J. Math. Cryptol. **8** (2014), 209–247
DOI 10.1515/jmc-2012-0015

Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies

Luca De Feo, David Jao and Jérôme Plût

Communicated by Neal Koblitz

Abstract. We present new candidates for quantum-resistant public-key cryptosystems based on the conjectured difficulty of finding isogenies between supersingular elliptic curves. The main technical idea in our scheme is that we transmit the images of torsion bases under the isogeny in order to allow the parties to construct a shared commutative square despite the non-commutativity of the endomorphism ring. We give a precise formulation of the necessary computational assumptions along with a discussion of their validity, and prove the security of our protocols under these assumptions. In addition, we present implementation results showing that our protocols are multiple orders of magnitude faster than previous isogeny-based cryptosystems over ordinary curves. This paper is an extended version of [19]. We add a new zero-knowledge identification scheme and detailed security proofs for the protocols. We also present a new, asymptotically faster, algorithm for key generation, a thorough study of its optimization, and new experimental data.

Keywords. Elliptic curves, isogenies, quantum-resistant cryptosystems.

2010 Mathematics Subject Classification. 94A60, 14G50, 11Y16, 14K02.

1 Introduction

The Diffie–Hellman scheme is a fundamental protocol for public-key exchange between two parties. Its original definition over finite fields is based on the hardness of computing the map $g, g^a, g^b \mapsto g^{ab}$ for $g \in \mathbb{F}_p^*$. Recently, Stolbunov [44] proposed a Diffie–Hellman type system based on the difficulty of computing isogenies between ordinary elliptic curves, with the stated aim of obtaining quantum-resistant cryptographic protocols. The fastest known (classical) probabilistic algorithm for solving this problem is the algorithm of Galbraith and Stolbunov [17], based on the algorithm of Galbraith, Hess, and Smart [16]. This algorithm is exponential, with a worst-case running time of $O(\sqrt[4]{q})$. However, on a quantum

This work is supported in part by NSERC CRD grant CRDPJ 405857-10 and by the French Agence Nationale de la Recherche through the ECLIPSES project under contract ANR-09-VERS-018.

computer, recent work of Childs et al. [10] showed that the private keys in Stolbunov's system can be recovered in subexponential time. Moreover, even if we only use classical attacks in assessing security levels, Stolbunov's scheme requires 229 seconds (even with precomputation) to perform one key exchange operation at the 128-bit security level on a desktop PC [44, Table 1].

In this work we present isogeny-based key-exchange, encryption, and identification schemes that address both the performance and security drawbacks of Stolbunov's system. Our primitive achieves performance on the order of 60 milliseconds (cf. Section 7) at the 128-bit security level (as measured against the fastest known quantum attacks) using desktop PCs, making our schemes far faster than Stolbunov's. In terms of security, our schemes are not vulnerable to the algorithm of Childs et al. [10], nor to any algorithm of this type, since they are not based on a group action. The fastest known attacks against our schemes, even on quantum computers, require fully exponential time. Our schemes involve new computational assumptions upon which their quantum resistance is based, and like all new computational assumptions, further study and the passage of time is needed for validation. Nevertheless, we believe our proposal represents a promising candidate for quantum-resistant isogeny-based public-key cryptography.

Our proposal, presented in Section 3, uses isogenies between *supersingular* elliptic curves rather than ordinary elliptic curves. The main technical difficulty is that, in the supersingular case, the endomorphism ring is non-commutative, whereas Diffie–Hellman type protocols require commutativity. We show how to overcome this obstacle by providing the outputs of the isogeny on certain points as auxiliary input to the protocols. To the best of our knowledge, nothing similar to this idea has ever previously appeared in the literature. Providing this auxiliary input does not seem to make the problem of finding isogenies any easier, and the added difficulty arising from non-commutativity seems to make our protocols stronger; see Section 5.1 for a full discussion. The multiple orders of magnitude of performance gains in our scheme arise from the fact that supersingular isogeny graphs are much faster to navigate than ordinary graphs, as described in Section 4. In Sections 5 and 6 we provide formal statements of the hardness assumptions and security reductions for our system. Finally, in Section 7 we present implementation results confirming the correctness and performance of our protocol.

2 Isogenies

Let E_1 and E_2 be elliptic curves defined over a finite field $\mathbb{F}_q$. An isogeny $\phi : E_1 \to E_2$ defined over $\mathbb{F}_q$ is a non-constant rational map defined over $\mathbb{F}_q$ which is also a group homomorphism from $E_1(\mathbb{F}_q)$ to $E_2(\mathbb{F}_q)$ (see [38, Section III.4]).

The degree of an isogeny is its degree as a rational map. For separable isogenies, having degree ℓ implies that the kernel of the isogeny has cardinality ℓ. Every isogeny of degree greater than 1 can be factored into a composition of isogenies of prime degree over $\bar{\mathbb{F}}_q$ (see [11]).

An endomorphism of an elliptic curve E defined over $\mathbb{F}_q$ is an isogeny $E \to E$ defined over $\mathbb{F}_{q^m}$ for some m. The set of endomorphisms of E together with the zero map forms a ring under the operations of pointwise addition and composition; this ring is called the endomorphism ring of E and denoted by $\mathrm{End}(E)$. The ring $\mathrm{End}(E)$ is isomorphic either to an order in a quaternion algebra or to an order in an imaginary quadratic field [38, Theorem V.3.1]; in the first case we say E is supersingular and in the second case we say E is ordinary.

Two elliptic curves E_1 and E_2 defined over $\mathbb{F}_q$ are said to be isogenous over $\mathbb{F}_q$ if there exists an isogeny $\phi\colon E_1 \to E_2$ defined over $\mathbb{F}_q$. A theorem of Tate states that two curves E_1 and E_2 are isogenous over $\mathbb{F}_q$ if and only if $\#E_1(\mathbb{F}_q) = \#E_2(\mathbb{F}_q)$ (see [46, §3]). Since every isogeny has a dual isogeny [38, Theorem III.6.1], the property of being isogenous over $\mathbb{F}_q$ is an equivalence relation on the finite set of $\bar{\mathbb{F}}_q$-isomorphism classes of elliptic curves defined over $\mathbb{F}_q$. Accordingly, we define an isogeny class to be an equivalence class of elliptic curves, taken up to $\bar{\mathbb{F}}_q$-isomorphism, under this equivalence relation.

The ℓ-torsion group of E, denoted by $E[\ell]$, is the set of all points $P \in E(\bar{\mathbb{F}}_q)$ such that ℓP is the identity. For ℓ such that $p \nmid \ell$, we have $E[\ell] \cong \mathbb{Z}/\ell\mathbb{Z} \oplus \mathbb{Z}/\ell\mathbb{Z}$.

Curves in the same isogeny class are either all supersingular or all ordinary. We assume for the remainder of this paper that we are in the supersingular case. Because we only care about curves up to isomorphism, and every supersingular curve is isomorphic to a curve defined over the field $\mathbb{F}_{p^2}$, we limit ourselves to this base field.

For every prime $\ell \nmid p$, there exist exactly $\ell + 1$ isogenies (counting multiplicities) of degree ℓ originating from any given such supersingular curve. Given an elliptic curve E and a finite subgroup Φ of E, there is up to isomorphism a unique isogeny $E \to E'$ having kernel Φ (see [38, Proposition III.4.12]). Hence we can identify an isogeny by specifying its kernel, and conversely given a kernel subgroup the corresponding isogeny can be computed using Vélu's formulas [48]. Typically, this correspondence is of little use, since the kernel, or any representation thereof, is usually as unwieldy as the isogeny itself. However, in the special case of kernels generated by $\mathbb{F}_{p^2}$-rational points of smooth order, specifying a generator of the kernel allows for compact representation and efficient computation of the corresponding isogeny, as we demonstrate in Section 4.

2.1 Ramanujan graphs

Let $G = (\mathcal{V}, \mathcal{E})$ be a finite graph on h vertices $\mathcal{V}$ with undirected edges $\mathcal{E}$. Suppose G is a regular graph of degree k, i.e., exactly k edges meet at each vertex. Given a labeling of the vertices $\mathcal{V} = \{v_1, \ldots, v_h\}$, the adjacency matrix of G is the symmetric $h \times h$ matrix A whose ij-th entry $A_{i,j}$ is 1 if an edge exists between v_i and v_j, and 0 otherwise.

It is convenient to identify functions on $\mathcal{V}$ with vectors in $\mathbb{R}^h$ via this labeling, and therefore also think of A as a self-adjoint operator on $L^2(\mathcal{V})$. All of the eigenvalues of A satisfy the bound $|\lambda| \leq k$. Constant vectors are eigenfunctions of A with eigenvalue k, which for obvious reasons is called the trivial eigenvalue λ_{triv}. A family of such graphs G with $h \to \infty$ is said to be a sequence of *expander graphs* if all other eigenvalues of their adjacency matrices are bounded away from $\lambda_{\text{triv}} = k$ by a fixed amount.[1] In particular, no other eigenvalue is equal to k; this implies the graph is connected. A Ramanujan graph is a special type of expander which has $|\lambda| \leq 2\sqrt{k-1}$ for any non-trivial eigenvalue which is not equal to $-k$ (this last possibility happens if and only if the graph is bipartite). The Ramanujan property was first defined in [26]. It characterizes the optimal separation between the two largest eigenvalues of the graph adjacency matrix, and implies the expansion property.

A fundamental use of expanders is to prove the rapid mixing of the random walk on $\mathcal{V}$ along the edges $\mathcal{E}$. The following rapid mixing result is standard but we present it below for completeness. For the proof, see [20] or [12, 25, 36].

Proposition 2.1. Let G be a regular graph of degree k on h vertices. Suppose that the eigenvalue λ of any non-constant eigenvector satisfies the bound $|\lambda| \leq c$ for some $c < k$. Let S be any subset of the vertices of G, and x be any vertex in G. Then a random walk of length at least $\frac{\log 2h/|S|^{1/2}}{\log k/c}$ starting from x will land in S with probability at least $\frac{|S|}{2h} = \frac{|S|}{2|G|}$.

2.2 Isogeny graphs

An isogeny graph is a graph whose nodes consist of all elliptic curves in $\mathbb{F}_q$ belonging to a fixed isogeny class, up to $\bar{\mathbb{F}}_q$-isomorphism (so that two elliptic curves which are isomorphic over $\bar{\mathbb{F}}_q$ represent the same node in the graph). In practice, the nodes are represented using j-invariants, which are invariant up to isomorphism. Isogeny graphs for supersingular elliptic curves were first considered by Mestre [27], and were shown by Pizer [32, 33] to have the Ramanujan property.

[1] Expansion is usually phrased in terms of the number of neighbors of subsets of G, but the spectral condition here is equivalent for k-regular graphs and also more useful for our purposes.

Every supersingular elliptic curve in characteristic p is defined over either $\mathbb{F}_p$ or $\mathbb{F}_{p^2}$ (see [38]), so it suffices to fix $\mathbb{F}_q = \mathbb{F}_{p^2}$ as the field of definition for this discussion. Thus, in contrast to ordinary curves, there are a finite number of isomorphism classes of supersingular curves in any given isogeny class; this number is in fact $g + 1$, where g is the genus of the modular curve $X_0(p)$, which is roughly $p/12$. It turns out that all supersingular curves defined over $\mathbb{F}_{p^2}$ belong to the same isogeny class [27]. For a fixed prime value of $\ell \neq p$, we define the vertices of the supersingular isogeny graph $\mathcal{G}$ to consist of these g isomorphism classes of curves, with edges given by isomorphism classes of degree-ℓ isogenies, defined as follows: two isogenies $\phi_1, \phi_2 \colon E_i \to E_j$ are isomorphic if there exists an automorphism $\alpha \in \mathrm{Aut}(E_j)$ (i.e., an invertible endomorphism) such that $\phi_2 = \alpha\phi_1$. Pizer [32, 33] has shown that $\mathcal{G}$ is a connected $k = \ell + 1$-regular multigraph satisfying the Ramanujan bound of $|\lambda| \leq 2\sqrt{\ell} = 2\sqrt{k-1}$ for the non-trivial eigenvalues of its adjacency matrix.

3 Public-key cryptosystems based on supersingular curves

In this section we present a key-exchange protocol and a public-key cryptosystem analogous to those of [35, 44], and a zero-knowledge identification scheme, all using supersingular elliptic curves.

