Key Compression for Isogeny-Based Cryptosystems

Reza Azarderakhsh[1], David Jao[2], Kassem Kalach[2,3], Brian Koziel[1], and Christopher Leonardi[2]

[1] Computer Engineering Department,
Rochester Institute of Technology, Rochester, USA
[2] Department of Combinatorics and Optimization
[3] Institute for Quantum Computing,
University of Waterloo, Ontario, Canada

Abstract. We present a method for key compression in quantum-resistant isogeny-based cryptosystems, which allows a reduction in and transmission costs of per-party public information by a factor of two, with no effect on security. We achieve this reduction by associating a canonical choice of elliptic curve to each j-invariant, and representing elements on the curve as linear combinations with respect to a canonical choice of basis. This method of compressing public information can be applied to numerous isogeny-based protocols, such as key exchange, zero-knowledge identification, and public-key encryption. We performed personal computer and ARM implementations of the key exchange with compression and decompression in C and provided timing results, showing the computational cost of key compression and decompression at various security levels. Our results show that isogeny-based cryptosystems achieve by far the smallest possible key sizes among all existing families of post-quantum cryptosystems at practical security levels; e.g. 3073-bit public keys at the quantum 128-bit security level, comparable to (non-quantum) RSA key sizes.

Keywords: elliptic curves, isogenies, key compression, post-quantum cryptography

1 Introduction

Traditional elliptic curve cryptography is based on the intractability of the elliptic curve discrete logarithm problem. These systems are not safe to use in a post-quantum world, since on a quantum computer, Shor's algorithm [16] can compute solutions to the discrete logarithm problem in any group (including elliptic curve groups) in polynomial time. In 2009 Stolbunov [19] created an encryption scheme relying on the computational difficulty of finding isogenies between ordinary elliptic curves. Soon after it was shown that one could determine the private keys of this system in subexponential time with a quantum attack [4]. Recent work of De Feo, Jao, and Plût [6] proposes to use isogenies (i.e. maps) between *supersingular* elliptic curves as the basis for quantum-safe elliptic curve cryptosystems. Unlike with discrete logarithms, there is no known polynomial-time algorithm to compute isogenies between elliptic curves in the general case, even on a quantum computer [2,4]. Implementation results [8] have shown that isogeny-based cryptosystems exhibit practical performance characteristics at standard security levels. Isogeny-based cryptography thus represents a promising and attractive candidate for constructing post-quantum cryptosystems.

We propose new methods for reducing the cost of transmitting and storing public keys and per-party public information in the isogeny-based cryptosystems of [6].

Table 1. Elliptic Curve Notations

Sym	Meaning
$\mathbb{Z}$	The set of integers
$\mathbb{F}_{p^n}$	A finite field of size p^n
$\overline{\mathbb{F}}_{p^n}$	The algebraic closure of the field $\mathbb{F}_{p^n}$
$\lvert\cdot\rvert$	The cardinality of the enclosed set
$\langle\cdot\rangle$	The orbit of the enclosed group element
$\oplus$	The bitwise exclusive-OR (XOR) operation

Our approach takes advantage of algebraic properties of elliptic curves to reduce the amount of bits the two communicating parties must publish, while only incurring the computational costs of compression and decompression once per key. Our algorithm compresses keys to half their original size, with no effect on security (Section 5.1). We explain the mathematical techniques that we use for key compression (Section 3), and give a detailed outline of our modified key-exchange, zero-knowledge identification, and public-key encryption protocols using compressed keys (Sections 4.1, 4.2, and 4.3). We present theoretical running times and empirical cost measurements for our implementation (Section 5.3), along with a discussion of applications (Section 6). We present the extension of a previous implementation of isogeny-based key exchange, [6], to include compression and decompression, written in C (Section 5.4). Finally, we compare compressed key sizes with those of other major families of post-quantum cryptographic encryption schemes (Section 5.2).

2 Preliminaries

In this section, we provide required mathematical background to help understanding the proposed contributions. Notations that will be used throughout the paper can be found in Table 1.

2.1 Elliptic curve invariants

Let $E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ be an elliptic curve defined over a finite field $\mathbb{F}_{p^n}$. When $p \neq 2, 3$, we can always write the elliptic curve in short Weierstrass form $E : y^2 = x^3 + Ax + B$. The points $(x, y) \in \mathbb{F}_{p^n} \times \mathbb{F}_{p^n}$ that satisfy the above equation together with the identity point ∞ form a group denoted by $E(\mathbb{F}_{p^n})$. For any $m \in \mathbb{N}$, the subgroup $E[m] = \{P \in E(\mathbb{F}_{p^n}) : mP = \infty\}$ is called the m-torsion subgroup of $E(\mathbb{F}_{p^n})$. This subgroup is the kernel of the multiplication-by-m map, $[m] : E \to E$, which maps P to mP. An important torsion subgroup is $E[p^r]$, for any $r \geq 1$. If $E[p^r] = \{\infty\}$, then we say the curve E is supersingular. Otherwise $E[p^r] \cong \mathbb{Z}/p^r\mathbb{Z}$, for all $r \geq 1$, in which case we say E is ordinary. A useful property of supersingular elliptic curves is that they are always defined over $\mathbb{F}_{p^2}$. Our compression algorithm applies to the cryptosystems described in [6], which use supersingular curves.

The following invariants will be useful later in this work. Define the b-invariants and the c-invariants of this curve:

$$\begin{aligned} b_2 &= a_1^2 + 4a_2, \\ b_4 &= 2a_4 + a_1 a_3, \\ b_6 &= a_3^2 + 4a_6, \\ c_4 &= b_2^2 - 24b_4, \\ c_6 &= -b_2^3 + 36 b_2 b_4 - 216 b_6. \end{aligned}$$

Finally, define the j-invariant by

$$j(E) = 1728 \frac{c_4^3}{c_4^3 - c_6^2} = 1728 \frac{4A^3}{4A^3 + 27B^2} \in \mathbb{F}_{p^n}.$$

Two elliptic curves are isomorphic over $\overline{\mathbb{F}}_p$ if and only if they have the same j-invariant [17].

2.2 Two-dimensional discrete log problem

In this work we consider the following generalization of the discrete logarithm problem on elliptic curves: Given an elliptic curve, E, defined over the finite field $\mathbb{F}_{p^2}$, for some prime number p, together with two points $\{R_1, R_2\}$ generating a subgroup S of $E(\mathbb{F}_{p^2})$, and an element $h \in S$, the two-dimensional discrete log problem is to determine $e_1, e_2 \in \mathbb{N}$ such that $h = e_1 R_1 + e_2 R_2$.

If we denote the largest prime dividing $|S|$ by p_{max}, then there is a variation of the Pohlig-Hellman algorithm [21] that solves this problem with time complexity $O(\sqrt{p_{max}} \log p)$. In the case where $|S|$ is smooth, this algorithm is efficient and practical. The smooth case is sufficient for our needs.

