that can be achieved by executing the algorithm on custom-designed hardware. Examples of such devices are TWINKLE [20] and TWIRL [21] that were designed for the number field sieve integer factorization algorithm.

In this section we analyze and compare the efficacy of the meet-in-the-middle algorithm, parallel collision search, and a mesh sorting algorithm for solving CSSI. We make two assumptions:

(1) The number m of processors available is at most 2^{64}.
(2) The total amount of storage w available is at most 2^{80} units.

Our analysis will ignore communication costs, and thus our running time estimates can be considered to be lower bounds on the "true" running time.

Remark 4. (*feasible amount of storage and number of processors*) The Sunway TaihuLight supercomputer, the most powerful in the world as of March 2018, has $2^{23.3}$ CPU cores [26]. In 2013, it was estimated that Google's data centres have a total storage capacity of about a dozen exabytes[3] [26]. Thus it is reasonable to argue that acquiring 2^{64} processors

[3]An exabyte is 2^{60} bytes.

and a storage capacity (with low access times) of several dozen yottabytes[4] for the purpose of solving a CSSI problem is prohibitively costly for the foreseeable future.

5.1. **Meet-in-the-middle.** As stated in §3.1, the running time of the meet-in-the-middle attack is approximately $2N$ and the storage requirements are N, where $N \approx p^{1/4}$. Since for $N \geq 2^{80}$ the storage requirements are infeasible, we deem the meet-in-the-middle attack to be prohibitively expensive when $N \gg 2^{80}$.

Of course, one can trade space for time. One possible time-memory tradeoff is to store a table with entries $(j(E_1/A_1), A_1)$, where A_1 ranges over a w-subset of order-$\ell^{e/2}$ subgroups of $E_1[\ell^e]$. Next, for each order-$\ell^{e/2}$ subgroup A_2 of $E_2[\ell^e]$, E_2/A_2 is computed and $j(E_2/A_2)$ is searched in the table. If no match is found, then the algorithm is repeated for a disjoint w-subset of order-$\ell^{e/2}$ subgroups of $E_1[\ell^e]$, and so on. The running time of this time-memory tradeoff is approximately

$$(w + N)\frac{N}{w} \approx N^2/w. \tag{10}$$

One can see that this time-memory-tradeoff can be parallelized perfectly.

Another possible time-memory tradeoff is to store $(j(E_1/A_1), A_1)$, where A_1 ranges over all order-ℓ^c subgroups of $E_1[\ell^e]$ and $c \approx \log_\ell w$. Let $d = e - c$. Then, for each order-ℓ^d subgroup A_2 of $E_2[\ell^e]$, E_2/A_2 is computed and $j(E_2/A_2)$ is searched in the table. One can check that the running time of this time-memory tradeoff is approximately N^2/w, and that it can be parallelized perfectly. Note that the unit of time here is an ℓ^d-isogeny computation instead of an $\ell^{e/2}$-isogeny computation.

5.2. **Parallel collision search.** As stated in §4.3, the running time of van Oorschot-Wiener parallel collision search is approximately

$$N^{3/2}/(w^{1/2}m). \tag{11}$$

The algorithm parallelizes perfectly.

5.3. **Mesh sorting.** The mesh sorting attack is analogous to the one described by Bernstein [1] for finding hash collisions. Suppose that one has m processors arranged in a two-dimensional grid. Each processor only communicates with its neighbours in the grid. In one unit of time, each processor computes and stores pairs $(j(E_1/A_1), A_1)$, where A_1 is an order-$\ell^{e/2}$ subgroup of $E_1[\ell^e]$. Next, these stored pairs are sorted in time $\approx m^{1/2}$ (e.g., see [18]). In the next stage, a second two-dimensional grid of m processors computes and stores pairs $(j(E_2/A_2, A_2)$, where A_2 is an order-$\ell^{e/2}$ subgroup of $E_2[\ell^e]$, and the two sorted lists are compared for a match. This is repeated for a disjoint m-subset of order-$\ell^{e/2}$ subgroups A_2 until all order-$\ell^{e/2}$ subgroups of $E_2[\ell^e]$ have been tested. Then, the process is repeated for a disjoint subset of order-$\ell^{e/2}$ subgroups A_1 of $E_1[\ell^e]$ until a match is found. One can check that the total running time is approximately

$$\left(m^{1/2} + m^{1/2}\frac{N}{m}\right)\frac{N}{m} \approx N^2/m^{3/2}. \tag{12}$$

[4]A yottabyte is 2^{80} bytes.

