Complete Analysis of Implementing Isogeny-based Cryptography using Huff Form of Elliptic Curves

Suhri Kim

School of Mathematics, Statistics and Data Science, Sungshin Women's University, South Korea
suhrikim@gmail.com, suhrikim@sungshin.ac.kr

Abstract. In this paper, we present the analysis of Huff curves for implementing isogeny-based cryptography. In this regard, we first investigate the computational cost of the building blocks when compression functions are used for Huff curves. We also apply the square-root Vélu formula on Huff curves and present a new formula for recovering the coefficient of the curve, from a given point on a Huff curve. From our implementation, the performance of Huff-SIDH and Montgomery-SIDH is almost the same, and the performance of Huff-CSIDH is 6% faster than Montgomery-CSIDH. We further optimized Huff-CSIDH by exploiting Edwards curves for computing the coefficient of the image curve and present the Huff-Edwards hybrid model. As a result, the performance of Huff-Edwards CSIDH is almost the same as Montgomery-Edwards CSIDH. The result of our work shows that Huff curves can be quite practical for implementing isogeny-based cryptography but has some limitations.

Keywords: Isogeny, Post-quantum cryptography, Montgomery curves, square-root Vélu formula, Huff curves, SIDH, CSIDH

1 Introduction

As the development of a quantum computer that is capable of implementing Shor's algorithm becomes visible, researches are being actively conducted to find quantum-resistant algorithms that can substitute the currently used public-key cryptography. These are called post-quantum cryptography (PQC), and among PQC primitives, isogeny-based cryptography is known to have the smallest key sizes.

Quantum-resistant cryptography based on isogenies was first proposed by Couveignes [13] and later rediscovered by Stolbunov [25], which are currently called the CRS scheme. However, not only the quantum sub-exponential attack exists for the scheme [8], but the algorithm was also inefficient for practical use. After the introduction of the Supersingular Isogeny Diffie-Hellman (SIDH) by De Feo and Jao [18], the isogeny-based cryptography gains back its attention. Due to the non-commutative structure of the endomorphism ring of supersingular curves, SIDH resists the attack proposed in [8]. Additionally, instead of relying on the discrete logarithm problems where the intractability assumption of the problem is broken by Shor's algorithm, the security relies on the problem of finding an isogeny between two given isogenous elliptic curves over a finite field, which is known to have quantum-exponential complexity [**?**]. The Supersingular Isogeny Key Encapsulation (SIKE), a key encapsulation mechanism based on SIDH, was submitted as one of the candidates to the NIST PQC standardization project [1], and is currently an alternative candidate of Round 3.

Recently, CRS was revisited by De Feo, Kieffer, and Smith in [14], and independently by Castryck et al. in [7]. The advantage of CRS is that CCA-secure encryption can be constructed so that a non-interactive key exchange can be obtained. In [14], they modernized the parameter selection of CRS for better performance and presented an efficient way to compute the CRS group action. CRS was further optimized by Castryck et al. in [7]. In [7], they proposed CSIDH (Commutative SIDH), which solves the parameter selection problem in CRS by using supersingular elliptic curves defined over $\mathbb{F}_p$. Currently, the full key exchange of CSIDH at a 128-bit classical security level requires approximately 80ms, which is slower than SIDH. However, the vital aspect of CSIDH is that a relatively efficient digital signature can be constructed based on CSIDH [4]. CSI-FiSh [4] offers a practical digital signature that requires 390ms to sign a message. For isogeny-based cryptography, this was a significant result at that time, which facilitated the construction of various cryptographic primitives through elliptic curve isogenies. To summarize, SIDH and CSIDH have their own advantages, and their common disadvantage is that the performance is slower than other quantum-resistant algorithms.

The implementation of isogeny-based cryptography involves isogeny operations in addition to the standard elliptic curve arithmetic over a finite field. Regarding the isogeny operations, the degree of an isogeny used in the algorithm depends on the prime chosen for the scheme. The SIDH-based algorithms use the prime p of the form $p = \ell_A^{e_A}\ell_B^{e_B}f \pm 1$, where ℓ_A and ℓ_B are coprime to each other. The ℓ_A and ℓ_B corresponds to the degree of isogenies used in the algorithm. Since the complexity of computing isogenies increases as the degree increases, isogenies of degrees 3- and 4- were mostly considered for implementation. The CSIDH-based algorithms use the prime p of the form $p = 4\ell_1\ell_2\cdots\ell_n - 1$, where ℓ_i are odd-primes. Similarly, as ℓ_i are degrees of isogenies used in the scheme, demands for an efficient odd-degree isogeny formula have increased after the proposal of CSIDH. In [10], Costello and Hisil proposed an efficient way to compute arbitrary odd-degree isogenies on Montgomery curves. Classical ways for computing ℓ-isogeny requires $\tilde{O}(\ell)$ field operations. In [3], Bernstein et al. proposed the square-root Vélu formula which computes the ℓ-isogeny in $\tilde{O}(\sqrt{\ell})$ field operations. This ground-breaking work allows computing higher odd-degree isogenies efficiently, which suits well for implementing B-SIDH [9] and CSIDH, where the recent quantum analysis on CSIDH shows that larger odd-degree isogenies are required to provide sufficient security level [**?**, **?**, **?**]. Regarding the elliptic curve arithmetic, it is important to select the form of elliptic curves that can provide efficient curve operations. The majority of implementations in isogeny-based cryptography use Montgomery curves as it offers fast isogeny computation and elliptic curve arithmetic. The state-of-the-art implementation proposed in [11,12] is also based on Montgomery curves.