2.3 Isogenies

Two elliptic curves are defined to be isogenous if and only if there is a non-trivial isogeny between them. For every subgroup S of $E(\mathbb{F}_{p^n})$, there is up to isomorphism a unique separable isogeny having S as its kernel. The standard way of computing the codomain of such an isogeny or the action of that isogeny on points is to use Vélu's formulas [22], which involves calculating a summation over all the elements of that subgroup S. In isogeny-based cryptography, such subgroups must be large for security purposes, but also must have smooth order in order to allow efficient calculation. In [6], it was shown how to apply Vélu's formulas to compute isogenies efficiently in such cases.

2.4 Twists of elliptic curves

Two elliptic curves $E_1 : y^2 = x^3 + A_1 x + B_1$ and $E_2 : y^2 = x^3 + A_2 x + B_2$ are isomorphic over $\overline{\mathbb{F}}_{p^n}$ if and only if there is some $u \in \overline{\mathbb{F}}_{p^n} \backslash \{0\}$ such that $A_1 = u^4 A_2$

and $B_1 = u^6 B_2$. It is possible for two elliptic curves each defined over $\mathbb{F}_{p^n}$ to be isomorphic over $\overline{\mathbb{F}}_{p^n}$ but not over $\mathbb{F}_{p^n}$; in such cases, we say that the curves are *twists*. In particular, for $p \neq 2, 3$, a quadratic twist of $E : y^2 = x^3 + Ax + B$ is any curve of the form $E^d : y^2 = x^3 + d^2Ax + d^3B$ for $d \in \mathbb{F}_{p^n} \backslash \mathbb{F}_{p^n}^2$ with $d \neq 0$. The curves E and E^d are isomorphic over the field extension $\mathbb{F}_{p^n}(\sqrt{d})$. Note that $\mathbb{F}_{p^n} \subsetneq \mathbb{F}_{p^n}(\sqrt{d})$ since $d \notin \mathbb{F}_{p^n}^2$. The cases for $j(E) = 1728$ or $j(E) = 0$ are treated separately. If $j(E) = 1728$, then the twist $E^d : y^2 = x^3 + dAx$ is quartic, and if $j(E) = 0$ the twist $E^d : y^2 = x^3 + dB$ is sextic.

2.5 Weil pairing

For $m \in \mathbb{N}$ and $E[m] = \{P \in E(\mathbb{F}_{p^n}) : mP = \infty\}$, recall that the Weil pairing is a map $e \colon E[m] \times E[m] \to \mathbb{F}_{p^n}$, satisfying bilinearity and non-degeneracy:

$$\begin{gathered} e(P_1 + P_2, Q) = e(P_1, Q)e(P_2, Q) \\ e(P, Q_1 + Q_2) = e(P, Q_1)e(P, Q_2) \\ \forall P \in E[m] \backslash \{\infty\}, \exists Q \in E[m] \text{ such that } e(P, Q) \neq 1. \end{gathered}$$

Miller's algorithm [11] provides an efficient way to compute the Weil pairing.

2.6 Curve Conversions

Since the formulas for compression primarily involve short Weierstrass curves and [6] performs isogeny computations in the Montgomery domain, this section will discuss the conversion between the Montgomery curve and short Weierstrass curves to perform the key exchange with compression and decompression, even if the starting basis is a Montgomery curve. We use the formulas given in [6]. For background information on Montgomery curves, refer to [5] or [13]. The conversion between Montgomery curves and short Weierstrass curves utilize points of orders 2 and order 4. A Montgomery curve always has an equivalent short Weierstrass curve, but the opposite is not necessarily true. A short Weierstrass curve will have an equivalent Montgomery curve as long as there are points of order 2 and 4. The Montgomery curve is defined as follows:

$$E : By^2 = x^3 + Ax^2 + x \tag{1}$$

Montgomery to Short Weierstrass As mentioned in [6], the change of coordinates $x = \bar{x}B$, $y = \bar{y}B$ converts from Montgomery form to long Weierstrass form:

$$\bar{E} : B^3\bar{y}^2 = B^3\bar{x}^3 + B^2A\bar{x}^2 + B\bar{x}$$

$$\bar{E} : \bar{y}^2 = \bar{x}^3 + \frac{A\bar{x}^2}{B} + \frac{\bar{x}}{B^2}$$

To convert from long Weierstrass to short Weierstrass, the additional change of coordinates $\bar{x} = \tilde{x} - A/3B$, $\bar{y} = \tilde{y}$ is made:

$$\tilde{E} : \tilde{y}^2 = (\tilde{x} - \frac{A}{3B})^3 + \frac{A(\tilde{x} - \frac{A}{3B})^2}{B} + \frac{(\tilde{x} - \frac{A}{3B})}{B^2}$$

$$\tilde{E} : \tilde{y}^2 = \tilde{x}^3 + \frac{3 - A^2}{3B^2}\tilde{x} + \frac{2A^3 - 9A}{27B^3} \tag{2}$$

Therefore, the total change of coordinates from Montgomery curves to short Weierstrass curves is $\tilde{x} = x/B + A/3B$, $\tilde{y} = y/B$. The change of curve coefficients is $a = (3 - A^2)/(3B^2)$, $b = (2A^3 - 9A)/(27B^3)$. Overall, the conversion from Montgomery coordinates to short Weierstrass curves is relativley cheap. The conversion requires inversions, multiplications, and squarings, but is constant time complexity.

Short Weierstrass to Montgomery A short Weierstrass curve requires a point of order 2 and a point of order 4 for the conversion to Montgomery coordinates, and by our choice of prime p, these points will always exist. To find a point of order 4, $P_4 = (\tilde{x}_4, \tilde{y}_4)$, we use the same approach to calculate a basis, which is discussed in Section 3.2. The point of order 4 is doubled to find a corresponding point of order 2, $P_2 = 2P_4 = (\tilde{x}_2, \tilde{y}_2)$. Since $2P_2 = O = 2(-P_2)$, $P_2 = -P_2$, $\tilde{y}_2 = -\tilde{y}_2$,so $\tilde{y}_2 = 0$. Therefore, $\tilde{x}_2^3 + a\tilde{x}_2 + b = 0$. From this notion, the change of coordinates $\tilde{x} = \bar{x} + \tilde{x}_2$, $\tilde{y} = \bar{y}$ can be used as a way to convert from short Weierstrass to long Weierstrass:

$$\bar{E} : \bar{y}^2 = (\bar{x} + \tilde{x}_2)^3 + a(\bar{x} + \tilde{x}_2) + b$$

$$\bar{E} : \bar{y}^2 = \bar{x}^3 + 3\tilde{x}_2\bar{x}^2 + (3\tilde{x}_2^2 + a)\bar{x} + (\tilde{x}_2^3 + a\tilde{x}_2 + b)$$