5.4. **Targetting the 128-bit security level.** The CSSI problem is said to have a 128-bit security level if the fastest known attack has total time complexity at least 2^{128} and feasible space and hardware costs.

Suppose that $p \approx 2^{512}$, whereby $N \approx 2^{128}$; this would be a reasonable choice for the bitlength of p if the standard meet-in-the-middle attack was assessed to be the fastest (classical) algorithm for solving CSSI. However, as noted above, the storage costs for the attack are prohibitive. Instead, one should consider the time complexity of the time-memory tradeoff, parallel collision search, and mesh sorting under realistic constraints on the storage space w and the number m of processors m. Table 3 lists the calendar time[5] and the total time of these CSSI attacks for $(m, w) \in \{(2^{48}, 2^{64}), (2^{48}, 2^{80}), (2^{64}, 2^{80})\}$. One sees that in all cases the total time complexity is significantly greater than 2^{128}, even though we have ignored communication costs.

	# processors m	space w	calendar time	total time
Meet-in-the-middle	48	64	144	192
time-memory tradeoff	48	80	128	176
	64	80	112	176
Van Oorschot-Wiener	48	64	112	160
parallel collision search	48	80	104	152
	64	80	88	152
Mesh sorting	48	—	184	184
	64	—	160	160

TABLE 3. Time complexity estimates of CSSI attacks for $p \approx 2^{512}$. All numbers are expressed in their base-2 logarithms. The unit of time is an $\ell^{e/2}$-isogeny computation.

Since the total times in Table 3 are all significantly greater than 2^{128}, one can consider using smaller primes p while still achieving the 128-bit security level. Table 4 lists the calendar time and the total time of these CSSI attacks for $(m, w) \in \{(2^{48}, 2^{64}), (2^{48}, 2^{80}), (2^{64}, 2^{80})\}$ when $p \approx 2^{448}$ and $N \approx 2^{112}$. One sees that all attacks have total time complexity at least 2^{128}, even though we have ignored communication costs. We can conclude that selecting SIDH parameters with $p \approx 2^{448}$ provides 128 bits of security against known classical attacks. What remains to be determined is the security of CSSI against quantum attacks.

Remark 5. (*communication costs*) Consider the case $p \approx 2^{448}$, $e = 224$, $m = 2^{64}$, $w = 2^{80}$. From (5) and (6) we obtain $\theta \approx 1/2^{15.62}$ and an expected running time of $2^{131.7}$. For each function version, the 2^{64} processors will generate approximately $2^{48.4}$ distinguished points per unit of time (i.e., a 2^{112}-isogeny computation). So, on average, the 2^{80} storage device will be accessed $2^{48.4}$ times during each unit of time. The cost of these accesses will dominate the computational costs. Thus our security estimates, which ignore communication costs, can be regarded as being conservative.

[5] *Calendar time* is the elapsed time taken for a computation, whereas *total time* is the sum of the time expended by all m processors.

	# processors m	space w	calendar time	total time
Meet-in-the-middle	48	64	112	160
time-memory tradeoff	48	80	96	144
	64	80	80	144
Van Oorschot-Wiener	48	64	88	136
parallel collision search	48	80	80	128
	64	80	64	128
Mesh sorting	48	—	152	152
	64	—	128	128

TABLE 4. Time complexity estimates of CSSI attacks for $p \approx 2^{448}$. All numbers are expressed in their base-2 logarithms. The unit of time is an $\ell^{e/2}$-isogeny computation.

The fastest known quantum attack on CSSI is Tani's algorithm [22]. Given two generic functions $g_1 : X_1 \to Y$ and $g_2 : X_2 \to Y$, where $\#X_1 \approx \#X_2 \approx N$ and $\#Y \gg N$, Tani's quantum algorithm finds a claw, i.e., $(x_1, x_2) \in X_1 \times X_2$ such that $g_1(x_1) = g_2(x_2)$ in time $O(N^{2/3})$. The CSSI problem can be recast as a claw-finding problem by defining X_i to be the set of all degree-$\ell^{e/2}$ isogenies originating at E_i, g_i to be the function that maps a degree-$\ell^{e/2}$ isogeny originating at E_i to the j-invariant of its image curve, and $Y = R$. Since $\#X_1 = \#X_2 = N \approx p^{1/4}$, this yields an $O(p^{1/6})$-time CSSI attack.