Our protocols require supersingular curves of smooth order. Such curves are normally unsuitable for cryptography since discrete logarithms on them are easy. However, since the discrete logarithm problem is unimportant in our setting, this issue does not affect us. In the supersingular setting, it is easy to construct curves of smooth order, and using a smooth order curve will give a large number of isogenies that are fast to compute. Specifically, we fix $\mathbb{F}_q = \mathbb{F}_{p^2}$ as the field of definition, where p is a prime of the form $\ell_A^{e_A}\ell_B^{e_B} \cdot f \pm 1$. Here ℓ_A and ℓ_B are small primes, and f is a cofactor such that p is prime. Then we construct a supersingular curve E defined over $\mathbb{F}_q$ of cardinality $(\ell_A^{e_A}\ell_B^{e_B} f)^2$. By construction, $E[\ell_A^{e_A}]$ is $\mathbb{F}_q$-rational and contains $\ell_A^{e_A-1}(\ell_A + 1)$ cyclic subgroups of order $\ell_A^{e_A}$, each defining a different isogeny; the analogous statement holds for $E[\ell_B^{e_B}]$.

Our protocols revolve around the following commutative diagram:

$$\begin{array}{ccc} E & \xrightarrow{\phi} & E/\langle P\rangle \\ {\scriptstyle\psi}\downarrow & & \downarrow \\ E/\langle Q\rangle & \to & E/\langle P, Q\rangle \end{array} \tag{3.1}$$

where ϕ and ψ are random walks in the graphs of isogenies of degrees ℓ_A and ℓ_B, respectively. Their security is based on the difficulty of finding a path connecting

two given vertices in a graph of supersingular isogenies. We refer to Section 4 for low-level algorithmic details, and Section 5 for a full discussion of security.

3.1 Zero-knowledge proof of identity

We begin with the protocol which is easiest to understand. Peggy knows a cyclic degree $\ell_A^{e_A}$ isogeny $\phi : E \to E/\langle S\rangle$, with the curves E and $E/\langle S\rangle$ publicly known, and wants to prove to Vic that she knows a generator for $\langle S\rangle$, without revealing it.

Our protocol is loosely inspired by the zero-knowledge proof of membership for Graph Isomorphism [18]. In that protocol, Peggy shows that she knows a graph isomorphism $G \simeq G'$ by first publishing a random H such that the following diagram commutes:

$$\begin{array}{ccc} G & \longleftrightarrow & G' \\ & {\scriptstyle\phi}\searrow \quad \swarrow{\scriptstyle\psi} & \\ & H & \end{array}$$

and then revealing only one among ϕ and ψ. Intuitively, this protocol is *perfectly* zero-knowledge because the information that Peggy reveals (i.e., a random permutation of G or G') could be easily computed by anyone without her help.

In an analogous way, our protocol consists in publishing the vertices of diagram (3.1), and then revealing some, but not all, of its arrows. Unlike the case of Graph Isomorphism, in our protocol Peggy needs to use her secret knowledge to create the diagram, thus we cannot achieve a *perfect* zero-knowledge. Nevertheless, we will show in Section 6 that, under suitable assumptions, our protocol is *computationally* zero-knowledge.

We show below the diagram used in our protocol. $\langle S\rangle$ is the kernel of the secret isogeny ϕ of degree $\ell_A^{e_A}$, while $\langle R\rangle$ is a cyclic group of order $\ell_B^{e_B}$.

$$\begin{array}{ccc} E & \xrightarrow{\phi} & E/\langle S\rangle \\ {\scriptstyle\psi}\downarrow & & \downarrow{\scriptstyle\psi'} \\ E/\langle R\rangle & \xrightarrow{\phi'} & E/\langle S, R\rangle \end{array} \tag{3.2}$$

Peggy can compute the diagram as follows:

- She uses Vélu's formulas to compute the isogeny $\psi : E \to E/\langle R\rangle$.
- She computes $R' = \phi(R)$ and the isogeny $\psi' : E/\langle S\rangle \to E/\langle S, R\rangle$.
- She computes $S' = \psi(S)$ and the isogeny $\phi' : E/\langle R\rangle \to E/\langle S, R\rangle$.

Now, the natural question is: which arrows of the diagram can Peggy reveal without compromising her secret ϕ? It is not hard to see, and we will show it in Theorem 6.3, that the knowledge of (ψ, ϕ') or (ψ', ϕ') allows anyone to compute the kernel of ϕ. However, we will argue that there is no obvious way to compute ϕ from the sole knowledge of ϕ'. Revealing one of ψ or ψ' is no problem either, however revealing (ψ, ψ') altogether is more subtle. Indeed, revealing the points R and $\phi(R)$ uncovers some information on the action of ϕ on $E[\ell_B^{e_B}]$: it is to be expected that after a few iterations Peggy will reveal a basis (P, Q) of $E[\ell_B^{e_B}]$ and the respective images $\phi(P)$, $\phi(Q)$, thus allowing anyone to evaluate ϕ on the whole $E[\ell_B^{e_B}]$. Nevertheless, we conjecture that this leakage does not compromise Peggy's secret either, and we make these data part of the public parameters.[2]

Finally, we present our protocol.

Secret parameters. A supersingular curve E defined over $\mathbb{F}_q$ and a primitive $\ell_A^{e_A}$-torsion point S defining an isogeny $\phi : E \to E/\langle S \rangle$.

Public parameters. The curves E and $E/\langle S \rangle$. Generators P, Q of $E[\ell_B^{e_B}]$ and their images $\phi(P), \phi(Q)$.

Identification. Repeat m times:

(i) Peggy chooses a random primitive $\ell_B^{e_B}$-torsion point R and computes diagram (3.2).

(ii) Peggy sends the curves $E_1 = E/\langle R \rangle$ and $E_2 = E/\langle S, R \rangle$ to Vic.

(iii) Vic selects a random bit b and sends it to Peggy.

(iv) If $b = 0$, Peggy reveals the points R and $\phi(R')$. Vic accepts if they have order $\ell_b^{e_B}$ and generate the kernels of isogenies $E \to E_1$ and $E/\langle S \rangle \to E_2$, respectively.

(v) If $b = 1$, Peggy reveals the point $\psi(S)$. Vic accepts if it has order $\ell_A^{e_A}$ and generates the kernel of an isogeny $E_1 \to E_2$.

3.2 Key exchange

The key exchange protocol is a variation à la Diffie–Hellman over diagram (3.1). The idea is to let Alice choose ϕ, while Bob chooses ψ. Although similar in spirit to the protocol based on the action of the class group on ordinary elliptic curves of [44], a main technical difference is that, since ideal classes no longer commute (or

[2] An alternative solution, that intuitively leaks less information on ϕ, would be to publish random generators of $\langle R \rangle$ and $\langle \phi(R) \rangle$. However, it is not clear that this idea would considerably improve the security of the protocol, and we will not pursue it further for coherence with the protocols that will follow.

indeed even multiply together) in the supersingular case, extra information must be communicated as part of the protocol in order to ensure that both parties arrive at the same common value.

We fix as public parameters a supersingular curve E_0 defined over $\mathbb{F}_{p^2}$, and bases $\{P_A, Q_A\}$ and $\{P_B, Q_B\}$ which generate $E_0[\ell_A^{e_A}]$ and $E_0[\ell_B^{e_B}]$, respectively, so that $\langle P_A, Q_A\rangle = E_0[\ell_A^{e_A}]$ and $\langle P_B, Q_B\rangle = E_0[\ell_B^{e_B}]$. Alice chooses two random elements $m_A, n_A \in_R \mathbb{Z}/\ell_A^{e_A}\mathbb{Z}$, not both divisible by ℓ_A, and computes an isogeny $\phi_A\colon E_0 \to E_A$ with kernel $K_A := \langle [m_A]P_A + [n_A]Q_A\rangle$. Alice also computes the image $\{\phi_A(P_B), \phi_A(Q_B)\} \subset E_A$ of the basis $\{P_B, Q_B\}$ for $E_0[\ell_B^{e_B}]$ under her secret isogeny ϕ_A, and sends these points to Bob together with E_A. Similarly, Bob selects random elements $m_B, n_B \in_R \mathbb{Z}/\ell_B^{e_B}\mathbb{Z}$ and computes an isogeny $\phi_B\colon E_0 \to E_B$ having kernel $K_B := \langle [m_B]P_B + [n_B]Q_B\rangle$, along with the points $\{\phi_B(P_A), \phi_B(Q_A)\}$. Upon receipt of E_B and $\phi_B(P_A), \phi_B(Q_A) \in E_B$ from Bob, Alice computes an isogeny $\phi'_A\colon E_B \to E_{AB}$ having kernel equal to $\langle [m_A]\phi_B(P_A) + [n_A]\phi_B(Q_A)\rangle$; Bob proceeds *mutatis mutandis*. Alice and Bob can then use the common j-invariant of

$$\begin{aligned} E_{AB} &= \phi'_B(\phi_A(E_0)) = \phi'_A(\phi_B(E_0)) \\ &= E_0/\langle [m_A]P_A + [n_A]Q_A, [m_B]P_B + [n_B]Q_B\rangle \end{aligned}$$

to form a secret shared key.

The full protocol is given in Figure 1. We denote by A and B the identifiers of Alice and Bob, and use sID to denote the unique session identifier.

3.3 Public-key encryption

The key-exchange protocol of Section 3.2 can easily be adapted to yield a public-key cryptosystem, in much the same way as Elgamal encryption follows from Diffie–Hellman. We briefly give the details here. All notation is the same as above. Stolbunov [44] uses a similar construction, upon which ours is based.

Setup. Choose $p = \ell_A^{e_A}\ell_B^{e_B} \cdot f \pm 1$, E_0, $\{P_A, Q_A\}$, $\{P_B, Q_B\}$ as above. Let $\mathcal{H} = \{H_k : k \in K\}$ be a hash function family indexed by a finite set K, where each H_k is a function from $\mathbb{F}_{p^2}$ to the message space $\{0,1\}^w$.

Key generation. Choose two random elements $m_A, n_A \in_R \mathbb{Z}/\ell_A^{e_A}\mathbb{Z}$, not both divisible by ℓ_A. Compute $E_A, \phi_A(P_B), \phi_A(Q_B)$ as above, and choose a random element $k \in_R K$. The public key is the tuple $(E_A, \phi_A(P_B), \phi_A(Q_B), k)$ and the private key is (m_A, n_A, k).

Encryption. Given a public key $(E_A, \phi_A(P_B), \phi_A(Q_B), k)$ and a message $m \in \{0,1\}^w$, choose two random elements $m_B, n_B \in_R \mathbb{Z}/\ell_B^{e_B}\mathbb{Z}$, not both divisible by

$\mathcal{A}$		$\mathcal{B}$
Input: A, B, sID		**Input:** B
$m_A, n_A \in_R \mathbb{Z}/\ell_A^{e_A}\mathbb{Z}$		$m_B, n_B \in_R \mathbb{Z}/\ell_B^{e_B}\mathbb{Z}$
$\phi_A := E_0/\langle [m_A]P_A + [n_A]Q_A\rangle$		$\phi_B := E_0/\langle [m_B]P_B + [n_B]Q_B\rangle$
	$\xrightarrow{A,\text{sID},\ \phi_A(P_B),\ \phi_A(Q_B),\ E_A}$	
	$\xleftarrow{B,\text{sID},\ \phi_B(P_A),\ \phi_B(Q_A),\ E_B}$	
$E_{AB} :=$		$E_{BA} :=$
$E_B/\langle [m_A]\phi_B(P_A) + [n_A]\phi_B(Q_A)\rangle$		$E_A/\langle [m_B]\phi_A(P_B) + [n_B]\phi_A(Q_B)\rangle$
Output: $j(E_{AB})$, sID		**Output:** $j(E_{BA})$, sID

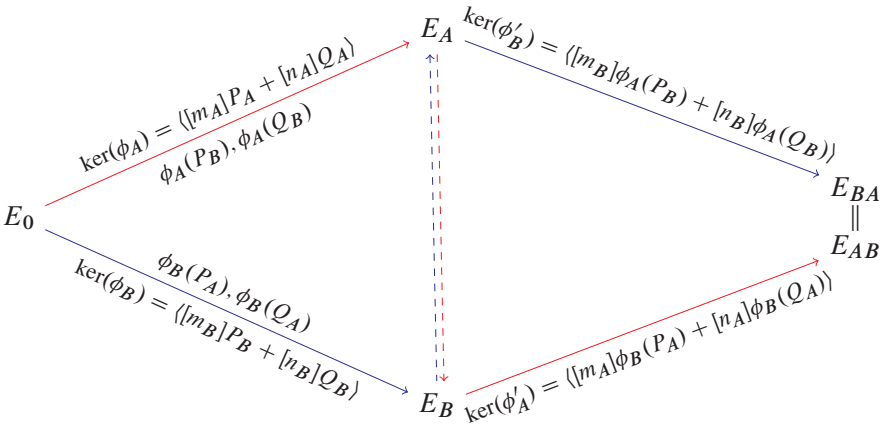

Figure 1. Key-exchange protocol using isogenies on supersingular curves.