The inverse of the difference between the x coordinates of the points of order 4 and order 2 is used as a major conversion factor between the two curve domains:

$$\beta = \frac{1}{\tilde{x}_4 - \tilde{x}_2} \tag{3}$$

With this value, the final change of coordinates $\bar{x} = x/\beta$, $\bar{y} = y/\beta$ to convert the corresponding Montgomery curve:

$$E : \frac{y^2}{\beta^2} = \frac{x^3}{\beta^3} + \frac{3\tilde{x}_2\beta x^2}{\beta^2} + \frac{x}{\beta} + (\tilde{x}_2^3 + a\tilde{x}_2 + b)$$

$$E : \beta y^2 = x^3 + 3\tilde{x}_2\beta x^2 + x \tag{4}$$

Therefore, the total change of coordinates from short Weierstrass curves to Montgomery curves is $x = \beta(\tilde{x} - \tilde{x}_2)$, $y = \beta\tilde{y}$. The corresponding curve coefficients are $A = 3\tilde{x}_2\beta$, $B = \beta$. The above conversion works with this protocol because there are $p+1$ points on the curve, which has many factors of 2 which guarantees the presence of multiple points of order 4. The conversion from short Weierstrass to Montgomery curves is expensive in that a point of order 4 has to be found through probabilistically searching. A different approach could be to factor the division polynomial ψ_4.

3 Compression of Public Information

The term "public information" has a different definition depending on the cryptosystem being discussed. In Section 4 we explain how our compression techniques can be used in a key-exchange protocol, a public-key encryption scheme, and a zero-knowledge identification protocol. Even though the structure of these cryptosystems differ, they each have some form of public information. The key exchange protocol has a public transcript, the public-key encryption scheme has a public key, and in the zero-knowledge proof of identity the participating parties communicate multiple times to one another over a public channel. The compressed public information in each setting is approximately half the size it was in previous work. For full details of the mathematical operation of the protocols, we refer to [6] or to the detailed protocol descriptions given in Section 4.

Let p be a prime of the form $\ell_A^{e_A}\ell_B^{e_B} \cdot f \pm 1$, where ℓ_A and ℓ_B are small prime numbers. In general, the public information for (say) Alice consists of her elliptic curve $E_A : y^2 = x^3 + Ax + B$, defined over $\mathbb{F}_{p^2}$, together with two points on that curve, $\phi_A(P_B)$ and $\phi_A(Q_B)$. To convey this information in a space-efficient manner, Alice would normally send the field elements A and B to describe the curve, and the x-coordinates of $\phi_A(P_B)$ and $\phi_A(Q_B)$ to describe the points. From the x-coordinate alone Bob can determine the point $(x, \pm y)$, and the sign ambiguity is immaterial for isogeny-based cryptosystems, since both choices of sign yield the same kernel subgroup.

In order to decrease the number of bits required for the participants to send, we propose two improvements to the transmission mechanism.

3.1 First improvement

Instead of sending the coefficients A and B to Bob, we suggest Alice sends $j(E_A)$. This alternate method requires the same amount of space as sending the j-invariant, In order to use this approach, we must solve two problems:

1. Given a j-invariant in $\mathbb{F}_{p^2}$, both Alice and Bob need some method to determine a canonical choice of curve E_A^* isomorphic to E_A.
2. Alice will also have to compute the isomorphism $\psi_A : E_A \to E_A^*$, in order to work out which points on E_A^* correspond to $\phi_A(P_B)$ and $\phi_A(Q_B)$.

Solutions:

1. Given a particular $j \in \mathbb{F}_{p^n}$, the following formulas [17] determine an elliptic curve E_j having j-invariant equal to j:

   Case 1: If $j = 0$, then $E_j : y^2 + y = x^3$,
   Case 2: If $j = 1728$, then $E_j : y^2 = x^3 + x$,
   Case 3: Otherwise, $E_j : y^2 + xy = x^3 - \frac{36}{j-1728}x - \frac{1}{j-1728}$.

   Notice neither Case 1 nor Case 3 yield equations in short Weierstrass form $y^2 = x^3 + Ax + B$, but since $p \neq 2, 3$, we can rewrite these curves in the form $y^2 = x^3 - 27c_4x - 54c_6$ using the c-invariants c_4 and c_6 described in Section 2.1. In all cases, this curve is guaranteed to be isomorphic to E over $\overline{\mathbb{F}}_{p^2}$ [17], but may not be isomorphic over $\mathbb{F}_{p^2}$ itself.
   Alice will convey either E_j or a twist E_j^d of E_j to Bob, depending on which one is isomorphic to E_A over $\mathbb{F}_{p^2}$. In order for Bob to obtain the same curve, both parties will have to use the same element d; note however that this element can be pre-computed as a public parameter. In addition, Alice needs to communicate one bit of information to Bob to indicate whether or not the twist was used.
2. If E_A and E_A^* are isomorphic over $\mathbb{F}_{p^2}$, then the isomorphism between them is easy to compute. Simply write out their equations

$$E_A : y^2 = x^3 + A_1x + B_1$$
$$E_A^* : y^2 = x^3 + A_2x + B_2$$

   and look for a field element $d' \in \mathbb{F}_{p^2}$ such that $A_1 = d'^4 A_2$. If $A_1 = A_2 = 0$, then use the B_i's instead, such that $B_1 = d'^6 B_2$.

An alternate approach would be to find an elliptic curve isomorphic to E_A with $A = -3$ (this occurs exactly when $h := 1 - \frac{1728}{j(E_A)}$ is a quadratic residue in $\mathbb{F}_{p^2}$), and then Alice would send only the coefficient $B = 2\sqrt{h}$. Suppose h does not have a square root in $\mathbb{F}_{p^2}$. Let $\beta \in \mathbb{F}_{p^2}$ be a global parameter with a cube root in $\mathbb{F}_{p^2}$ but no square root. Then the product $h\beta$ will be a quadratic residue, and Alice can send $B = 2\sqrt{h\beta}$ and an additional bit to alert Bob that she used β. Bob now can compute the elliptic curve $E_{A2} : y^2 = x^3 - 3\beta^{1/3}x + 2\sqrt{\beta(1 - \frac{1728}{j(E_A)})}$ which is isomorphic to E_A, if a specific root $\beta^{1/3}$ is publicly determined as well. This alternate method requires the same amount of space as sending the j-invariant, slightly increases the amount of computation Alice must do, but decreases the length of Bob's computation to determine the short Weierstrass form of E_A.

3.2 Second improvement

Instead of sending the x-coordinates of the points $\phi_A(P_B)$ and $\phi_A(Q_B)$, Alice sends the coefficients α_1, β_1, α_2, $\beta_2 \in \mathbb{Z}/\ell_B^{e_B}\mathbb{Z}$ of the representations of these points with respect to some fixed basis. Again, this raises two problems.