Currently, there is ongoing research on whether other forms of the elliptic curve can yield efficient arithmetic or isogeny computation. The primary candidate is twisted Edwards curves, as it is birationally equivalent to Montgomery curves, and mapping a point on one curve to a point on the other curve is costless when projective coordinates are used. The first use of Edwards curves was by Meyer et al. in [23], which used twisted Edwards curves for elliptic curve arithmetic and Montgomery curves for isogeny computation [23]. This was further optimized in [20], which used the Edwards curves for isogeny computation and Montgomery curves for the elliptic curve arithmetic. However, as stated in [5] and [20], using only Edwards curves for implementing SIDH-based algorithms is not as efficient as using only Montgomery curves.

The efficiency of using Edwards curves began to stand out when used for implementing CSIDH. Unlike SIDH-based algorithms, CSIDH-based algorithms use higher odd-degree isogenies. Montgomery curves offer efficient isogeny evaluation of arbitrary odd-degree isogenies [10]. However, it is hard to obtain an efficient formula for recovering the coefficient of the image curve on Montgomery curves. On the other hand, Edwards curves can provide an efficient formula for computing the coefficient of the image curve. Therefore, in [22], they implemented CSIDH by using Montgomery curves for isogeny evaluation and twisted Edwards curves for recovering the coefficient of the image curve. In [21], they proposed an optimized odd-degree isogeny formula by using the w-coordinate on Edwards curves. By adapting the formula in [21], Edwards-only CSIDH can be implemented, which is faster than Montgomery-CSIDH [7] or Hybrid-CSIDH [22]. The work of [21] shows that a certain form of an elliptic curve can lead to a better result for certain isogeny-based algorithms. Hence, it is important to check the implementation results on various elliptic curves.

1.1 Our Contributions

This work aims to provide an insight to exploit Huff curves in isogeny-based cryptography. Isogenies on Huff curves were first proposed in [24]. However, due to inefficient elliptic curve arithmetic and isogeny formula, it has not been studied until the work of [15], and in [17]. The proposed compression functions in [15, 17] for the points on a Huff curve allow Montgomery-like elliptic curve arithmetic formulas. Considering the fact that implementing SIDH entirely on Edwards curves is not faster than Montgomery curves as differential addition is slower in Edwards curves, it is very appealing that the compression functions in the Huff curve lead to an elliptic curve arithmetic formula having the same computational cost as on Montgomery curve.

In this paper, based on the formula presented in [15] and [17], we examine the applicability of Huff curves for isogeny-based cryptography. The following list details the main contributions of this work.

Analysis of the computational costs We examine the computational costs of the lower-level functions for implementing isogeny-based cryptography on Huff curves when a compression method in [15] and [17] are used. Details of the functions and their computational costs are presented in Section 3. Also, we present 4-isogeny on Huff curves using the compression method proposed in [15], by applying a similar method to derive the 4-isogeny formula presented in [17]. The formulas for 4-isogeny are presented in the Appendix.

Square-root Vélu formula on Huff curves We apply the square-root Vélu formula on Huff curves to compute higher-odd degree isogenies efficiently. To use the square-root Vélu formula proposed in [3], biquadratic polynomials must be redefined to express the relationship between points $P, Q, P-Q$, and $P+Q$ on a Huff curve. We derive biquadratic polynomials for Huff curves and demonstrate that the computational cost for evaluating the main polynomial h_S is the same as Montgomery curves. The definition of h_S and details of the formula is presented in Section 4.

The Huff-Edwards hybrid model for CSIDH We further optimized the odd-degree isogeny formula on Huff curves by exploiting Edwards curves for recovering the coefficient of the image curves in Huff form. The Montgomery-Edwards hybrid model has been studied extensively. We noticed that the cost of the conversion between the x-coordinate on Montgomery curves and w-coordinate on Huff curve is free so that Edwards curves can also be exploited to enhance the performance of Huff isogeny. The details are presented in Section 4.