ℓ_B, and compute

$$h = H_k(j(E_{AB})), \quad c = h \oplus m.$$

The ciphertext is $(E_B, \phi_B(P_A), \phi_B(Q_A), c)$.

Decryption. Given a ciphertext $(E_B, \phi_B(P_A), \phi_B(Q_A), c)$ together with a private key (m_A, n_A, k), compute the j-invariant $j(E_{AB})$ and set

$$h = H_k(j(E_{AB})), \quad m = h \oplus c.$$

The plaintext is m.

4 Algorithmic aspects

We now give specific algorithms to implement the aforementioned steps efficiently.

4.1 Parameter generation

For any fixed choice of $\ell_A^{e_A}$ and $\ell_B^{e_B}$, one can easily test random values of f (of any desired cryptographic size) until a value is found for which $p = \ell_A^{e_A}\ell_B^{e_B} \cdot f - 1$ or $p = \ell_A^{e_A}\ell_B^{e_B} \cdot f + 1$ is prime. The prime number theorem in arithmetic progressions (specifically, the effective version of Lagarias and Odlyzko [24]) provides a sufficient lower bound for the density of such primes.

Once the prime $p = \ell_A^{e_A}\ell_B^{e_B} \cdot f \pm 1$ is known, Bröker [6] has shown that it is easy to find a supersingular curve E over $\mathbb{F}_{p^2}$ having cardinality $(p \mp 1)^2 = (\ell_A^{e_A}\ell_B^{e_B} \cdot f)^2$. Starting from E, one can select a random supersingular curve E_0 over $\mathbb{F}_{p^2}$ by means of random walks on the isogeny graph (cf. Proposition 2.1); alternatively, one can simply take $E_0 = E$. In either case, E_0 has group structure $(\mathbb{Z}/(p\mp1)\mathbb{Z})^2$. To find a basis for $E_0[\ell_A^{e_A}]$, choose a random point $P \in_R E_0(\mathbb{F}_{p^2})$ and multiply it by $(\ell_B^{e_B} \cdot f)^2$ to obtain a point P' of order dividing $\ell_A^{e_A}$. With high probability, P' will have order exactly $\ell_A^{e_A}$; one can of course check this by multiplying P' by powers of ℓ_A. If the check succeeds, then set $P_A = P'$; otherwise try again with another P. A second point Q_A of order $\ell_A^{e_A}$ can be obtained in the same way. To check whether Q_A is independent of P_A, simply compute the Weil pairing $e(P_A, Q_A)$ in $E[\ell_A^{e_A}]$ and check that the result has order $\ell_A^{e_A}$; as before, this happens with high probability, and if not, just choose another point Q_A. Note that the choice of basis has no effect on the security of the scheme, since one can convert from one basis to another using extended discrete logarithms, which are easy to compute in $E_0[\ell_A^{e_A}]$ by [47].

4.2 Key exchange and other protocols

The key exchange is performed in two rounds. In each round Alice and Bob do the following operations on each side:

(i) Compute $\langle R\rangle = \langle [m]P + [n]Q\rangle$ for some points P, Q.

(ii) Compute the isogeny $\phi : E \to E/\langle R\rangle$ for a curve E.

(iii) In the first round (only), compute $\phi(R)$ and $\phi(S)$ for some points R, S.

Here the curve E and the points P, Q, R, S depend on the round and the player, as shown in Figure 1. Similar operations are needed by the other protocols of Section 3. We demonstrate how to implement efficiently each of these steps.

4.2.1 Computing $\langle [m]P + [n]Q \rangle$

We first observe that it suffices to compute any generator of $\langle [m]P + [n]Q \rangle$. Without loss of generality we can assume that m is invertible modulo the order of the group, in which case $R' = P + [m^{-1}n]Q$ is one such generator. Computing R' by a standard double-and-add approach requires half the effort of computing $[m]P + [n]Q$ naively (see [1, 13, 39], though, for better ways of computing the latter).

However, computing $P + [m^{-1}n]Q$ by double-and-add has one major drawback: it is trivially vulnerable to simple power analysis (SPA) [22]. To avoid SPA, one can use a Montgomery ladder [28] to compute $[m^{-1}n]Q$, and then add P, but this is significantly slower.

Instead, we propose in Algorithm 1 a much more efficient ladder, computing $P + [m^{-1}n]Q$ directly. The idea behind it is simple: at each iteration the registers A, B and C contain respectively the values $[x]Q$, $[x+1]Q$ and $P + [x]Q$, for x equal to the leftmost bits of $m^{-1}n$. The function $\text{dadd}(A, B, C)$ is a *differential addition*: it computes the sum $A + B$ knowing $C = A - B$. Montgomery curves have a very efficient differential addition [28], making the implementation of our ladder as efficient as the naive double-and-add on twisted Edwards curves (see Section 4.3).

Algorithm 1 Three-point ladder to compute $P + [t]Q$.

Input: t, P, Q

Set $A = 0$, $B = Q$, $C = P$.
Compute $Q - P$.
for i decreasing **from** $|t|$ **to** 1 **do**
  Let t_i be the i-th bit of t.
  if $t_i = 0$ **then**
    $A = 2A$, $B = \text{dadd}(A, B, Q)$, $C = \text{dadd}(A, C, P)$.
  else
    $A = \text{dadd}(A, B, Q)$, $B = 2B$, $C = \text{dadd}(B, C, Q - P)$.
  end if
end for

Output: $C = P + [t]Q$

4.2.2 Computing smooth degree isogenies

It remains to describe how Alice and Bob can compute and evaluate the isogenies. Let E be an elliptic curve, and let R be a point of order ℓ^e. Our goal is to compute

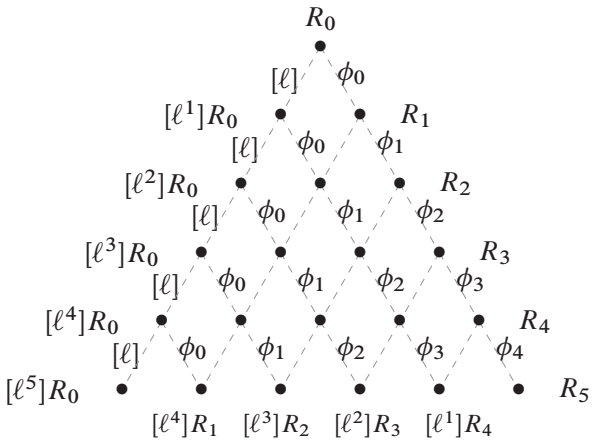

Figure 2. Computational structure of the construction of $\phi = \phi_5 \circ \cdots \circ \phi_0$.

the image curve $E/\langle R\rangle$ and to evaluate the isogeny $\phi : E \to E/\langle R\rangle$ at some points of E.

Since the degree of ϕ is smooth, it is best to decompose it as a chain of ℓ-isogenies. Set $E_0 = E$, $R_0 = R$ and, for $0 \leq i < e$, let

$$E_{i+1} = E_i/\langle \ell^{e-i-1} R_i\rangle, \quad \phi_i : E_i \to E_{i+1}, \quad R_{i+1} = \phi_i(R_i).$$

Then $E/\langle R\rangle = E_e$ and $\phi = \phi_{e-1} \circ \cdots \circ \phi_0$.

The curve E_{i+1} and the isogeny ϕ_i can be computed using Vélu's formulas [48], once the ℓ-torsion subgroup $\langle R_i\rangle$ of E_i is known. This immediately suggests two strategies having quadratic complexity in e, as described in [19].

However, we can do much better. Figure 2 summarizes the computational structure of the problem for $e = 6$. Bullets represent points, with points on the same horizontal level having the same order, and points on the same left diagonal belonging to the same curve. Dashed edges are directed and go from top to bottom; leftward edges represent multiplication by ℓ, and rightward edges represent evaluation of an ℓ-isogeny.

At the beginning of the algorithm, only the point R_0 is known. Our goal is to compute all the points on the bottom line. Indeed, from the knowledge of the point $[\ell^{e-i-1}]R_i$ we can compute the kernel of ϕ_i using $O(\ell)$ point additions. We then apply Vélu's formulas to compute ϕ_i and E_{i+1}. From this point on, we can forget about the number theoretic nature of the problem and purely focus on its combinatorial structure. Lemma 4.2 guarantees that any solution to the combinatorial problem yields a valid strategy to compute ϕ. Before stating it, we first formally rephrase the picture in Figure 2.

Definition 4.1. Let T_n be the portion of the unit triangular equilateral lattice contained between the x-axis, the line $y = \sqrt{3}x$ and the line $y = -\sqrt{3}(x - n + 1)$. We call T_n the *discrete equilateral triangle* (DET) of side n.

An *edge* is any segment of unit length directed downwards to the x-axis connecting two points of T_n. We say an edge is a *left edge* if it has positive slope, a *right edge* otherwise. This definition yields a directed acyclic graph (DAG) structure on T_n.

We equip the vertices of any directed graph G with the ordering $\to_G$ defined by: $x \to_G y$ if and only if there exists a path in G from x to y. We use $\to$ as a shorthand for $\to_{T_n}$. The *leaves* of G are the final vertices, and the *roots* of G are the initial vertices. The graph T_n has the n vertices on the x-axis as leaves and the top-most vertex as its unique root. We write $|G|$ for the number of leaves of G.

For any two vertices y, y' of T_n, there exists a most final vertex x with the property that $x \to y$ and $x \to y'$. We write $x = y \wedge y'$.

A *strategy* S is a sub-graph of T_n having a unique root. It is *full* if its leaves contain those of T_n; in this case, the root of S is the same as that of T_n. The *fork* of S is the final-most vertex x of S that is comparable (for the ordering $\to_S$) to all vertices of S.

Lemma 4.2. *Any full strategy yields a valid algorithm to compute the isogeny* $\phi = \phi_{n-1} \circ \cdots \circ \phi_0$ *by decorating the DET as shown in Figure* 2.

Proof. Travel the graph in depth-first left-first order. Upon reaching the bottom, apply Vélu's formulas before going right. □

At this point it should be clear that a strategy that passes twice through an interior vertex does unnecessary computations. Another waste of resources is a strategy having a leaf distinct from the leaves of T_n. Figure 3 shows examples of such ill-formed strategies.

We define a *well-formed* strategy to be one that has no such useless edge[3]. Figure 4 shows all the seven well-formed full strategies for $n = 4$, the leftmost and the rightmost ones corresponding respectively to the *isogeny-based* and the *multiplication-based* algorithms from [19].

It is clear that some strategies are computationally better than others. From the figures, we see that the multiplication-based and isogeny-based strategies have a number of edges quadratic in n, whereas the middle strategy in Figure 4 extends to a family of *balanced strategies* with asymptotically $\frac{1}{2\log 2} n \log n$ left and right edges.

[3] We do not know much about well-formed full strategies. One remarkable fact is that they are in one-to-one correspondence with certain instances of Gelfand–Tsetlin patterns [29].

Figure 3. Two ill-formed strategies.

Figure 4. The seven well-formed full strategies for $n = 4$.

We are thus interested in computing the "best" full strategy, according to some measure of computational effort. We first observe that any well-formed strategy has a binary tree topology, obtained by forgetting the internal nodes of out-degree less than two and by keeping the same connectivity structure. Conversely, to any binary tree A with n leaves one can canonically associate a strategy S on T_n as follows: if $n = 1$, then $S = T_1$; else, let S' be the strategy associated to the left branch of A, let S'' be the translated to the right by $|S'|$ of the strategy associated to the right branches of A, and let r', r'' be their roots. Then $r' \wedge r''$ is the root of T_n and we define $S = rr' \cup rr'' \cup S' \cup S''$, where rr' and rr'' are the (unique) paths from r to r' and r'' in T_n.

For example, in Figure 4 the three middle strategies share the same tree topology, and the middle one is the canonical one.

In our original problem, left edges are multiplications by ℓ, while right edges are ℓ-isogeny evaluations, and these steps have different costs. Thus it makes sense to assign different weight to left and right edges.

We fix a *measure* on T_n, i.e., a pair (p, q) of positive real numbers, where p represents the weight of a left edge (i.e., a point multiplication) and q represents the weight of a right edge (i.e., an isogeny). For any set of edges S of T_n, we write (S) for the sum of the weights of all edges of S.

If $x \to y$, then all paths going from x to y in T_n have the same measure; we write (xy) for this measure. For all vertices x, y, y', y'' such that $x \to y$, $y \to y''$ and $y \to y''$, we have the inequality $(xy) + (yy') + (yy'') \leq (xy') + (xy'')$.

We find optimal strategies in two steps. The first step (Lemma 4.3) shows that, for any measure, only canonical strategies are interesting. Then we determine the optimal full strategy for a given measure (Proposition 4.6).

Lemma 4.3. *Among all the strategies sharing the same tree topology, the canonical strategy is minimal with respect to any measure.*

Lemma 4.3 follows from Lemma 4.4.