1. Given a supersingular elliptic curve, E, defined over $\mathbb{F}_{p^2}$, both Alice and Bob need to be able to determine the same basis $\{R_1, R_2\}$ for $E[\ell_B^{e_B}]$.

2. Given an elliptic curve E_A, two points $\phi_A(P_B)$, $\phi_A(Q_B)$, an isomorphism $\psi_A : E_A \to E_A^*$, and a basis $\{R_1, R_2\}$ for $E_A^*[\ell_B^{e_B}]$, Alice must compute $\alpha_1, \beta_1, \alpha_2, \beta_2 \in \mathbb{Z}/\ell_B^{e_B}\mathbb{Z}$ such that $\psi_A(\phi_A(P_B)) = \alpha_1 R_1 + \beta_1 R_2$, and $\psi_A(\phi_A(Q_B)) = \alpha_2 R_1 + \beta_2 R_2$.

Solutions:

1. We will explain how to find a basis for $E[\ell_A^{e_A}]$; the case of $E[\ell_B^{e_B}]$ is similar. Using a deterministic pseudo-random number generator, choose a random point $P \in_R E(\mathbb{F}_{p^2})$ and multiply it by $\ell_B^{e_B} \cdot f$ to obtain a point P'. The order of P' divides $\ell_A^{e_A}$, and with high probability, is exactly $\ell_A^{e_A}$. To check the order of P' efficiently, one can multiply P' by powers of ℓ_A until the identity is found. If the order is $\ell_A^{e_A}$, then set $R_1 = P'$, otherwise pick another random P and start again. To find R_2, repeat this process with random points $Q \in_R E(\mathbb{F}_{p^2})$ until a point Q' of order $\ell_A^{e_A}$ is found, and check whether it is independent of R_1 by computing the Weil pairing of R_1 and Q'. If the pairing evaluates to anything other than 1, then set $R_2 = Q'$, otherwise find another Q'. The bases that Alice and Bob find will be the same, as long as they both choose the same random points P and Q in the same order, which will be achieved if Alice and Bob provide to PRNG with identical seed values.
2. Using the Pohlig-Hellman algorithm, we can efficiently compute the discrete logarithm over finite extensions of fields with small prime characteristic. Suppose $\psi_A(\phi_A(P_B)) = \alpha R_1 + \beta R_2$. Then, using the (efficiently computable) Weil pairing:

$$\begin{aligned} e(R_1, \psi_A(\phi_A(P_B))) &= e(R_1, \alpha R_1 + \beta R_2) \\ &= e(R_1, \alpha R_1) e(R_1, \beta R_2) \\ &= e(R_1, R_1)^{\alpha} e(R_1, R_2)^{\beta} \\ &= e(R_1, R_2)^{\beta}, \end{aligned}$$

   Hence solving the discrete logarithm of $e(R_1, \psi_A(\phi_A(P_B)))$ with respect to $e(R_1, R_2)$ will output the value of β. With β known, α can be computed by taking the discrete logarithm of $e(R_2, \psi_A(\phi_A(P_B)) - \beta R_2)$ with respect to $e(R_2, R_1)$ since

$$\begin{aligned} e(R_2, \psi_A(\phi_A(P_B)) - \beta R_2) &= e(R_2, \alpha R_1) \\ &= e(R_2, R_1)^{\alpha} \end{aligned}$$

   Both of these discrete log computations can be performed efficiently with the Pohlig-Hellman algorithm [14], since they take place in a group of order $\ell_B^{e_B}$, which is smooth.

Because $j(E_A)$ is in $\mathbb{F}_{p^2}$, it takes $2 \log p$ bits to express. Each of A and B is also in $\mathbb{F}_{p^2}$ and is $2 \log p$ bits, so sending the j-invariant requires only half as many bits as sending both A and B. In addition, each of $\alpha_1, \beta_1, \alpha_2, \beta_2$ has size $e_B \log \ell_B \approx \frac{1}{2} \log p$, which means sending all four to Bob will only require $2 \log p$ bits, half as many as required if sending the x-coordinates of each point (each x-coordinate is $2 \log p$ bits, and there is one for $\phi_A(P_B)$ and one for $\phi_A(Q_B)$). Overall, we achieve a reduction in the size of the public information, from $8 \log p$ bits to $4 \log p$ bits.

4 Isogeny-Based Cryptography with compressed keys

In this section, we explain how to apply the compression techniques (Section 3) to public-key isogeny-based cryptosystems: key exchange, zero-knowledge proof of identity, and public-key encryption.

4.1 Key-exchange with compressed keys

Setup: Fix $\mathbb{F}_{p^2}$ as the field of definition, where p is a prime number of the form $\ell_A^{e_A}\ell_B^{e_B} \cdot f \pm 1$, ℓ_A and ℓ_B are small prime numbers, and f is chosen so that p is prime. Fix a supersingular curve, $E : y^2 = x^3 + a_4x + a_6$, defined over $\mathbb{F}_{p^2}$, having cardinality $(p \mp 1)^2 = (\ell_A^{e_A}\ell_B^{e_B} \cdot f)^2$, which can be done efficiently [3]. Fix a non-square element $d \in \mathbb{F}_{p^2}$. Lastly, fix a basis P_A, Q_A which generates $E[\ell_A^{e_A}]$, and a basis P_B, Q_B which generates $E[\ell_B^{e_b}]$.

Compression of public information: 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. Using Vélu's formulas, she computes $E_A, \phi_A(P_B)$, and $\phi_A(Q_B)$ where $\ker(\phi_A) = \langle [m_A]P_A + [n_A]Q_A \rangle$. Normally, Alice would just send $E_A, \phi_A(P_B)$, and $\phi_A(Q_B)$ to Bob, but we now add key compression. Alice computes the canonical curve E_{j_A} from $j(E_A)$, along with

$$E_{j_A}^* : y^2 = x^3 - 27c_1(E_{j_A})x - 54c_2(E_{j_A})$$

to put E_{j_A} in short Weierstrass form. If E_A is not isomorphic to $E_{j_A}^*$ over $\mathbb{F}_{p^2}$, then she sets $T_A = 1$ and computes the twist

$$E_A^* : y^2 = x^3 + d^2a_4(E_{j_A}^*)x + d^3a_6(E_{j_A}^*)$$

of $E_{j_A}^*$. Otherwise she sets E_A^* to $E_{j_A}^*$, and T_A to 0.