CSSI can also be solved by an application of Grover's quantum search [9]. Recall that if $g : X \to \{0, 1\}$ is a generic function such that $g(x) = 1$ for exactly one $x \in X$, then Grover's algorithm can determine the x with $g(x) = 1$ in quantum time $O(\sqrt{\#X})$. The CSSI problem can be recast as a Grover search problem by defining X to be the set of all ordered pairs (ϕ_1, ϕ_2) of degree-$\ell^{e/2}$ isogenies originating at E_1, E_2, respectively, and defining $g(\phi_1, \phi_2)$ to be equal to 1 if and only if the j-invariants of the image curves of ϕ_1 and ϕ_2 are equal. Since $\#X = N^2 \approx p^{1/2}$, this yields an $O(p^{1/4})$-time CSSI quantum attack.

Even though Tani's algorithm has a smaller total running time than Grover's search, a careful cost analysis by Jaques and Schanck [13] reveals that Tani's algorithm is significantly costlier than Grover's search using reasonable cost measures. This is not unexpected since Tani's algorithm has $O(p^{1/6})$ space requirements, whereas the space requirements of Grover's search are negligible compared to its running time. Now, Grover's search does not parallelize perfectly — using m quantum circuits only yields a speedup by a factor of $\sqrt{m}$ and this speedup has been proven to be optimal [28]. Furthermore, it is envisioned that running long serial quantum computations will be very difficult — NIST suggests that 2^{40} is the maximum depth of a quantum circuit that can be executed in one year [14]. Thus, it can reasonably be concluded that Grover's search for the $p \approx 2^{448}$ case will be significantly costlier than parallel collision search.

We conclude that using SIDH parameters with $p \approx 2^{448}$ offers CSSI security of at least 128 bits against known classical and quantum attacks. Such a prime is significantly smaller than a 768-bit prime (such as the 751-bit prime $p = 2^{372}3^{239} - 1$ proposed in [5, 10]) that

would be needed if Tani's algorithm was deemed to be the most cost-effective CSSI attack. For example, one could select the 434-bit prime

$$p_{434} = 2^{216}3^{137} - 1;$$

this prime is balanced in the sense that $3^{137} \approx 2^{217}$, thus providing maximal resistant to Petit's SIDH attack [17].

5.5. **Targetting the 160-bit security level.** Using similar arguments as in §5.4, one surmises that SIDH parameters with $p \approx 2^{536}$ offer at least 160 bits of CSSI security against known classical (see Table 5) and quantum attacks. Such primes are significantly smaller than a 960-bit prime (such as the 964-bit prime $p = 2^{486}3^{301} - 1$ proposed in [10]) that would be needed if Tani's algorithm was deemed to be the most cost-effective CSSI attack. For example, one could select the 546-bit prime

$$p_{546} = 2^{273}3^{172} - 1;$$

this prime is nicely balanced since $3^{172} \approx 2^{273}$.

	# processors m	space w	calendar time	total time
Meet-in-the-middle	48	64	156	204
time-memory tradeoff	48	80	140	188
	64	80	124	188
Van Oorschot-Wiener	48	64	121	169
parallel collision search	48	80	113	161
	64	80	97	161
Mesh sorting	48	—	196	196
	64	—	172	172

TABLE 5. Time complexity estimates of CSSI attacks for $p \approx 2^{536}$. All numbers are expressed in their base-2 logarithms.

5.6. **SIDH performance.** The 434-bit prime $p_{434} = 2^{216}3^{137} - 1$ and the 546-bit prime $p_{546} = 2^{273}3^{177} - 1$ are significantly smaller than the 751-bit prime $p_{751} = 2^{372}3^{239} - 1$ proposed in [5, 10], and therefore can offer substantial improvements in the performance of SIDH. The boost in the performance gain is mainly due to the following.

First, since the computation of the ground field $\mathbb{F}_p$ multiplication operation has a quadratic complexity, any reduction in the size of p will result in significant savings. Since high-end processors have a word size of 64 bits, the primes p_{751}, p_{546} and p_{434} can be accommodated using twelve, nine and seven 64-bit words, respectively. Hence, if $\mathbb{F}_p$ multiplication using p_{751} can be computed in T clock cycles, then a rough estimation of the computational costs for $\mathbb{F}_p$ multiplication using p_{434} and p_{546} is as low as $0.34T$ and $0.56T$, respectively. Second, since the exponents of the primes 2 and 3 in p_{434} and p_{546} are smaller than the ones in p_{751}, the computation of the isogeny chain described in §2.2 (see Remark 1) is faster.

Table 6 lists the timings for arithmetic operations in $\mathbb{F}_p$, $\mathbb{F}_{p^2}$, and $E(\mathbb{F}_{p^2})$ for p_{434}, p_{546} and p_{751} using the SIDH library [4]. Table 7 shows that SIDH operations are about 4.8 times faster when p_{434} is used instead of p_{751} in the SIDH library of Costello et al. [5]. Also shown in Table 7 are SIDH timings for p_{751} from [7].