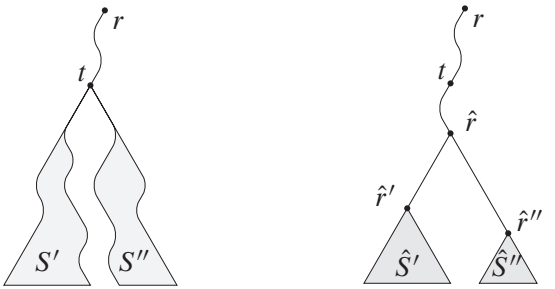

Figure 5. Proof of Lemma 4.4.

Lemma 4.4. *Let S be a strategy with root r. Let $\hat{S}$ be the canonical strategy associated to S and $\hat{r}$ be the root of $\hat{S}$. Then $r \to \hat{r}$ and $(r\hat{r}) + (\hat{S}) \leq (S)$.*

Proof. We prove this by induction on S. Let t be the fork of S. If t has no children in S, then it is the unique leaf of S and the lemma is obviously true. Therefore, we may assume that t has two children in S.

Let S', S'' be the right and left sub-strategies of S with root t, as shown in Figure 5. We apply the induction hypothesis on S' and S'' and use $\hat{r} = \hat{r}' \wedge \hat{r}''$ to derive the following (in)equalities:

$$\begin{aligned}
(S') + (S'') + (rt) &= (S) && \text{(definition of } (S)\text{)},\\
(t\hat{r}') + (\hat{S}') &\leq (S') && \text{(induction hypothesis on } \hat{S}'\text{)},\\
(t\hat{r}'') + (\hat{S}'') &\leq (S'') && \text{(induction hypothesis on } \hat{S}''\text{)},\\
(t\hat{r}) + (\hat{r}\hat{r}') + (\hat{r}\hat{r}'') &\leq (t\hat{r}') + (t\hat{r}'') && \text{(definition of } \hat{r}\text{)},\\
(r\hat{r}) &= (rt) + (t\hat{r}) && \text{(path } r \to t \to \hat{r}\text{)},\\
(\hat{S}) &= (\hat{r}\hat{r}') + (\hat{r}\hat{r}'') + (\hat{S}') + (\hat{S}'') && \text{(definition of } \hat{S}\text{)}.
\end{aligned}$$

By summing all of these and cancelling all extra terms, we obtain the desired result. □

Now that we have ruled out non-canonical strategies, we proceed to determine the best ones. From this point on, we are only interested in full strategies. Thus, we shall call *optimal* a strategy that is minimal, among all full strategies with n leaves, with respect to a measure (p, q).

In practice, we can compute optimal strategies using the fact that, for a fixed measure, optimality is a local property. For a full canonical strategy $S \neq T_1$, having i leaves to the left of its root, we define its *left branch* as $S \cap T_i$. We also

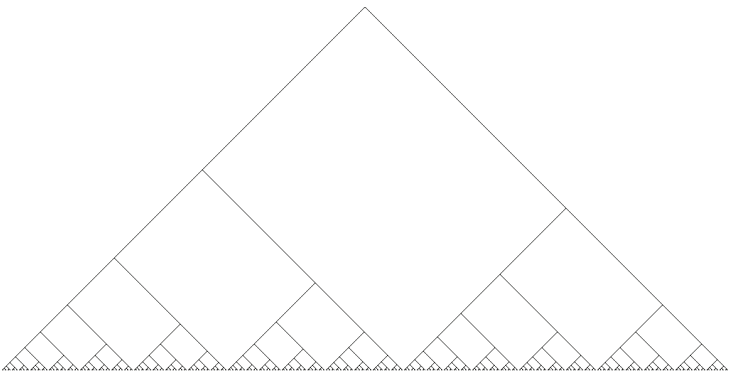

Figure 6. Optimal strategy for $n = 512$, $p = 4.6$, $q = 2.8$.

define its *right branch* as $S \cap T'$, where T' is $T_{|S|-i}$ translated by i to the right. Because S is full and canonical, both branches are full canonical strategies too.

Lemma 4.5. *Let S be an optimal strategy. Let S' and S'' be, respectively, its left and right branch. Then S' and S'' are optimal strategies.*

Proof. By Lemma 4.3, we know that S is a canonical strategy. Hence, S' and S'' are well defined. Now, suppose that S' were not optimal. By substituting an optimal strategy for S' inside S, we obtain a strategy with weight lower than (S). The same argument holds for S''. □

Define $C_{p,q}(n)$ to be the cost of the optimal strategies with n leaves. Lemma 4.5 says that

$$C_{p,q}(n) = \min_{i=1,\dots,n-1} \big(C_{p,q}(i) + C_{p,q}(n-i) \ + \ (n-i)p + iq\big).$$

This equality suggests a dynamic-programming algorithm that, given p and q, computes $C(n)$ in time $O(n^2)$. A straightforward Python implementation computes the optimal strategies for $n = 1024$ in less than one second, indicating that the dynamic-programming approach is very satisfying in practice. Figure 6 shows an optimal strategy for $n = 512$, $p = 4.6$, $q = 2.8$.

However, it is also possible to mathematically characterize the optimal strategies. Since the optimal strategies for (p, q) are symmetrical to those for (q, p), we may assume that $p < q$. For simplicity, we will also assume that p and q are integers: the generalization to the case of rational p and q is straightforward.

Let (u_m) be the sequence defined by $u_{p+1} = \cdots = u_{p+q} = 1$ and $u_m = u_{m-p} + u_{m-q}$, and $f(n)$ be the function defined by $f(1) = 0$, $f(n) = f(u_m) +$

$m(n - u_m)$ for all $n \in [u_m, u_{m+1}]$. We check that f satisfies the recurrence relation

$$f(u_m) = f(u_{m-p}) + f(u_{m-q}) + pu_{m-p} + qu_{m-q}. \tag{4.1}$$

Proposition 4.6. *A full strategy S is optimal if, and only if, it is canonical, both its left and right branches S' and S'' are optimal, and*

$$u_{m-q} \leq |S'| \leq u_{m-q+1}, \quad u_{m-p} \leq |S''| \leq u_{m-p+1},$$

where m is such that $u_m \leq |S| < u_{m+1}$. In this case, the weight of S is $f(|S|)$.

For example, in the case where $p = 1$ and $q = 2$, the sequence (u_m) is the sequence of Fibonacci numbers, starting at $u_2 = 1$, $u_3 = 1$, $u_4 = 2, \ldots$, and a strategy S with $u_m \leq |S| < u_{m+1}$ is optimal if, and only if, the left and right branches S' and S'' satisfy $u_{m-2} \leq |S'| \leq u_{m-1}$ and $u_{m-1} \leq |S''| \leq u_m$.

Proof. We prove this by induction on $n = |S|$. Let $n = n' + n''$ with $n', n'' > 0$ and let $S = S_{n',n''}$ be any canonical full strategy with optimal left and right branches S', S'', such that $|S'| = n'$ and $|S''| = n''$. Then any optimal strategy is one of the $S_{n',n''}$. Therefore, we find them by looking at the sign of $\delta_n = (S_{n'+1,n''-1}) - (S_{n',n''})$.

Let m', m'' be such that $u_{m'} \leq n' < u_{m'+1}$ and $u_{m''} \leq n'' < u_{m''+1}$. Using the induction hypothesis on S' and S'', we have $(S_{n',n''}) = f(n') + f(n'') + n'q + n''p$, and therefore

$$\delta_n = \begin{cases} (m' + q) - (m'' + p), & \text{if } n'' \geq u_{m''} + 1, \\ (m' + q) - (m'' + p) + 1, & \text{if } n'' = u_{m''}. \end{cases}$$

We find that $\delta_n \leq 0$ exactly when $n' < u_{m-q+1}$ and $n'' > u_{m-p}$, and $\delta_n \geq 0$ exactly when $n' \geq u_{m-q}$ and $n'' \leq u_{m-p+1}$. The optimality condition follows from this. It remains only to check that $(S) = f(|S|)$: this results from relation (4.1). □

In particular, there are in general several optimal strategies of a given size. Moreover, with z being the (unique) root in $[0, 1]$ of the equation $z^p + z^q - 1 = 0$, this gives the asymptotic equivalents

$$|S'| \sim z^q |S|,$$

$$C_{p,q}(n) = (S) \sim -\frac{1}{\log z}\, n \log n.$$

The relative cost of an optimal strategy is therefore asymptotically $-\frac{2 \log 2}{\log(z^{p+q})}$ times the cost of the balanced strategy. In the case where $(p, q) = (4.6, 2.8)$, as in Table 1 with $\ell = 2$, this gives an improvement of 2.1% over the balanced strategy.

4.3 Choice of the models

At each phase of the key generation it is important to use models for elliptic curves that offer the fastest formulas for doubling, addition, isogeny computation and evaluation, etc.

To measure efficiency, we count the number of elementary operations in $\mathbb{F}_{p^2}$: we write I, M, S for the costs of one inversion, multiplication and squaring respectively, and we make the assumption $S \leq M \leq I$. We neglect additions, subtractions and comparisons. We refer to the Explicit Formulas Database (EFD) [4] for operation counts of elliptic point addition, doubling, etc. in various models and coordinate systems. Contrary to the convention taken in the EFD, we count multiplications by constants (other than small integers) as ordinary multiplications.

Any curve used in our cryptosystem has group structure $(\mathbb{Z}/(p \mp 1)\mathbb{Z})^2$. Hence either the curve or its twist has a point of order 4. Consequently, it is isomorphic to a twisted Edwards curve and to a Montgomery curve [3].

Twisted Edwards curves [3] have equation

$$E_{a,d} : ax^2 + y^2 = 1 + dx^2y^2.$$

They have many interesting properties, but what interests us most is their very efficient addition and doubling formulas. Using projective coordinates, one point addition costs $12M + 1S$ and one point doubling costs $4M + 4S$. When one of the points is scaled to have Z-coordinate equal to 1, these costs drop to $11M + 1S$ and $3M + 4S$, respectively.

Montgomery curves [28] have equation

$$M_{B,A} : By^2 = x^3 + Ax^2 + x. \tag{4.2}$$

They have very efficient arithmetic on their Kummer line, i.e., by representing points by the coordinates $(X : Z)$ where $x = X/Z$. Using this representation, a point can be doubled using $3M + 2S$, or $2M + 2S$ when it is scaled to have Z-coordinate equal to 1.

The Kummer line identifies P with $-P$; thus it is not possible to add two distinct points. However, it is still possible to perform what is called *differential addition*, i.e., adding points P and Q for which the difference $P - Q$ is known. One differential addition can be computed using $4M + 2S$, or $3M + 2S$ when the difference $P - Q$ is scaled to have Z-coordinate equal to 1.

By using doublings and differential additions, it is possible to compute any scalar multiple of a point using a Montgomery ladder [28]. Also observe that since P and $-P$ generate the same subgroup, isogenies can be defined and evaluated correctly on the Kummer line. We shall give formulas for this operation later.

It is shown in [3] that the twisted Edwards curve $E_{a,d}$ is birationally equivalent to the Montgomery curve $M_{A,B}$ where

$$A = \frac{2(a+d)}{a-d}, \quad B = \frac{4}{a-d},$$

and where the transformation is given by the following maps (in affine coordinates)

$$\psi : E_{a,d} \to M_{A,B}, \quad (x,y) \mapsto \Big(\frac{1+y}{1-y}, \frac{1+y}{(1-y)x}\Big),$$
$$\psi^{-1} : M_{A,B} \to E_{a,d}, \quad (x,y) \mapsto \Big(\frac{x}{y}, \frac{x-1}{x+1}\Big).$$

Hence, after the models $E_{a,d}$ and $M_{A,B}$ are computed, points in projective coordinates can be moved from one model to the other at a cost of a few multiplications (and no inversions).

Now we detail the cost of each step of the key generation algorithm.

4.3.1 Computing $\langle [m]P + [n]Q \rangle$

We have already mentioned two algorithms to compute the value of $P + [m^{-1}n]Q$. One solution is to express the points P and Q in projective Edwards coordinates and perform a double-and-add followed by an addition. If we scale Q to have Z-coordinate equal to 1, computing $[m^{-1}n]Q$ costs $9.5M + 4.5S$ per bit on average.

Because addition by P is performed last, this approach cannot be used with points on the Montgomery curve in Kummer coordinates, but we can use the ladder given in Algorithm 1 instead. We first compute $P - Q$ using point addition in full projective coordinates (either by using the standard chord-and-tangent law, or by using the equivalence with twisted Edwards curves). Then we scale P, Q and $P - Q$ to have Z-coordinate equal to 1. We can now discard the Y-coordinate and work on the Kummer line. At each iteration we perform one doubling and two differential additions (with one of the scaled points P, Q and $P - Q$). The total cost of one iteration is thus $9M + 6S$.