Next, Alice computes the isomorphism $\psi_A : E_A \to E_A^*$ and canonical basis $\{R_1, R_2\}$ for $E_A^*[\ell_B^{e_B}]$. Finally, she solves the 2-dimensional discrete log problem to determine $\alpha_1, \beta_1, \alpha_2, \beta_2 \in \mathbb{Z}/\ell_B^{e_B}\mathbb{Z}$ such that

$$\begin{aligned} \alpha_1R_1 + \beta_1R_2 = \psi_A(\phi_A(P_B)), \alpha_2R_1 + \beta_2R_2 \\ = \psi_A(\phi_A(Q_B)) \end{aligned}$$

Alice's compressed public information is equal to the tuple $(j(E_A), \alpha_1, \beta_1, \alpha_2, \beta_2, T_A)$, and her private key is (m_A, n_A). She exchanges this information with Bob, who in turn sends Alice his public information consisting of $(j(E_B), \gamma_1, \kappa_1, \gamma_2, \kappa_2, T_B)$, where

- $\{S_1, S_2\}$ is the canonical basis for $E_B^*[\ell_A^{e_A}]$,
- $\gamma_1S_1 + \kappa_1S_2 = \psi_B(\phi_B(P_A))$,
- $\gamma_2S_1 + \kappa_2S_2 = \psi_B(\phi_B(Q_A))$,
- $\ker(\phi_B) = \langle [m_B]P_B + [n_B]Q_B \rangle$,
- $\psi_B \colon E_B \to E_B^*$,

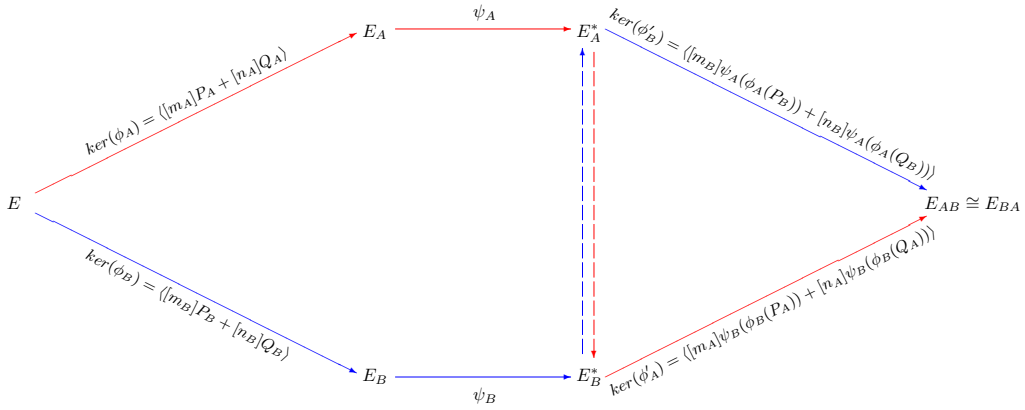

Fig. 1. Key exchange protocol with compression

and T_B is 1 if a twist is required, and 0 otherwise.

Decompression and computing a shared secret key: Alice determines E_B^* by computing the canonical curve associated with $j(E_B)$, putting it in short Weierstrass form, and computing a twist depending on the bit from Bob. After computing the canonical basis $\{S_1, S_2\}$ for $E_B^*[\ell_A^{e_A}]$, Alice uses $\gamma_1, \kappa_1, \gamma_2, \kappa_2$ to compute $\psi_B(\phi_B(P_A))$ and $\psi_B(\phi_B(Q_A))$. Using Vélu's formulas once more, Alice computes the isogeny

$$\phi'_A : E_B^* \to E_{AB},$$

with $\ker(\phi'_A) = \langle [m_A]\psi_B(\phi_B(P_A)) + [n_A]\psi_B(\phi_B(Q_A)) \rangle$.

Bob then computes the same curve $E_{BA} = E_A^*/\langle [m_B]\psi_A(\phi_A(P_B)) + [n_B]\psi_A(\phi_A(Q_B)) \rangle$ by doing an analogous decompression operation. Afterwards, both Alice and Bob possess the shared secret key $j(E_{AB}) = j(E_{BA}) \in \mathbb{F}_{p^2}$.

4.2 Zero-knowledge proof of identity with compressed information

Here we outline how to use the compression technique to reduce the amount of information sent in each round of an isogeny-based zero-knowledge proof of identity. Let p be a prime number of the form $\ell_A^{e_A}\ell_B^{e_B} \cdot f \pm 1$, where ℓ_A and ℓ_B are small prime numbers, and f is chosen so that p is prime. Throughout this subsection let E^* denote the canonical curve associated to the j-invariant of the isomorphism class of E.

Secret parameters: A supersingular curve E defined over $\mathbb{F}_{p^2}$, a primitive $\ell_A^{e_A}$-torsion point S defining an isogeny $\phi : E \to E/\langle S \rangle$, and an isomorphism $\phi_0 : E/\langle S \rangle \to (E/\langle S \rangle)^*$.

Public parameters: The elliptic curves E and $E/\langle S \rangle$, generators P, Q for $E[\ell_A^{e_A}]$ and the points $\phi_0(\phi(P))$ and $\phi_0(\phi(Q))$.

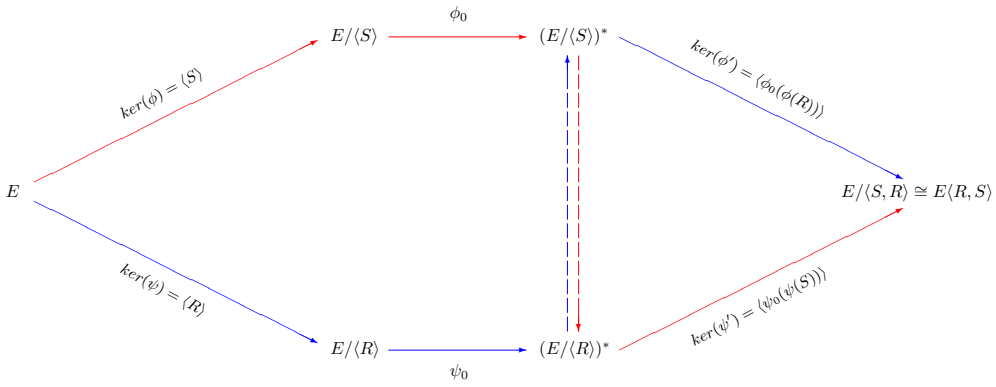

Fig. 2. Zero-knowledge proof of identity with compression

Identification: Repeat m times:

1. Peggy picks $R \in E[\ell_B^{e_B}]$ and, using Vélu's formulas, computes the elliptic curves
$$E/\langle R\rangle, (E/\langle R\rangle)^*, E/\langle S,R\rangle,$$
and the isogenies
$$\psi : E \to E/\langle R\rangle, \phi' : (E/\langle S\rangle)^* \to E/\langle S,R\rangle$$
$$\psi' : (E/\langle R\rangle)^* \to E/\langle S,R\rangle.$$
She also computes the isomorphism
$$\psi_0 : E/\langle R\rangle \to (E/\langle R\rangle)^*.$$
2. Peggy sends $j(E/\langle R\rangle)$ and $j(E/\langle S,R\rangle)$ to Victor.
3. Victor selects a random bit b and sends it to Peggy.
4. If $b = 0$, then Peggy computes the canonical basis $\{B_1, B_2\}$ for $E[\ell_B^{e_B}]$ and the canonical basis $\{B_1', B_2'\}$ for $(E/\langle S\rangle)^*[\ell_B^{e_B}]$. Peggy then sends the values $\alpha_1, \beta_1, \alpha_2, \beta_2 \in Z/\ell_B^{e_B}Z$ to Victor, where $\alpha_1 B_1 + \beta_1 B_2 = R$, and $\alpha_2 B_1' + \beta_2 B_2' = \phi_0(\phi(R))$.
5. If $b = 1$, then Peggy computes the canonical basis $\{B_1'', B_2''\}$ for $(E/\langle R\rangle)^*[\ell_A^{e_A}]$. Peggy then sends $\alpha_3, \beta_3 \in Z/\ell_A^{e_A}Z$ to Victor, such that $\alpha_3 B_1'' + \beta_3 B_2'' = \psi_0(\psi(S))$.

4.3 Key compression for public-key cryptography

Setup: The setup is identical to the key-exchange system described above, except this cryptosystem also requires a family of hash functions, $\mathcal{H} = \{H_k : k \in K\}$, from $\mathbb{F}_{p^2}$ to the message space $\{0,1\}^w$, indexed by a finite set K.

Compressed 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, and a random $k \in_R K$. As in Section 4.1, compute and

publish the tuple $(j(E_A), \alpha_1, \beta_1, \alpha_2, \beta_2, T_A)$ as the public key, and store the private key tuple (m_A, n_A, k).

Decompression: Given a public key $(j(E_A), \alpha_1, \beta_1, \alpha_2, \beta_2, T_A, k)$ it is described above (4.1) how to decompress to the tuple $(E_A^*, \psi_A(\phi_A(P_B)), \psi_A(\phi_A(Q_B)), k)$.

Encryption: Given the decompressed public key $(E_A^*, \psi_A(\phi_A(P_B)), \psi_A(\phi_A(Q_B)), k)$, and message $m \in \{0,1\}^w$, the sender chooses $m_B, n_B \in_R \mathbb{Z}/\ell_B^{e_B}\mathbb{Z}$ not both divisible by ℓ_B. Next, as in Section 4.1, the sender computes the values $E_B, \phi_B(P_A)$, $\phi_B(Q_A), E_B^*, T_B, \psi_B, \{S_1, S_2\}$, the coefficients $\gamma_1, \kappa_1, \gamma_2, \kappa_2 \in \mathbb{Z}/\ell_A^{e_A}\mathbb{Z}$ such that $\gamma_1 S_1 + \kappa_1 S_2 = \psi_B(\phi_B(P_A))$, and $\gamma_2 S_1 + \kappa_2 S_2 = \psi_B(\phi_B(Q_A))$, and $\phi'_B : E_A^* \to E_{BA}$, with $\ker(\phi'_B) = \langle [m_B]\psi_A(\phi_A(P_B)) + [n_B]\psi_A(\phi_A(Q_B))\rangle$. The sender then sets

$$c = H_k(j(E_{BA})) \oplus m.$$

The ciphertext is the tuple $(c, j(E_B), \gamma_1, \kappa_1, \gamma_2, \kappa_2, T_B)$.

Decryption: Given a ciphertext $(c, j(E_B), \gamma_1, \kappa_1, \gamma_2, \kappa_2, T_B)$ and private key (m_A, n_A, k), compute the curve E_B^* from $j(E_B)$ and T_B, the points $\psi_B(\phi_B(P_A))$ and $\psi_B(\phi_B(Q_A))$ from $\gamma_1, \kappa_1, \gamma_2, \kappa_2$, and the isogeny $\phi'_A : E_B^* \to E_{AB}$, with $\ker(\phi'_A) = \langle [m_A]\psi_B(\phi_B(P_A)) + [n_A]\psi_B(\phi_B(Q_A))\rangle$. The plaintext is

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

5 Complexity and Security

In this section we discuss the impact of key compression on isogeny-based cryptosystems, in terms of security, key size, and running time costs.

5.1 Security

The compression and decompression operations require no private key material, and indeed the operations can be unilaterally performed by any party or any observer. Accordingly, the operations cannot have any effect whatsoever on the security of the underlying scheme.

5.2 Space requirements

Our compressed keys require $4\log p + 1$ bits to store (the extra bit is to convey if a twist was used or not), as explained in Section 3. The best known quantum attack against this scheme [20] has complexity $O(\sqrt[6]{p})$, and the classical version runs in $O(\sqrt[4]{p})$. This formula allows us to compute the minimum number of bits required to store the compressed keys for any given level of security.

In Table 2 we present a comparison between the size of our compressed keys and the minimum key size required for the 128-bit and 256-bit security level from

Table 2. Public key sizes in bits

	Security level	
Scheme	128-bit	256-bit
NTRU [10]	4 939	11 957
Ring-LWE [18]	7 498	15 690
McEliece (Goppa)[1]	1 991 880	9 276 241
McEliece (QC-MDPC) [12]	9 857	32 771
Isogenies	3 073	6 145

two other major families of post-quantum encryption primitives: Lattice-based and Code-based.

The key sizes in Table 2 for lattice and code-based schemes are based on classical attacks, because we took these key size estimates from sources which only perform classical security analyses. Quantum attacks may be slightly faster: for instance [9] demonstrates a 2^{112} quantum attack against the "128-bit" NTRU parameters. By contrast, isogeny-based cryptosystems are motivated mainly by post-quantum applications, and our security analysis and key sizes for isogeny-based cryptosystems are based on quantum attacks. Against classical attacks, key sizes for isogeny-based cryptosystems can be further reduced by 33%.

5.3 Computational requirements

It is computationally easy to compute j-invariants, canonical curves from j-invariants, twists, scalar multiples of points on elliptic curves, isomorphisms between elliptic curves, sums of points on elliptic curves, and to put elliptic curves in Weierstrass form. These costs are relatively negligible compared to the costs of computing a basis and performing discrete logarithms, so we ignore them.

The basis computation is done probabilistically, with expected running time $O(e_A)$, since for a random point P, the point $(\ell_B^{e_B} \cdot f)P$ has order $\ell_A^{e_A}$ with probability $1/e_A$. As for discrete logarithms, to solve two discrete logarithms in $\mathbb{F}_{p^2}$ one can use Pohlig-Hellman twice (explained in Section 3.2), each with a runtime of $O(e_A^2 + e_A\sqrt{\ell_A})$ [14]. To compress her information, Alice performs both of the above steps. After she has exchanged information with Bob, she performs another basis computation, with cost $O(e_A)$, to decompress. This gives a total theoretical runtime of $O(e_A\sqrt{\ell_A} + e_A^2)$.