Domain	Operation	p_{751} $2^{372} \cdot 3^{239} - 1$	p_{434} $2^{216} \cdot 3^{137} - 1$	p_{546} $2^{273} \cdot 3^{172} - 1$
$\mathbb{F}_p$	Modular reduction	212	78	103
	Multiplication	486	180	276
	Inversion	456,621	89,406	166,405
$\mathbb{F}_{p^2}$	Multiplication	1,582	530	846
	Squaring	1,026	394	562
	Inversion	458,706	90,132	167,457
$E(\mathbb{F}_{p^2})$	Point doubling	8,855	3,105	4,664
	Point tripling	17,799	6,199	9,303
	Degree-3 isog. computation	8,537	3,147	4,605
	Degree-4 isog. computation	5,980	2,285	3,203
	Degree-3 isog. evaluation	11,864	4,040	6,169
	Degree-4 isog. evaluation	15,932	5,453	8,287

TABLE 6. Timings of selected base field, quadratic field, and elliptic curve arithmetic operations in comparison with the SIDH v2 library [4]. All timings are reported in clock cycles on an Intel Core i7-6700 supporting a Skylake micro-architecture.

6. Concluding remarks

Our implementations of the meet-in-the-middle and golden collision search CSSI attacks are, to the best of our knowledge, the first ones reported in the literature. The implementations confirm that the performance of these attacks is accurately predicted by the heuristic analysis.

Protocol Phase		Skylake			
		CLN [5]	FLOR [7]		
		p_{751}	p_{751}	p_{434}	p_{546}
Key Gen.	Alice	35.7	26.9	7.51	13.20
	Bob	39.9	30.5	8.32	14.84
Shared Secret	Alice	33.6	24.9	7.01	12.56
	Bob	38.4	28.6	7.94	14.35

TABLE 7. Performance of the SIDH protocol. All timings are reported in 10^6 clock cycles, measured on an Intel Core i7-6700 supporting a Skylake micro-architecture.

Our concrete cost analysis of the attacks leads to the conclusion that golden collision search is more effective that the meet-in-the-middle attack. Thus one can use 448-bit primes and 536-bit primes p in SIDH to achieve the 128-bit and 160-bit security levels against *known* classical attacks on the CSSI problem. We emphasize that these conclusions are based on our understanding of how to best implement these algorithms, and on assumptions on the amount of storage and the number of processors that an adversary might possess. On the other hand, our conclusions are somewhat conservative in that the analysis does not account for communication costs. Moreover, whereas it is generally accepted that the AES-128 and AES-256 block ciphers attain the 128-bit security level in the classical and quantum settings, the time it takes to compute a degree-2^{112} isogeny (which is the unit of time for the CSSI attack with balanced 448-bit prime p) is considerably greater than the time for one application of AES-128 or AES-256.

Our analysis in combination with the cost analysis in [13] of the *known* quantum attacks on CSSI leads to the conclusion that van Oorschot-Wiener parallel collision search is the most powerful attack on the CSSI problem. Thus, 448- and 536-bit primes p offer 128- and 160-bits of SIDH security against all *known* classical and quantum attacks.

References

[1] D. Bernstein, "Cost analysis of hash collisions: Will quantum computers make SHARCS obsolete?", In: Workshop Record of SHARCS'09: Special-purpose Hardware for Attacking Cryptographic Systems, 2009. Available from https://cr.yp.to/papers.html#collisioncost.

[2] G. Brassard, P. Høyer and A. Tapp, "Quantum cryptanalysis of hash and claw-free functions", *Latin American Symposium on Theoretical Informatics — LATIN'98*, LNCS 1380 (1998), 163–169.

[3] D. Charles, E. Goren and K. Lauter, "Cryptographic hash functions from expander graphs", *Journal of Cryptology*, 22 (2009), 93–113.

[4] C. Costello et al., SIDH Library, https://www.microsoft.com/en-us/research/project/sidh-library/.

[5] C. Costello, P. Longa and M. Naehrig, "Efficient algorithms for supersingular isogeny Diffie-Hellman", *Advances in Cryptology — CRYPTO 2016*, LNCS 9814 (2016), 572–601.

[6] L. De Feo, D. Jao and J. Plût, "Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies", *Journal of Mathematical Cryptology*, 8 (2014), 209–247.