In general the cost of one squaring is close to the cost of one multiplication. Thus the ladder algorithm is slightly slower than double-and-add. However, the advantages of the ladder are SPA resistance and an implementation simplified by not using Edwards coordinates at all for key generation.

4.3.2 Isogenies of Montgomery curves

The literature on efficient formulas for evaluating small degree isogenies is much less extensive than for point multiplication. We now give explicit formulas for

isogenies of Montgomery curves, and optimize the degree 2 and 3 cases. Our goal is to obtain the most efficient formulas for isogeny evaluation, and thus we seek to avoid inversions and square root computations as much as possible.

Let E be the Montgomery curve defined by (4.2). It has a point $P_2 = (0,0)$ of order two, and a point $P_4 = (1, \sqrt{(A+2)/B})$ of order four (sometimes defined over a quadratic extension) such that $[2]P_4 = P_2$. Montgomery curves have twists of the form $\tilde{y} = \sqrt{c}\,y$; these are isomorphisms when c is a square. The change of coordinates $\tilde{x} = x/B$, $\tilde{y} = y/B$ brings the curve E to the Weierstrass form

$$\tilde{y}^2 = \tilde{x}^3 + \frac{A}{B}\tilde{x}^2 + \frac{1}{B^2}\tilde{x},$$

and the point P_4 to $P_4' = (1/B, \ldots)$. Conversely, given a Weierstrass curve with equation $\tilde{y}^2 = \tilde{x}^3 + a\tilde{x}^2 + b\tilde{x}$, together with a point $P_4 = (1/\beta, \ldots)$ (with its ordinate possibly lying in a quadratic extension) such that $[2]P_4 = (0,0)$, the change of variables $\tilde{x} = x/\beta$, $\tilde{y} = y/\beta$ brings the curve to the Montgomery form $\beta y^2 = x^3 + a\beta x^2 + x$.

Given a point $P \neq \pm P_4$ of order 4, we will need to compute the isomorphism of Montgomery curves that brings $[2]P$ in $(0,0)$ and P in $(1, \ldots)$. Let X be the abscissa of P and X_0 the abscissa of $[2]P$; by a straightforward calculation, we find that this isomorphism is given by the map

$$\iota : E \to E', \quad (x,y) \mapsto \Big(\frac{x - X_0}{X - X_0}, \frac{y}{X - X_0}\Big), \tag{4.3}$$

and the new curve has equation

$$E' : \frac{B}{X - X_0}y^2 = x^3 + \frac{3X_0 + A}{X - X_0}x^2 + x.$$

By precomputing $1/(X - X_0)$, and sharing common subexpressions, the above map can be evaluated on a point in full projective coordinates using $3M$ operations, or $2M$ using Kummer coordinates.

Let G be a subgroup of the Montgomery curve E of odd cardinality ℓ and let h be the degree $(\ell - 1)/2$ polynomial vanishing on the abscissas of G. With a twist $y = \tilde{y}/\sqrt{B}$, we can put E in the form

$$E : \tilde{y}^2 = \tilde{x}^3 + A\tilde{x}^2 + \tilde{x},$$

and this does not change the abscissas of G nor the polynomial h. Now, composing the twist with Vélu's formulas gives an isogeny

$$\phi : E \to E/G, \quad (x,y) \mapsto \Big(\frac{g(x)}{h(x)^2}, y\sqrt{B}\Big(\frac{g(x)}{h(x)^2}\Big)'\Big).$$

We now need to express E/G in Montgomery form. Because ℓ is odd, the point $(0,0)$ of E is sent to a point of order two in E/G, and the change of variables $\tilde{x} = x - g(0)/h(0)^2$ brings this point to $(0,0)$.

Now, $\phi(P_4)$ is a point of order four lying above $(0,0)$ (possibly in a quadratic extension). Its abscissa is rational and is given by

$$\frac{1}{\beta} = \frac{g(1)}{h(1)^2} - \frac{g(0)}{h(0)^2},$$

so we further apply the change of variables $\tilde{x} = \bar{x}/\beta$, $\tilde{y} = \bar{x}/\beta$ to obtain a Montgomery curve. Finally, we have to twist back the model in order to obtain a curve isogenous over the base field: the twist $\bar{y} = y\sqrt{B}$ cancels with the first one and leaves us with square-root-free formulas.

Given h as input, the cost of evaluating the whole path is $O(\ell)$ operations in the base field, using the formula in [5, Proposition 4.1] to evaluate g/h.

Let now P be a 3-torsion point, let $G = \{0, P, -P\}$ and let X be the abscissa of P (and $-P$). If we specialize the formula to the case $\ell = 3$, we have $h = x - X$ and

$$\phi : E \to E/G,$$
$$(x,y) \mapsto \Big(\frac{x(x-\frac{1}{X})^2}{(x-X)^2}X^2,\ y\frac{(x-\frac{1}{X})\big((x-\frac{1}{X})(x-X)+2x(\frac{1}{X}-X)\big)}{(x-X)^3}X^2\Big),$$

and the curve E/G has equation

$$E/G : BX^2y^2 = x^3 + \Big(A + \frac{6}{X} - 6X\Big)X^2x^2 + x.$$

By precomputing X and X^2, and sharing common subexpressions, the above isogeny can be evaluated on a point in full projective coordinates using $11M + 2S$ operations, or $4M + 2S$ using Kummer coordinates.

When ℓ is even, things get more complicated. Recall that $P_2 = (0,0)$ and $P_4 = (1, \sqrt{(A+2)/B})$. The isogeny of degree 2 vanishing on P_2 and mapping P_4 to $(0,0)$ is readily seen as being

$$\begin{aligned} F &: By^2 = x^3 + (A+6)x^2 + 4(2+A)x, \\ \phi &: E \to F, \quad (x,y) \mapsto \Big(\frac{(x-1)^2}{x}, y\Big(1 - \frac{1}{x^2}\Big)\Big). \end{aligned} \tag{4.4}$$

It is not immediately evident how to put F in Montgomery form without computing square roots. Let P_8 be a point satisfying $[2]P_8 = P_4$. Then we have $\phi(P_8) = (2\sqrt{2+A}, \ldots)$, and F can be put in the form

$$\frac{B}{2\sqrt{2+A}}y^2 = x^3 + \frac{A+6}{2\sqrt{2+A}}x^2 + x,$$

with the point $(1, \ldots)$ being the image of P_8. Any other isogeny of degree 2 can be treated by applying (4.3) to move the generator of the kernel in $(0, 0)$.

By precomputing $1/\phi(P_8)$, and sharing common subexpressions, the above isogeny can be evaluated on a point in full projective coordinates using $5M + 3S$ operations, or $2M + S$ using Kummer coordinates. Unfortunately, this formula takes no square roots only if an 8-torsion point above P_2 is known.

Alternatively, ϕ and F being as before, we consider the isogeny $\psi : F \to F/\langle(0, 0)\rangle$ given by

$$\begin{aligned} &G : \frac{B}{2-A}y^2 = x^3 - 2\frac{A+6}{2-A}x^2 + x, \\ &\psi : F \to G, \\ &(x, y) \mapsto \Big(\frac{1}{2-A}\frac{(x+4)(x+(A+2))}{x}, \frac{y}{2-A}\Big(1 - \frac{4(2+A)}{x^2}\Big)\Big). \end{aligned} \tag{4.5}$$

Then $\phi_4 = \psi \circ \phi$ is an isogeny $\phi_4 \colon E \to E/\langle P_4\rangle$ of degree 4, and the point $(1, \sqrt{-4(A+2)/B})$ of G generates the kernel of the dual isogeny $\hat{\phi}_4$. Any other isogeny of degree 4 can be treated by applying (4.3) to move the kernel point to $(1, \ldots)$.

By precomputing $1/(2-A)$, and sharing common subexpressions, the isogeny ϕ_4 can be evaluated on a point in full projective coordinates using $10M + 4S$ operations, or $4M + S$ using Kummer coordinates.

Unfortunately, this formula cannot be applied twice to obtain a degree 16 cyclic isogeny: indeed, a double application yields the multiplication-by-4 isogeny. If a chain of degree 4 isogenies is wanted, as for the algorithm in Section 4.2.2, this formula must be combined with the isomorphism in (4.3).

Isogenies of composite smooth degree are computed by composing small degree isogenies as discussed in Section 4.2.2. We focus on the fine-tuning of this algorithm when the small isogenies have degrees 2, 3 and 4.

It is important to remark that, after a generator R of the kernel of the isogeny has been computed, the algorithm does not use its ordinate at all: indeed, all the previous formulas for small degree isogenies only use the abscissas of the kernel points. Hence, we can throw away the ordinate of R altogether and only use scalar multiplication and isogeny evaluation formulas for points in Kummer coordinates.

As usual, some more care must be taken when computing cyclic isogenies of degree 2^e. In principle, one could compose either degree 2 or degree 4 isogenies (see (4.4) and (4.5)); however we have already pointed out some caveats:

- Both equations require the kernel point to be moved to some specific coordinates. This can be achieved using the isomorphism in (4.3).

ℓ	2	3	4
Isogeny	$2M + S$	$4M + 2S$	$6M + S$
	2.8	5.6	6.8
Multiplication	$3M + 2S$	$7M + 4S$	$6M + 4S$
	4.6	10.2	9.2

Table 1. Comparative costs for multiplication and isogeny evaluation in projective Kummer coordinates, in number of multiplications and squarings, and assuming $S = 0.8M$.

- After the first change of variables, (4.4) can be repeatedly chained with itself; however, from a point of order 2^e only $e - 2$ degree 2 isogenies can be computed this way, because the formula requires the knowledge of a point of order 8.
- (4.5) cannot be directly chained with itself to compute a cyclic isogeny of degree 4^e. It must be composed, instead, with (4.3) at each step.

Having this in mind, two obvious strategies to compute degree 2^e isogenies are:

(i) Use degree 2 isogenies as much as possible; use one degree 4 isogenies for the last two steps.

(ii) Use one degree 2 isogeny if e is odd, then use only degree 4 isogenies composed with isomorphisms.

Table 2 suggests that the second strategy yields a very small improvement over the first. The experiments of Section 7 show that operations not accounted for in this analysis might give a greater advantage to the second strategy. However, for lack of time, we only implemented the first one.

Finally, it should be noted that both approaches, if implemented as described above, leak two bits of security. Indeed, the point $(1, \ldots)$ of the image curve is a point of order 4 in the kernel of the dual of the computed isogeny (the secret). It is easy to mask this leakage by taking a random change of coordinates. However, for practical purposes, we found this masking more costly than simply adding two bits of security and letting the information leak, although technically the leaked information alters the complexity assumptions needed for the security proofs. For ease of analysis, we assume in Section 5 that coordinate masking has been applied.

Table 1 summarizes the costs of the isogeny evaluation and scalar multiplication formulas in light of these remarks. Because squaring in $\mathbb{F}_{p^2}$ is faster than multiplication, we also report the cost obtained by taking $S = 0.8M$ (a factor that roughly

	Optimal strategy			Balanced strategy		
ℓ	2	3	4	2	3	4
Isogenies	2741	1610	1166	2323	1430	1033
Multiplications	1995	1151	921	2307	1288	1025
Total cost	16852	20756	16402	17117	21146	16454

Table 2. Comparative costs of the balanced and the optimal strategies for computing a degree 2^{514} ($\ell = 2, 4$) or 3^{323} ($\ell = 3$) isogeny, assuming $S = 0.8M$.

approximates the fact that squaring requires two multiplications in $\mathbb{F}_p$ instead of three).

Table 2 compares the total cost of isogeny evaluations and scalar multiplications in a balanced and an optimal strategy for $\ell = 2, 3, 4$, based on the costs given in Table 1, at the classical 256-bit security level (see Section 5 for our complexity assumptions). We make the following observations, backed up by the benchmarks in Section 7:

- There may be a small advantage (less than 2%) in using isogenies of degree 4 instead of 2.
- The gain in using an optimal strategy instead of a balanced one is consistent with the predictions of Section 4.2.2.
- The difference between 2^e and 3^e isogenies is more significant (about 20%), suggesting that degree 2^e isogenies may be preferable for constrained devices.

5 Complexity assumptions

As before, let p be a prime of the form $\ell_A^{e_A}\ell_B^{e_B} \cdot f \pm 1$, and fix a supersingular curve E_0 over $\mathbb{F}_{p^2}$ together with bases $\{P_A, Q_A\}$ and $\{P_B, Q_B\}$ of $E_0[\ell_A^{e_A}]$ and $E_0[\ell_B^{e_B}]$, respectively. In analogy with the case of isogenies over ordinary elliptic curves, we define the following computational problems, adapted for the supersingular case:

Problem 5.1 (Decisional Supersingular Isogeny (DSSI) problem). Let E_A be another supersingular curve defined over $\mathbb{F}_{p^2}$. Decide whether E_A is $\ell_A^{e_A}$-isogenous to E_0.