We remark that, assuming cheap storage and expensive bandwidth, the logical strategy is to store both compressed and uncompressed copies of the key, and transmit only compressed copies. In this scenario, the computational costs of compression and decompression are incurred only once per key; for compression, once per key for the lifetime of the key, and for decompression, once per key per recipient. Unlike space-saving strategies with NTRU and LWE, public key compression imposes no per-message encryption or decryption overhead. NTRU and LWE also have a nonzero probability of decryption failure (they are based on adding error vectors into the ciphertext, and occasionally the error overcomes the intended signal), which causes a tradeoff between security, efficiency, and error-rate, and represents

Table 3. Isogeny Key Exchange Primes

$p = \ell_A^{e_A} \ell_B^{e_B} f \pm 1$	Key Size (bits)
$2^{258}3^{161}186 - 1$	512
$2^{386}3^{242}2 - 1$	768
$2^{514}3^{323}353 - 1$	1024

a limiting factor in reducing key size for these two schemes, whereas isogeny-based encryption schemes have no possibility of mathematical error.

5.4 Software Implementations

The C implementation of the isogeny-based key exchange protocol in [6] was extended to include the key compression and decompression schemes introduced in this paper. The original implementation used GNU Multiprecision Library to handle large number operations. The reliance on this library limits the speed of the total computation, but allows for portability. The original implementation implemented the key exchange over Montgomery curves because of their fast isogeny computations. This work deals primarily with short Weierstrass curves, so an extra layer to convert from Montgomery curves to short Weierstrass curves before the key compression and an extra layer to convert from the short Weierstrass curves back to Montgomery curves after the key decompression were both added to integrate the two new protocols. The primes that were used to test are shown in Table 3.

The C code only requires access to OpenSSL and the GNU Multiprecision Library to compile. To evaluate the performance of the proposed compression and decompression schemes, the C code was implemented on an ARM processor and an Intel processor. Tables 4 and 5 show the timing results for the key exchange and individual compression and decompression times on a Jetson TK1 and an Intel i7-4790K, respectively. The compression and decompression utilize probabilistic algorithms, so the median time over many iterations was used as a benchmark. The compression algorithm relies on the elliptic curve discrete log to factor the double point multiplication instead of a pairing approach to the discrete log over finite fields, which results in slower times. As a side note, the compression time includes the conversion from Montgomery to short Weierstrass and the decompression time includes the conversion from short Weierstrass to Montgomery.

The NVIDIA Jetson Tegra K1 development board represents the expected timing for a medium tier embedded device. The board features a a quad-core ARM Cortex-A15 CPU under the ARM v7 microarchitecture, which runs at 2.3 GHz.

The Intel i7-4790K represents the expected timing for the high tier of personal computers. The Intel processor is a quad-core desktop processor under the Haswell microarchitecture, running at 4.0 GHz.

These results show that the decompression time is roughly the amount of the key exchange protocol, but the compression time is larger by a factor of magnitude. The compression for Bob is typically faster than Alice's because the elliptic curve discrete log over base 3 requires far fewer rounds than the elliptic curve discrete log over base 2. Decompression is relatively fast, demonstrating the effectiveness of the

Table 4. ARM Implementations on Jetson TK1 Device

Prime size (bits)	512	768	1024
Classical security (bits)	128	192	256
Quantum security (bits)	85	128	168
Key Exchange (sec)	0.291	0.808	1.749
Alice Compression (sec)	1.816	6.906	18.042
Bob Compression (sec)	2.324	8.582	22.991
Alice Decompression (sec)	0.311	0.811	1.969
Bob Decompression (sec)	0.284	0.772	1.488

Table 5. PC Implelementations on Intel i7-4790K

Prime size (bits)	512	768	1024
Classical security (bits)	128	192	256
Quantum security (bits)	85	128	168
Key Exchange (sec)	0.0540	0.1337	0.2669
Alice Compression (sec)	0.4529	1.5203	3.7020
Bob Compression (sec)	0.5740	1.9368	4.6312
Alice decompression (sec)	0.0580	0.1348	0.2493
Bob Decompression (sec)	0.0530	0.1233	0.2364

protocol with decompression on a compressed key at rest, for instance. Overall, the C implementation results are promising and can be further improved in the future.

6 Applications

The most relevant application today for reduced key sizes is when there is an upper bound in the protocol on the amount of bits allowed to be sent at a time. A few popular systems that have this restriction in place include Tor and Bitcoin. It is also usually preferable to send as little information as possible in systems like SSH and SSL.

Tor is currently the most widely deployed software for anonymous communication, which it achieves by directing internet traffic through thousands of relays. Each cell of data in Tor's onion rooting network must be less than 514 bytes [7], and public keys are transmitted within blocks of this size. Compared to isogeny-based cryptography, no other known quantum-resistant cryptosystem can function well under this restriction. Recently, [15] showed how to incorporate NTRUEncrypt into the ntor protocol, but only after increasing the cell size. For a security level of 128, the scheme of [15] requires a handshake message of approximately 600 bytes. By contrast, compressed keys at the 128-bit level of security for isogeny-based cryptography require only 384 bytes to store and send (see Table 2).

Both SSH and SSL currently provide confidentiality or privacy using public-key encryption schemes which are not secure against quantum attacks. Isogeny-based public-key encryption using our compression method is extremely space efficient, providing a strong candidate for quantum-resistant deployment of these protocols.

7 Conclusion

We propose a method to compress the public transcript for isogeny-based key-exchange, the public keys for isogeny-based public-key encryption, and the public information in each round of an isogeny-based zero-knowledge proof of identity. In each case in this family of cryptosystems, the compressed information is half the size it was in previous work. Our compression routines do not affect the security of the systems, since compression is done with entirely public information. The small key sizes come at the expense of a modest computational cost, which we measured and find small enough to represent an acceptable tradeoff in many contexts.