[7] A. Faz-Hernández, J. López, E. Ochoa-Jiménez and F. Rodríguez-Henríquez, "A faster software implementation of the supersingular isogeny Diffie-Hellman key exchange protocol", *IEEE Transactions on Computers*, to appear; also available from http://eprint.iacr.org/2017/1015.

[8] S. Galbraith, C. Petit and J. Silva, "Identification protocols and signature schemes based on supersingular isogeny problems", *Advances in Cryptology — ASIACRYPT 2017*, LNCS 10624 (2017), 3–33.

[9] L. Grover, "A fast quantum mechanical algorithm for database search", *Proceedings of the Twenty-Eighth Annual Symposium on Theory of Computing — STOC '96*, ACM Press (1996), 212–219.

[10] D. Jao et al., "Supersingular isogeny key encapsulation", Round 1 submission, NIST Post-Quantum Cryptography Standardization, November 30, 2017.

[11] D. Jao and L. De Feo, "Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies", *Post-Quantum Cryptography — PQCrypto 2011*, LNCS 7071 (2011), 19–34.

[12] D. Jao and V. Soukharev, "Isogeny-based quantum-resistant undeniable signatures", *Post-Quantum Cryptography — PQCrypto 2014*, LNCS 8772 (2014), 160–179.

[13] S. Jaques and J. Schanck, "Cost analyses of Tani's algorithm", in preparation.

[14] National Institute of Standards and Technology, "Submission requirements and evaluation criteria for the post-quantum cryptography standardization process", December 2016. Available from https://csrc.nist.gov/csrc/media/projects/post-quantum-cryptography/documents/call-for-proposals-final-dec-2016.pdf.

[15] P. van Oorschot and M. Wiener, "Improving implementable meet-in-the-middle attacks by orders of magnitude", *Advances in Cryptology — CRYPTO '96*, LNCS 1109 (1996), 229–236.

[16] P. van Oorschot and M. Wiener, "Parallel collision search with cryptanalytic applications", *Journal of Cryptology*, 12 (1999), 1–28.
[17] C. Petit, "Faster algorithms for isogeny problems using torsion point images", *Advances in Cryptology — ASIACRYPT 2017*, LNCS 10625 (2017), 330–353.
[18] C. Schnorr and A. Shamir, "An optimal sorting algorithm for mesh connected computers", *Proceedings of the Eighteenth Annual Symposium on Theory of Computing — STOC '86*, ACM Press (1986), 255–263.
[19] R. Schoof, "Nonsingular plane cubic curves over finite fields", *Journal of Combinatorial Theory, Series A*, 46 (1987), 183–211.
[20] A. Shamir, "Factoring large numbers with the TWINKLE device", *Cryptographic Hardware and Embedded Systems — CHES 1999*, LNCS 1717 (1999), 2–12.
[21] A. Shamir and E. Tromer, "Factoring large numbers with the TWIRL device", *Advances in Cryptology — CRYPTO 2003*, LNCS 2729 (2003), 1–26.
[22] S. Tani, "Claw finding algorithms using quantum walk", *Theoretical Computer Science*, 410 (2009), 5285–5297.
[23] J. Vélu, "Isogénies entre courbes elliptiques", *C. R. Acad. Sc. Paris*, 273 (1971), 238–241.
[24] L. Washington, *Elliptic Curves: Number Theory and Cryptography*. Chapman & Hall/CRC, second edition, 2008.
[25] Wikipedia, "Sunway TaihuLight", https://en.wikipedia.org/wiki/Sunway_TaihuLight.
[26] Wikipedia, "Exabyte", https://en.wikipedia.org/wiki/Exabyte#Google.
[27] Y. Yoo, R. Azarderakhsh, A. Jalali, D. Jao and V. Soukharev, "A post-quantum digital signature scheme based on supersingular isogenies", *Financial Cryptography and Data Security — FC 2017*, LNCS 10322 (2018), 163–181.
[28] C. Zalka, "Grover's quantum searching algorithm is optimal", *Physical Review A*, 60 (1999), 2746–2751.

Department of Combinatorics & Optimization, University of Waterloo, Canada
E-mail address: gora.adj@gmail.com

Computer Science Department, CINVESTAV-IPN
E-mail address: dcervantes@computacion.cs.cinvestav.mx

Computer Science Department, CINVESTAV-IPN
E-mail address: jjchi@computacion.cs.cinvestav.mx

Department of Combinatorics & Optimization, University of Waterloo, Canada
E-mail address: ajmeneze@uwaterloo.ca

Computer Science Department, CINVESTAV-IPN
E-mail address: francisco@cs.cinvestav.mx