Problem 5.2 (Computational Supersingular Isogeny (CSSI) problem). Let the map $\phi_A \colon E_0 \to E_A$ be an isogeny whose kernel is $\langle [m_A]P_A + [n_A]Q_A \rangle$, where m_A and n_A are chosen at random from $\mathbb{Z}/\ell_A^{e_A}\mathbb{Z}$ and not both divisible by ℓ_A. Given E_A and the values $\phi_A(P_B)$, $\phi_A(Q_B)$, find a generator R_A of $\langle [m_A]P_A + [n_A]Q_A \rangle$.

We remark that given a generator $R_A = [m_A]P_A + [n_A]Q_A$, it is easy to solve for (m_A, n_A), since E_0 has smooth order and thus extended discrete logarithms are easy in E_0 (see [47]).

Problem 5.3 (Supersingular Computational Diffie–Hellman (SSCDH) problem). Let $\phi_A: E_0 \to E_A$ be an isogeny whose kernel is equal to $\langle [m_A]P_A + [n_A]Q_A \rangle$, and let $\phi_B: E_0 \to E_B$ be an isogeny whose kernel is $\langle [m_B]P_B + [n_B]Q_B \rangle$, where m_A, n_A (respectively m_B, n_B) are chosen at random from $\mathbb{Z}/\ell_A^{e_A}\mathbb{Z}$ (respectively $\mathbb{Z}/\ell_B^{e_B}\mathbb{Z}$) and not both divisible by ℓ_A (respectively ℓ_B). Given the curves E_A, E_B and the points $\phi_A(P_B), \phi_A(Q_B), \phi_B(P_A), \phi_B(Q_A)$, find the j-invariant of $E_0/\langle [m_A]P_A + [n_A]Q_A, [m_B]P_B + [n_B]Q_B \rangle$.

Problem 5.4 (Supersingular Decision Diffie–Hellman (SSDDH) problem). Given a tuple sampled with probability $1/2$ from one of the following two distributions, determine from which distribution the tuple is sampled:

- $(E_A, \phi_A(P_B), \phi_A(Q_B), E_B, \phi_B(P_A), \phi_B(Q_A), E_{AB})$, wherein the quantities $E_A, \phi_A(P_B), \phi_A(Q_B), E_B, \phi_B(P_A)$, and $\phi_B(Q_A)$ are as in the SSCDH problem and
$$E_{AB} \cong E_0/\langle [m_A]P_A + [n_A]Q_A, [m_B]P_B + [n_B]Q_B \rangle,$$
- $(E_A, \phi_A(P_B), \phi_A(Q_B), E_B, \phi_B(P_A), \phi_B(Q_A), E_C)$, wherein the quantities $E_A, \phi_A(P_B), \phi_A(Q_B), E_B, \phi_B(P_A)$, and $\phi_B(Q_A)$ are as in the SSCDH problem and
$$E_C \cong E_0/\langle [m'_A]P_A + [n'_A]Q_A, [m'_B]P_B + [n'_B]Q_B \rangle,$$
where m'_A, n'_A (respectively m'_B, n'_B) are chosen at random from $\mathbb{Z}/\ell_A^{e_A}\mathbb{Z}$ (respectively $\mathbb{Z}/\ell_B^{e_B}\mathbb{Z}$) and not both divisible by ℓ_A (respectively ℓ_B).

The ordinary case analogue of the following problem is trivially solvable in polynomial time. Its supposed difficulty in the supersingular case is at the heart of the security of our identification scheme.

Problem 5.5 (Decisional Supersingular Product (DSSP) problem). Given a degree $\ell_A^{e_A}$ isogeny $\phi : E_0 \to E_3$ and a tuple sampled with probability $1/2$ from one of the following two distributions, determine from which distribution the tuple is sampled:

- (E_1, E_2, ϕ'), where the product $E_1 \times E_2$ is chosen at random among those $\ell_B^{e_B}$-isogenous to $E_0 \times E_3$, and where $\phi' : E_1 \to E_2$ is an isogeny of degree $\ell_A^{e_A}$,

- (E_1, E_2, ϕ'), where E_1 is chosen at random among the curves having the same cardinality as E_0, and $\phi' : E_1 \to E_2$ is a random isogeny of degree $\ell_A^{e_A}$.

We conjecture that these problems are computationally infeasible, in the sense that for any polynomial-time solver algorithm, the advantage of the algorithm is a negligible function of the security parameter $\log p$. The resulting security assumptions are referred to as the DSSI assumption, CSSI assumption, etc.

5.1 Hardness of the underlying assumptions

Given a CSSI (respectively, SSCDH) solver, it is trivial to solve SSCDH (respectively, SSDDH). It is also trivial to solve SSDDH given a DSSI solver. There are no known reductions in the other direction, and given that the corresponding question of equivalence for discrete logarithms and Diffie–Hellman has not yet been completely resolved in all cases, it is reasonable to assume that the question of equivalence of CSSI, SSCDH, and SSDDH is at least hard to resolve. For the purposes of this discussion, we will presume that DSSI and CSSI are equivalent to SSDDH. Concerning DSSP, there is an evident reduction to DSSI. However, it seems reasonable to assume that DSSP is easier than the latter.

In the context of cryptography, the problem of computing an isogeny between isogenous supersingular curves was first considered by Galbraith [15] in 1999. The first published cryptographic primitive based on supersingular isogeny graphs is the hash function proposal of Charles et al. [9], which remains unbroken to date (the cryptanalysis of [31] applies only to the LPS graph-based hash function from [9], and not to the supersingular isogeny graph-based hash functions). The fastest known algorithm for finding isogenies between supersingular curves in general takes $O(\sqrt{p}\log^2 p)$ time [9, §5.3.1]; however our problem is less general because the degree of the isogeny is known in advance and is smooth. In addition, the distribution of isogenous curves obtained from taking kernels of the form $\langle [m_A]P_A + [n_A]Q_A\rangle$ is not quite uniform: a simple calculation against Proposition 2.1 indicates that a sequence of e_A isogenies of degree ℓ_A falls short of the length needed to ensure uniform mixing, regardless of the value of p. Since we are the first to propose using isogenies of this type, there is no existing literature addressing the security of the isogenies of the special form that we propose.

There are easy exponential attacks against DSSI and CSSI that improve upon exhaustive search. To find an isogeny of degree $\ell_A^{e_A}$ between E and E_A, an attacker builds two trees of all curves isogenous to E (respectively, E_A) via isogenies of degree $\ell_A^{e_A/2}$. Once the trees are built, the attacker tries to find a curve lying in both trees. Since the degree of the isogeny ϕ_A is $\sim \sqrt{p}$ (much shorter than the size of

the isogeny graph), it is unlikely that there will be more than one isogeny path – and thus more than one match – from E to E_A. Given two functions $f : A \to C$ and $g : B \to C$ with domain of equal size, finding a pair (a, b) such that $f(a) = g(b)$ is known as the *claw problem* in complexity theory. The claw problem can obviously be solved in $O(|A|+|B|)$ time and $O(|A|)$ space on a classical computer by building a hash table holding $f(a)$ for any $a \in A$ and looking for hits for $g(b)$ where $b \in B$. This gives a $O(\ell_A^{e_A/2}) = O(\sqrt[4]{p})$ classical attack against our cryptosystems. With a quantum computer, one can do better using the algorithm in [45], which has complexity $O(\sqrt[3]{|A||B|})$, thus giving an $O(\ell_A^{e_A/3}) = O(\sqrt[6]{p})$ quantum attack against our cryptosystems. These complexities are optimal for a black-box claw attack [49].

We consider the question of whether the auxiliary data points $\phi_A(P_B)$ and $\phi_A(Q_B)$ might assist an adversary in determining ϕ_A. Since (P_B, Q_B) forms a basis for $E_0[\ell_B^{e_B}]$, the values $\phi_A(P_B)$ and $\phi_A(Q_B)$ allow the adversary to compute ϕ_A on all of $E_0[\ell_B^{e_B}]$. This is because any element of $E_0[\ell_B^{e_B}]$ is a (known) linear combination of P_B and Q_B (known since extended discrete logarithms are easy [47]). However, there does not appear to be any way to use this capability to determine ϕ_A. Even on a quantum computer, where finding abelian hidden subgroups is easy, there is no hidden subgroup to find, since ϕ_A has degree $\ell_A^{e_A}$, and thus does not annihilate any point in $E_0[\ell_B^{e_B}]$ other than the identity. Of course, if one could evaluate ϕ_A on arbitrary points of $E_0[\ell_A^{e_A}]$, then a quantum computer could easily break the scheme, and indeed in this case the scheme is also easily broken classically by using a few calls to the oracle to compute a generator of the kernel of the dual isogeny $\hat{\phi}_A$. However, it does not seem possible to translate the values of ϕ_A on $E_0[\ell_B^{e_B}]$ into values on $E_0[\ell_A^{e_A}]$.

For both ordinary and supersingular curves, there is a natural bijection between isogenies (up to isomorphism) and (left) ideals in the endomorphism ring. In the ordinary case the endomorphism ring is commutative, and ideal classes form a finite abelian group. This property has been used by Childs et al. [10] to solve the ordinary analogue of CSSI in quantum subexponential time. It is natural to ask whether their algorithm can be adapted to the supersingular setting. Here the endomorphism ring is a maximal order in a non-commutative quaternion algebra, and the left ideal classes do not form a group at all (though they do form a groupoid). Since the algorithm of Childs et al. depends crucially on the properties of abelian groups, we believe that no reasonable variant of this strategy would apply.

The same correspondence between isogenies and ideals can be applied to DSSP. Indeed, deciding DSSP amounts to deciding whether the ideals S, S' associated to ϕ, ϕ' are conjugated, i.e., whether there exists a left ideal $R \in \mathrm{End}(E_0)$ such that $S = RS'R^{-1}$. Although it can be hoped that deciding conjugacy of ideal classes

in the quaternion algebra $\mathbb{Q}_{p,\infty}$ is feasible, we are still faced with the problem that the best known algorithms to compute the endomorphism rings of supersingular curves are exponential in $\log p$ (see [2,8,23]). Hence, we deem DSSP secure given the current knowledge.

The fact that it is possible to obtain a zero-knowledge identification scheme from CSSI comes as no surprise, since it is well known that a zero-knowledge protocol can be obtained from any problem in NP [18]. Nevertheless, the generic construction is not very efficient, and many efforts have been made to obtain efficient *ad-hoc* schemes from NP-complete problems [34, 37, 41, 42]. While the security of most of these schemes is based on two solid assumptions, namely that $\mathrm{P} \neq \mathrm{NP}$ and that *secure commitment schemes* exist, our identification scheme stands on a much weaker ground: the CSSI and DSSP problems. As performances go, it is reasonable to assume that our scheme will be some orders of magnitude slower than the best zero-knowledge protocols. We can thus conclude that our scheme is of a purely theoretical and pedagogical interest. Yet it is remarkable that an efficient identification scheme based on graphs of supersingular isogenies simply exists, while the analogous construction for ordinary curves is trivially broken and no other identification scheme is currently known to work in that case [44].

6 Security proofs

In this section we state formal security reductions relating the security of our protocols to the hardness of the appropriate underlying isogeny computation problem. The security proofs are routine, but tedious, and contain little original contribution on our part. For this reason, we only prove a representative selection of our theorems.

The statements of the theorems are as follows:

Theorem 6.1. *If the SSDDH assumption holds, then the key-agreement protocol of Section* 3.2 *is session-key secure in the authenticated-links adversarial model of Canetti and Krawczyk* [7].

Theorem 6.2. *If the SSDDH assumption holds and the hash function family $\mathcal{H}$ is entropy-smoothing, then the public-key cryptosystem of Section* 3.3 *is IND-CPA.*

Theorem 6.3. *Under the CSSI and DSSP assumptions, the identification scheme of Section* 3.1 *is zero-knowledge.*

The proofs of Theorems 6.1 and 6.2 are easily adapted from the corresponding proofs given by Stolbunov [43]. As an illustration of the proof techniques, we provide a proof of Theorem 6.1.

6.1 Proof of Theorem 6.1

We recall the definition of session-key security in the authenticated-links adversarial model of Canetti and Krawczyk [7]. We consider a finite set of *parties* $P_1, P_2, \ldots, P_n$ modeled by probabilistic Turing machines. The adversary $\mathcal{I}$, also modeled by a probabilistic Turing machine, controls all communication, with the exception that the adversary cannot inject or modify messages (except for messages from corrupted parties or sessions), and any message may be delivered at most once. Parties give outgoing messages to the adversary, who has control over their delivery via the Send query. Parties are activated by Send queries, so the adversary has control over the creation of protocol sessions, which take place within each party. Two sessions s and s' are *matching* if the outgoing messages of one are the incoming messages of the other, and vice versa.