8 Acknowledgement

The authors would like to thank the reviewers for their constructive comments. This work was supported by the NSERC CREATE Training Program in Building a Workforce for the Cryptographic Infrastructure of the 21st Century (CryptoWorks21), and Public Works and Government Services Canada. Also, this material is based upon work supported by the National Science Foundation under award No. CNS-1464118 and and by the US Army Research Laboratory under award No. W911NF-16-1-0204. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

1. Paulo S.L.M. Barreto. *A Panorama of Post-quantum Cryptography*. Springer, 2014.
2. Jean-François Biasse, David Jao, and Anirudh Sankar. A quantum algorithm for computing isogenies between supersingular elliptic curves. In *Progress in Cryptology – INDOCRYPT 2014*, volume 8885 of *Lecture Notes in Computer Science*, pages 428–442. Springer International Publishing, 2014.
3. Reinier Broker. Constructing supersingular elliptic curves. *J. Comb. Number Theory*, 1:269–273, 2009.
4. Andrew Childs, David Jao, and Vladimir Soukharev. Constructing elliptic curve isogenies in quantum subexponential time. *J. Math. Cryptol*, 8:1–29, 2014.
5. Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, and Frederik Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. Chapman & Hall/CRC, 2012.
6. Luca De Feo, David Jao, and Jérôme Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *Journal of Mathematical Cryptology*, 8(3):209–247, 2014.
7. Roger Dingledine and Nick Mathewson. Tor protocol specification. https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt.
8. Dieter Fishbein. Machine-level software optimization of cryptographic protocols. Master's thesis, University of Waterloo, 2014. http://hdl.handle.net/10012/8400.
9. Scott Fluhrer. Quantum cryptanalysis of NTRU. Cryptology ePrint Archive, Report 2015/676, 2015.
10. P. Hirschhorn, J. Hoffstein, N. Howgrave-Graham, and W. Whyte. Choosing NTRUEncrypt parameters in light of combined lattice reduction and MITM approaches. In *In Proc. ACNS 2009, LNCS 5536*, pages 437–455. Springer-Verlag, 2009.

11. Victor Miller. Short programs for functions on curves. *Unpublished*, 1986.
12. Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo S. L. M. Barreto. MDPC-McEliece: New McEliece Variants from Moderate Density Parity-Check Codes. Cryptology ePrint Archive, Report 2012/409, 2012.
13. Katsuyuki Okeya, Hiroyuki Kurumatani, and Kouichi Sakurai. Elliptic curves with the montgomery-form and their cryptographic applications. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography*, volume 1751 of *Lecture Notes in Computer Science*, pages 238–257. Springer Berlin Heidelberg, 2000.
14. Stephen Pohlig and Martin Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Transactions on Information Theory*, 1978.
15. John Schanck, William Whyte, and Zhenfei Zhang. A quantum-safe circuit-extension handshake for Tor. Cryptology ePrint Archive, Report 2015/287, 2015.
16. Peter Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26:1484–1509, 1997.
17. Joseph Silverman. *The Arithmetic of Elliptic Curves*. Springer, 1986.
18. Vikram Singh. A practical key exchange for the internet using lattice cryptography. Cryptology ePrint Archive, Report 2015/138, 2015.
19. Anton Stolbunov. Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves. *NISK*, pages 97–109, 2009.
20. Seiichiro Tani. Claw finding algorithms using quantum walk. *Theoretical Computer Science*, 410:5285–5297, 2009.
21. Edlyn Teske. The Pohlig-Hellman method generalized for group structure computation. *J. Symbolic Computation*, pages 521–534, 1999.
22. Jacques Vélu. Isogénies entre courbes elliptiques. *C. R. Acad. Sci. Paris Sér. A-B*, 273:A238–A241, 1971.
23. Wikipedia. Post-quantum cryptography — Wikipedia, the Free Encyclopedia, 2015. https://en.wikipedia.org/w/index.php?title=Post-quantum_cryptography&oldid=684559487, [Online; accessed 14-October-2015].
24. Wikipedia. Supersingular isogeny key exchange — Wikipedia, the Free Encyclopedia, 2015. https://en.wikipedia.org/w/index.php?title=Supersingular_Isogeny_Key_Exchange&oldid=679293871, [Online; accessed 14-October-2015].

Appendix: Key size claims in Wikipedia

As of this writing, one can find claims on Wikipedia of 3072-bit public key sizes for isogeny-based cryptosystems at the 128-bit security level, e.g. [23,24]. A closer reading of the articles in question demonstrates clearly that these claims are based on a misunderstanding. For example, [24] states:

> ... A and B will each transmit information 2 (mod p) coefficients defining an elliptic curve and 2 elliptic curve points. Each elliptic curve coefficient requires $\log_2 p$ bits. Each elliptic curve point can be transmitted in $\log_2 p + 1$ bits. Hence the transmission is $4 \log_2 p + 2$ bits. For a 768-bit modulus p for the elliptic curve this is 3074 bits ...

This calculation is incorrect, because it assumes the public key curve is defined over $\mathbb{F}_p$; in reality, the public key curve is defined over $\mathbb{F}_{p^2}$, so that field elements require double the claimed bitlengths. Our work, on the other hand, achieves reduced public-key sizes correctly, without any inaccuracies or misunderstandings.

Appendix: Security assumptions

For completeness, we state the security assumptions under which the security of isogeny-based cryptosystems can be proved. The corresponding security proofs can be found in [6].

Problem 1 Let $\phi_A : E \to E_A$ be an isogeny whose kernel is $\langle [m_A]P_A + [n_A]Q_A \rangle$, where m_A and n_A are randomly chosen from $\mathbb{Z}/\ell_A^{e_A}\mathbb{Z}$ and are not both divisible by ℓ_A. Given E_A and the values $\phi_A(P_B), \phi_A(Q_B)$, the *Supersingular Isogeny (SSI)* problem is to find a generator R_A of $\langle [m_A]P_A + [n_A]Q_A \rangle$.

Problem 2 Let $\phi_A : E \to E_A$ be an isogeny whose kernel is $\langle [m_A]P_A + [n_A]Q_A \rangle$, and let $\phi_B : E \to E_B$ 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 randomly chosen 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 E_A, E_B and the points $\phi_A(P_B)$, $\phi_A(Q_B), \phi_B(P_A), \phi_B(Q_A)$, the *Supersingular Computational Diffie-Hellman (SSCDH)* problem is to find the j-invariant of $E/\langle [m_A]P_A + [n_A]Q_A, [m_B]P_B + [n_B]Q_B \rangle$.

Problem 3 Given a tuple sampled with probability 1/2 from one of the following two distributions:

$$(E_A, E_B, \phi_A(P_B), \phi_A(Q_B), \phi_B(P_A), \phi_B(Q_A), E_{AB}),$$

where $E_A, E_B, \phi_A(P_B), \phi_A(Q_B), \phi_B(P_A)$, $\phi_B(Q_A), E_{AB}$ are as in the SSCDH problem and $E_{AB} \cong E/\langle [m_A]P_A + [n_A]Q_A, [m_B]P_B + [n_B]Q_B \rangle$,

$$(E_A, E_B, \phi_A(P_B), \phi_A(Q_B), \phi_B(P_A), \phi_B(Q_A), E_C),$$

where $E_A, E_B, \phi_A(P_B), \phi_A(Q_B), \phi_B(P_A)$, $\phi_B(Q_A), E_{AB}$ are as in the SSCDH problem, m'_A, n'_A (respectively m'_B, n'_B) are randomly chosen 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), and $E_{AB} \cong E/\langle [m'_A]P_A + [n'_A]Q_A, [m'_B]P_B + [n'_B]Q_B \rangle$, the *Supersingular Decision Diffie-Hellman* problem (SSDDH) is to determine from which distribution the triple is sampled.