We allow the adversary black-box access to the queries SessionStateReveal, SessionKeyReveal, and Corrupt. The SessionStateReveal($\mathbb{s}$) query allows the adversary to obtain the contents of the session state, including any secret information. The query is noted and $\mathbb{s}$ produces no further output. The SessionKeyReveal($\mathbb{s}$) query enables the adversary to obtain the session key for the specified session $\mathbb{s}$, so long as $\mathbb{s}$ holds a session key. The Corrupt(P_i) query allows the adversary to take over the party P_i, i.e., the adversary has access to all information in P_i's memory, including long-lived keys and any session-specific information still stored. A corrupted party produces no further output. We say a session $\mathbb{s}$ with owner P_i is *locally exposed* if the adversary has issued SessionKeyReveal($\mathbb{s}$), SessionStateReveal($\mathbb{s}$), or Corrupt(P_i) before $\mathbb{s}$ is expired. We say $\mathbb{s}$ is *exposed* if $\mathbb{s}$ or its matching session have been locally exposed, and otherwise we say $\mathbb{s}$ is *fresh*.

We allow the adversary $\mathcal{I}$ a single Test($\mathbb{s}$) query, which can be issued at any stage to a completed, fresh, unexpired session $\mathbb{s}$. A bit b is then picked at random. If $b = 0$, the test oracle reveals the session key, and if $b = 1$, it generates a random value in the key space. $\mathcal{I}$ can then continue to issue queries as desired, with the exception that it cannot expose the test session. At any point, the adversary can try to guess b. Let $\text{GoodGuess}^{\mathcal{I}}(k)$ be the event that $\mathcal{I}$ correctly guesses b, and define

$$\text{Advantage}^{\mathcal{I}}(k) = \max\left\{0, \left|\Pr[\text{GoodGuess}^{\mathcal{I}}(k)] - \frac{1}{2}\right|\right\},$$

where k is a security parameter.

The definition of security is as follows:

Definition 6.4. A key exchange protocol Π in security parameter k is said to be *session-key secure* in the authenticated-links adversarial model of Canetti and Krawczyk if the following holds for any polynomial-time adversary $\mathcal{I}$:

(i) If two uncorrupted parties have completed matching sessions, these sessions produce the same key as output.

(ii) $\text{Advantage}^{\mathcal{I}}(k)$ is negligible.

Algorithm 2 SSDDH distinguisher

Input: $E_A, E_B, \phi_A(P_B), \phi_A(Q_B), \phi_B(P_A), \phi_B(Q_A), E$

1: $r \xleftarrow{R} \{1, \ldots, k\}$, where k is an upper bound on the number of sessions activated by $\mathcal{I}$ in any interaction.
2: Invoke $\mathcal{I}$ and simulate the protocol to $\mathcal{I}$, except for the r-th activated protocol session.
3: For the r-th session, let Alice send A, i, E_A, $\phi_A(P_B)$, $\phi_A(Q_B)$ to Bob, and let Bob send B, i, E_B, $\phi_B(P_A)$, $\phi_B(Q_A)$ to Alice, where i is the session identifier.
4: **if** the r-th session is chosen by $\mathcal{I}$ as the test session **then**
5: Provide $\mathcal{I}$ as the answer to the test query.
6: $d \leftarrow \mathcal{I}$'s output.
7: **else**
8: $d \xleftarrow{R} \{0, 1\}$.
9: **end if**

Output: d

Proof of Theorem 6.1. We adapt the proof given by Canetti and Krawczyk [7, §5.1] for two-party Diffie–Hellman over $\mathbb{Z}_q^*$. A similar strategy was used by Stolbunov [43] in the case of ordinary elliptic curves.

We have shown in Section 3 that two uncorrupted parties in matching sessions output the same session key, and thus the first part of Definition 6.4 is satisfied. To show that the second part of the definition is satisfied, assume that there is a polynomial-time adversary $\mathcal{I}$ with a non-negligible advantage ε. We claim that Algorithm 2 forms a polynomial-time distinguisher for SSDDH having non-negligible advantage.

To prove the claim, we must show that Algorithm 2 has non-negligible advantage (it is clear that it runs in polynomial time). We consider separately the cases where the r-th session is (respectively, is not) chosen by $\mathcal{I}$ as the test session. If the r-th session is not the test session, then Algorithm 2 outputs a random bit, and thus its advantage in solving the SSDDH is 0. If the r-th session is the test session, then $\mathcal{I}$ will succeed with advantage ε, since the simulated protocol provided to $\mathcal{I}$ is indistinguishable from the real protocol. Since the latter case occurs with probability $1/k$, the overall advantage of the SSDDH distinguisher is ε/k, which is non-negligible. □

6.2 Proof of Theorem 6.3

Using classical techniques from [14, 18], Theorem 6.3 is proved in three steps, known by the names of *completeness*, *soundness* and *zero-knowledge*.

Proof of Theorem 6.3 *(sketch).* Completeness is obvious: using the algorithms of Section 4, Peggy can always compute the diagram (3.2) in polynomial time and make Vic accept.

To prove soundness, we let Charles be any polynomially bounded adversary capable of convincing Vic with a non-negligible probability. We use Charles as a black-box for which we can control the random coin tosses. By restarting it a polynomial number of times with the same random input, and by asking each time a different set of questions, we learn with overwhelming probability a diagram

$$\begin{array}{ccc} E & & E/\langle S\rangle \\ {\scriptstyle\psi}\downarrow & & \downarrow{\scriptstyle\psi'} \\ E/\langle R\rangle & \xrightarrow{\phi'} & E/\langle S,R\rangle \end{array}$$

It is then straightforward to compute the secret. Let R be a generator of the kernel of ϕ'. Using the theorem of the dual isogeny it is easy to compute $\hat{\psi}$, then $\langle\hat{\psi}(R)\rangle$ is the kernel of an isogeny $E \to E/\langle S\rangle$ of degree $\ell_A^{e_a}$. This contradicts the CSSI assumption.

To prove zero-knowledge, we use a *cheating verifier* (CV) as a black-box to construct a *simulator* (S). At each iteration, S makes a (uniformly) random guess at what the next question by V will be.

If S guesses $b = 0$, it chooses a random primitive $\ell_B^{e_B}$-torsion point $R \in E$ and computes $\phi(R)$ (recall that the action of ϕ on $E[\ell_B^{e_B}]$ is part of the public data). Then it constructs the isogenies $\psi : E \to E/\langle R\rangle$ and $\psi' : E/\langle S\rangle \to E/\langle S,R\rangle$,

$$\begin{array}{ccc} E & \overset{\phi}{\dashrightarrow} & E/\langle S\rangle \\ {\scriptstyle\psi}\downarrow & & \downarrow{\scriptstyle\psi'} \\ E/\langle R\rangle & & E/\langle S,R\rangle \end{array}$$

and sends $E_1 = E/\langle R\rangle$ and $E_2 = E/\langle S,R\rangle$ to CV.

If S guesses $b = 1$, it chooses a random supersingular curve E' having the same cardinality as E, and a random primitive $\ell_A^{e_A}$-torsion point $R \in E'$. Then, it constructs the isogeny $\phi' : E' \to E'/\langle R\rangle$,

$$E \overset{\phi}{\dashrightarrow} E/\langle S\rangle$$
$$E' \overset{\phi'}{\longrightarrow} E'/\langle R\rangle$$

and sends $E_1 = E'$ and $E_2 = E'/\langle R\rangle$ to CV.

If CV does not ask the expected question, S simply discards the attempt and restarts. If CV asks the expected question, S writes (E_1, E_2, b, R) on its output. S stops whenever CV rejects, or after m successful interactions with CV.

To prove zero-knowledge, we must show that S runs in polynomial time and that its output is polynomially indistinguishable from the transcript of a conversation between CV and Peggy.

To show that S runs in polynomial time, it is enough to show that, at any iteration, for any guess b made by S, the probability that CV asks question $1 - b$ is exponentially close to $1/2$. Suppose this were not the case, then CV can be used as an oracle for DSSP.

To prove indistinguishability, using the *hybrid* technique of [18, Claim 4.2], it is enough to prove that no polynomial-time distinguisher exists for a single round of the identification scheme. It is obvious that no such distinguisher can exist for questions of type $b = 0$, because the outputs of S and Peggy are identical in this case. Suppose, now, that there exists a distinguisher D which, on input $\phi' : E_1 \to E_2$, can tell with non-negligible probability whether it comes from S or from a conversation between CV and Peggy, then D can be used as an oracle for DSSP. □

7 Implementation results and example

To evaluate the performance of our schemes, we implemented the key exchange protocol in the computer algebra system Sage [40] using a mixed C/Cython/Sage architecture. This allows us to access the large palette of number theoretic algorithms distributed with Sage, while still permitting very efficient code in C/Cython for the critical parts such as the algorithms of Section 4.2. The source code can be downloaded from De Feo's web page.

Arithmetic in $\mathbb{F}_{p^2}$ is written in C. We use the library GMP for arithmetic modulo p. The field $\mathbb{F}_{p^2}$ is implemented as $\mathbb{F}_{p^2}[X]/(X^2+1)$ (this requires $p = 3 \bmod 4$); using this representation, one multiplication in $\mathbb{F}_{p^2}$ requires three multiplications in $\mathbb{F}_p$, one squaring requires two multiplications in $\mathbb{F}_p$, and one inversion requires one inversion, two squarings, and two multiplications in $\mathbb{F}_p$. Our experiments show that, for the sizes we are interested in, $I = 10M$ and $S = 0.8M$.

	tuned 2/1			balanced	
	512 bits	768 bits	1024 bits	768 bits	1024 bits
Alice round 1	28.1 ms	65.7 ms	122 ms	66.8 ms	123 ms
Alice round 2	23.3 ms	54.3 ms	101 ms	55.5 ms	102 ms
Bob round 1	28.0 ms	65.6 ms	125 ms	67.1 ms	128 ms
Bob round 2	22.7 ms	53.7 ms	102 ms	55.1 ms	105 ms

Table 3. Benchmarks for various key sizes. Alice uses $\ell = 2$, Bob uses $\ell = 3$.

The computation of the optimal strategies as described in Section 4.2.2 is done in pure Python, using a dynamic programming algorithm.

We implemented the key exchange algorithm in C for $\ell = 2, 3$, and in Cython for any ℓ. Our experiments show that, in the C implementation, the ratio between scalar multiplications and isogeny evaluations is about 2, which is consistent with the predictions made in Table 1. We used this ratio to tune the optimal strategies for $\ell = 2, 3$: starting from 768 bits, a gain over the balanced strategy, comprised between 1% and 3%, starts getting noticeable. The performances of 3-isogenies are comparable to those of 2-isogenies, thus contradicting the prediction made by Table 2; this is explained by the operations not accounted for in Table 2, such as the computation of the isogenies and of the isogenous curves (3-isogenies gain a factor of about $\log_3 2$ on these operations). This suggests that well optimized 4-isogeny formulas may eventually outperform 2-isogenies.

Finally, the parameter generation is implemented in plain Sage. Because Sage is a collection of many open source mathematical systems, various of its subsystems are involved in this last part. Of these, Pari [30] plays an important role because it is used to compute Hilbert class polynomials and to factor polynomials over finite fields.

All tests ran on a 2.4 GHz Opteron running in 64-bit mode. The results are summarized in Table 3. At the quantum 128-bit security level (768-bit p), our numbers improve upon Stolbunov's reported performance figures [44, Table 1] by over three orders of magnitude (0.066 seconds vs. 229 seconds). This is the highest security level appearing in [44, Table 1], so comparisons at higher levels are difficult. Nevertheless, it seems safe to assume that the improvement is even greater at the 256-bit security level. Our results demonstrate that the proposed scheme is practical.

7.1 Example

As a convenience, we provide an example calculation of a key-exchange transaction. Let $\ell_A = 2$, $\ell_B = 3$, $e_A = 63$, $e_B = 41$, and $f = 11$. We use the starting curve $E_0 : y^2 = x^3 + x$. For the torsion bases, we use

$$\begin{aligned} P_A = (&237409306833625077410793642140789388589 7i \\ &+ 252464670185239634930842532821820356969 3, \\ &194486926041457420622915324351010478172 5i \\ &+ 130909941321176707805523276846048341720 1), \\ P_B = (&155671603365753087672852505928443176120 6i \\ &+ 174740732959516524133513164792986606521 5, \\ &345695620285202883552941999547591538848 3i \\ &+ 197591287424745857265472071715575500556 6) \end{aligned}$$

and $Q_A = \psi(P_A), Q_B = \psi(P_B)$, where $i = \sqrt{-1}$ in $\mathbb{F}_{p^2}$ and $\psi(x, y) = (-x, iy)$ is a distortion map [21]. The secret values are

$$\begin{aligned} m_A &= 2575042839726612324, \quad n_A = 8801426132580632841, \\ m_B &= 4558164392438856871, \quad n_B = 20473135767366569910. \end{aligned}$$

The isogeny $\phi_A: E_0 \to E_A$ is specified by its kernel, and thus the curve E_A is only well defined up to isomorphism; its exact value may vary depending on the implementation. In our case, the curve is $E_A : y^2 = x^3 + ax + b$ where

$$\begin{aligned} a = &428128245356224894562061821180718114127i \\ &+ 214770800990771179013498662460467452576 9, \\ b = &323035926720219746030433183517042405309 3i \\ &+ 157726433648237019704536235910489488486 2, \end{aligned}$$

and the values of $\phi_A(P_B)$ and $\phi_A(Q_B)$ are

$$\begin{aligned} \phi_A(P_B) = (&121624303795507829290097485944106602697 6i \\ &+ 166629113680473868483263718767433090557 2, \\ &313292160945399836185337294189350010792 3i \\ &+ 282316493857354948561980003461685523666), \end{aligned}$$

$$\begin{aligned}\phi_A(Q_B) = (&203972869442093051915573296501829191066 0i\\ &+ 24220926143229881124929316155281557273 88,\\ &16881158126943551455498892385104570342 72i\\ &+ 13791859846082406389129488903497384675 36).\end{aligned}$$

Similarly, in our implementation $E_B : y^2 = x^3 + ax + b$ is the curve with

$$\begin{aligned}a = &25747223980940229685783138618846089431 22i\\ &+ 46450755714955906218417413257164742772 2,\\ b = &28634789075130887921449983112297728861 97i\\ &+ 17670780367141094057967770650898683867 53,\end{aligned}$$

and the values of $\phi_B(P_A)$ and $\phi_B(Q_A)$ are

$$\begin{aligned}\phi_B(P_A) = (&25190860033479732147704991541625400981 81i\\ &+ 14597029740096091987239811254575484408 72,\\ &20720570679332925993269287662551550813 80i\\ &+ 89162210063825884940161855214523231139 5),\\ \phi_B(Q_A) = (&53793994522803393243921432982798543666i\\ &+ 36987416097881386855884895683431905048 44,\\ &28538680739718083986496636521612153237 50i\\ &+ 18697304800536241413723732827958586911 39).\end{aligned}$$

The common j-invariant of $E_{AB} \cong E_{BA}$, computed by both Alice and Bob, is equal to

$$\begin{aligned}j(E_{AB}) = &14371454943626551191684828087021114137 44i\\ &+ 83349809677838645295172228531059205635 1.\end{aligned}$$

8 Conclusion

We propose a new family of conjecturally quantum-resistant public-key cryptographic protocols using isogenies between supersingular elliptic curves of smooth order. In order to compensate for the non-commutative endomorphism rings that arise in this setting, we introduce the idea of providing the images of torsion bases as part of the protocol. Against the fastest known attacks, the resulting key exchange scheme improves upon all previous isogeny-based schemes by orders of

magnitude in performance at conventional security levels, making it the first practical isogeny-based public-key cryptosystem. Unlike prior such schemes, our proposal admits no known subexponential-time attacks even in the quantum setting.

Acknowledgments. We would like to thank Gaëtan Bisson, Andrew M. Childs, Alfred Menezes, Vladimir Soukharev, and the anonymous reviewers for helpful comments and suggestions.

Bibliography

[1] A. Antipa, D. Brown, R. Gallant, R. Lambert, R. Struik and S. Vanstone, Accelerated verification of ECDSA signatures, in: *Selected Areas in Cryptography 2005*, Lecture Notes in Comput. Sci. 3897, Springer, Berlin (2006), 307–318.

[2] J. V. Belding, *Number theoretic algorithms for elliptic curves*, Ph.D. thesis, University of Maryland, 2008.

[3] D. Bernstein, P. Birkner, M. Joye, T. Lange and C. Peters, Twisted Edwards curves, in: *Progress in Cryptology – AFRICACRYPT 2008*, Lecture Notes in Comput. Sci. 5023, Springer (2008), 389–405.

[4] D. J. Bernstein and T. Lange, *Explicit-formulas database*, 2007, `www.hyperelliptic.org/EFD/index.html`.

[5] A. Bostan, F. Morain, B. Salvy and É. Schost, Fast algorithms for computing isogenies between elliptic curves, *Math. Comp.* **77** (2008), 1755–1778.

[6] R. Bröker, Constructing supersingular elliptic curves, *J. Comb. Number Theory* **1** (2009), 269–273.

[7] R. Canetti and H. Krawczyk, Analysis of key-exchange protocols and their use for building secure channels, in: *EUROCRYPT*, Lecture Notes in Comput. Sci. 2045, Springer (2001), 453–474.

[8] J. M. Cerviño, On the correspondence between supersingular elliptic curves and maximal quaternionic orders, preprint (2004), `http://arxiv.org/abs/math/0404538`.

[9] D. X. Charles, K. E. Lauter and E. Z. Goren, Cryptographic hash functions from expander graphs, *J. Cryptology* **22** (2009), 93–113.

[10] A. Childs, D. Jao and V. Soukharev, Constructing elliptic curve isogenies in quantum subexponential time, preprint (2010), `http://arxiv.org/abs/1012.4019/`.

[11] J.-M. Couveignes, Hard homogeneous spaces, preprint (2006), `http://eprint.iacr.org/2006/291/`.

[12] G. Davidoff, P. Sarnak and A. Valette, *Elementary Number Theory, Group Theory, and Ramanujan Graphs*, London Math. Soc. Stud. Texts 55, Cambridge University Press, Cambridge, 2003.

[13] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Trans. Inform. Theory* **31** (1985), 469–472.

[14] U. Feige, A. Fiat and A. Shamir, Zero-knowledge proofs of identity, *J. Cryptology* **1** (1988), 77–94.

[15] S. D. Galbraith, Constructing isogenies between elliptic curves over finite fields, *LMS J. Comput. Math.* **2** (1999), 118–138.

[16] S. D. Galbraith, F. Hess and N. P. Smart, Extending the GHSWeil descent attack, in: *Advances in Cryptology – EUROCRYPT 2002*, Lecture Notes in Comput. Sci. 2332, Springer, Berlin (2002), 29–44.

[17] S. D. Galbraith and A. Stolbunov, Improved algorithm for the isogeny problem for ordinary elliptic curves, preprint (2011), `http://arxiv.org/abs/1105.6331/`.

[18] O. Goldreich, S. Micali and A. Wigderson, Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems, *J. Assoc. Comput. Mach.* **38** (1991), 690–728.

[19] D. Jao and L. De Feo, Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies, in: *PQCrypto*, Lecture Notes in Comput. Sci. 7071, Springer (2011), 19–34.

[20] D. Jao, S. D. Miller and R. Venkatesan, Expander graphs based on GRH with an application to elliptic curve cryptography, *J. Number Theory* **129** (2009), 1491–1504.

[21] A. Joux, The Weil and Tate pairings as building blocks for public key cryptosystems, in: *Algorithmic Number Theory* (Sydney 2002), Lecture Notes in Comput. Sci. 2369, Springer, Berlin (2002), 20–32.

[22] P. Kocher, J. Jaffe and B. Jun, Differential power analysis, in: *Advances in Cryptology – CRYPTO' 99*, Lecture Notes in Comput. Sci. 1666, Springer, Berlin (1999), 388–397.

[23] D. Kohel, *Endomorphism rings of elliptic curves over finite fields*, Ph.D. thesis, University of California, Berkeley, 1996.

[24] J. C. Lagarias and A. M. Odlyzko, Effective versions of the Chebotarev density theorem, in: *Algebraic Number Fields: L-Functions and Galois Properties* (Durham 1975), Academic Press, London (1977), 409–464.

[25] A. Lubotzky, *Discrete Groups, Expanding Graphs and Invariant Measures*, Progr. Math. 125, Birkhäuser, Basel, 1994.

[26] A. Lubotzky, R. Phillips and P. Sarnak, Ramanujan graphs, *Combinatorica* **8** (1988), 261–277.

[27] J.-F. Mestre, La méthode des graphes. Exemples et applications, in: *Proceedings of the International Conference on Class Numbers and Fundamental Units of Algebraic Number Fields* (Katata 1986), Nagoya University (1986), 217–242.

[28] P. L. Montgomery, Speeding the Pollard and elliptic curve methods of factorization, *Math. Comp.* **48** (1987), 243–264.

[29] OEIS Foundation Inc., *The on-line encyclopedia of integer sequences*, 2012, `http://oeis.org/A130715`.

[30] The PARI Group, Bordeaux, *PARI/GP, version 2.4.3*, 2008, available from `http://pari.math.u-bordeaux.fr/`.

[31] C. Petit, K. Lauter and J.-J. Quisquater, Full cryptanalysis of LPS and Morgenstern hash functions, in: *Proceedings of the 6th International Conference on Security and Cryptography for Networks* (SCN '08), Springer, Berlin (2008), 263–277.

[32] A. K. Pizer, Ramanujan graphs and Hecke operators, *Bull. Amer. Math. Soc. (N.S.)* **23** (1990), 127–137.

[33] A. K. Pizer, Ramanujan graphs, in: *Computational Perspectives on Number Theory* (Chicago 1995), AMS/IP Stud. Adv. Math. 7, American Mathematical Society, Providence (1998), 159–178.

[34] D. Pointcheval, A new identification scheme based on the perceptrons problem, in: *Advances in Cryptology – EUROCRYPT '95*, Lecture Notes in Comput. Sci. 921, Springer, Berlin (1995), 319–328.

[35] A. Rostovtsev and A. Stolbunov, Public-key cryptosystem based on isogenies, preprint (2006), `http://eprint.iacr.org/2006/145/`.

[36] P. Sarnak, *Some Applications of Modular Forms*, Cambridge Tracts in Math. 99, Cambridge University Press, Cambridge, 1990.

[37] A. Shamir, An efficient identification scheme based on permuted kernels (extended abstract), in: *Proceedings on Advances in Cryptology – CRYPTO '89*, Springer, New York (1989), 606–609.

[38] J. H. Silverman, *The Arithmetic of Elliptic Curves*, Grad. Texts in Math. 106, Springer, New York, 1992. Corrected reprint of the 1986 original.

[39] J. A. Solinas, *Low-weight binary representations for pairs of integers*, National Security Agency, USA, Report, 2001.

[40] W. A. Stein et al., *Sage Mathematics Software (version 4.6.2)*, The Sage Development Team, 2011, `www.sagemath.org`.

[41] J. Stern, A new identification scheme based on syndrome decoding, in: *Advances in Cryptology – CRYPTO' 93*, Lecture Notes in Comput. Sci. 773, Springer, Berlin (1994), 13–21.

[42] J. Stern, Designing identification schemes with keys of short size, in: *Advances in Cryptology – CRYPTO '94*, Lecture Notes in Comput. Sci. 839, Springer, Berlin (1994), 164–173.

[43] A. Stolbunov, Reductionist security arguments for public-key cryptographic schemes based on group action, in: *Norsk informasjonssikkerhetskonferanse (NISK)* (2009), 97–109.

[44] A. Stolbunov, Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves, *Adv. Math. Commun.* **4** (2010), 215–235.

[45] S. Tani, Claw finding algorithms using quantum walk, preprint (2008), `http://arxiv.org/abs/0708.2584`.

[46] J. Tate, Endomorphisms of abelian varieties over finite fields, *Invent. Math.* **2** (1966), 134–144.

[47] E. Teske, The Pohlig–Hellman method generalized for group structure computation, *J. Symbolic Comput.* **27** (1999), 521–534.

[48] J. Vélu, Isogénies entre courbes elliptiques, *C.R. Acad. Sci. Paris Sér. A-B* **273** (1971), A238–A241.

[49] S. Zhang, Promised and distributed quantum search computing and combinatorics, in: *Proceedings of the Eleventh Annual International Conference on Computing and Combinatorics*, Lecture Notes in Comput. Sci. 3595, Springer, Berlin (2005), 430–439.

Received June 29, 2012; revised May 14, 2014; accepted May 16, 2014.

Author information

Luca De Feo, Laboratoire PRiSM, Université de Versailles, 78035 Versailles, France.
E-mail: `luca.de-feo@prism.uvsq.fr`

David Jao, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada.
E-mail: `djao@math.uwaterloo.ca`

Jérôme Plût, Laboratoire PRiSM, Université de Versailles, 78035 Versailles, France.
E-mail: `jerome.plut@prism.uvsq.fr, jerome.plut@ssi.gouv.fr`