Formula for recovering the curve coefficient of Huff curves We additionally present the formula to recover the coefficient of the Huff curve for SIDH-based cryptography. We analyze that using the compression method in [15] is faster for recovering the coefficient, so that such function is an efficient choice for implementing SIDH. For CSIDH-based cryptography, one has to examine whether supersingular Huff curves exist for a chosen prime. We deduce that for a prime $p \equiv 7 \mod 8$, there exist supersingular Huff curves over $\mathbb{F}_p$, and $\mathbb{F}_p$ has no supersingular Huff curves when $p \equiv 3 \mod 8$.

Implementation of isogeny-based cryptography on Huff curves We present the implementation result of SIDH and CSIDH using Huff curves. Based on our experiment, for SIDH, the performance of Montgomery-SIDH and Huff-SIDH is almost the same. For CSIDH, Huff-CSIDH is 6% faster than Montgomery-CSIDH. Also, the performance of Huff-Edwards CSIDH is almost the same as Montgomery-Edwards CSIDH. The details of the results are presented in Section 5.

1.2 Organization

This paper is organized as follows: In Section 2, we introduce two main isogeny-based key exchange algorithms and a form of Huff curves that will be used for the implementation. In Section 3, we demonstrate the computational cost of lower-level functions for implementing isogeny-based cryptography. In Section 4, we introduce the square-root Vélu formula and present the main polynomials for Huff curves to exploit the square-root Vélu formula. In Section 5, we present the implementation result of SIDH and CSIDH on Huff curves. We draw our conclusion in Section 6.

2 Preliminary

In this section, we provide the necessary background that will be used throughout the paper. First, we introduce two main streams in isogeny-based cryptography – SIDH and CSIDH. Lastly, we describe variants of Huff curves and their arithmetic.

2.1 Isogeny-based cryptography

We recall the SIDH and CSIDH key exchange protocol proposed in [18] and [7]. For more information, please refer to [18] and [7] for SIDH and CSIDH, respectively. The notations used in this section will continue to be used throughout the paper.

SIDH protocol Fix two coprime numbers ℓ_A and ℓ_B. Let p be a prime of the form $p = \ell_A^{e_A}\ell_B^{e_B}f \pm 1$ for some integer cofactor f, and e_A and e_B be positive integers such that $\ell_A^{e_A} \approx \ell_B^{e_B}$. Then construct a supersingular elliptic curve E over $\mathbb{F}_{p^2}$ of order $(\ell_A^{e_A}\ell_B^{e_B}f)^2$. We have full ℓ^e-torsion subgroup on E over $\mathbb{F}_{p^2}$ for $\ell \in \{\ell_A, \ell_B\}$ and $e \in \{e_A, e_B\}$. Choose basis $\{P_A, Q_A\}$ and $\{P_B, Q_B\}$ for the $\ell_A^{e_A}$- and $\ell_B^{e_B}$-torsion subgroups, respectively.

Suppose Alice and Bob want to exchange a secret key. Let $\{P_A, Q_A\}$ be the basis for Alice, and $\{P_B, Q_B\}$ be the basis for Bob. For key generation, Alice chooses random elements $m_A, n_A \in \mathbb{Z}/\ell_A^{e_A}\mathbb{Z}$, not both divisible by ℓ_A, and computes the subgroup $\langle R_A\rangle = \langle [m_A]P_A + [n_A]Q_A\rangle$. Then using Vélu's formula, Alice computes a curve $E_A = E/\langle R_A\rangle$ and an isogeny $\phi_A : E \to E_A$ of degree $\ell_A^{e_A}$, where $ker\phi_A = \langle R_A\rangle$. Alice computes and sends $(E_A, \phi_A(P_B), \phi_A(Q_B))$ to Bob. Bob repeats the same operation as Alice so that Alice receives $(E_B, \phi_B(P_A), \phi_B(Q_A))$.

For the key establishment, Alice computes the subgroup $\langle R'_A\rangle = \langle [m_A]\phi_B(P_A) + [n_A]\phi_B(Q_A)\rangle$. By using Vélu's formula, Alice computes a curve $E_{AB} = E_B/\langle R'_A\rangle$. Bob repeats the same operation as Alice and computes a curve $E_{BA} = E_A/\langle R'_B\rangle$. The shared secret between Alice and Bob is the j-invariant of E_{AB}, *i.e.* $j(E_{AB}) = j(E_{BA})$.

CSIDH protocol CSIDH uses commutative group action on supersingular elliptic curves defined over a finite field $\mathbb{F}_p$. Let $\mathcal{O}$ be an imaginary quadratic order. Let $\mathcal{E}\ell\ell_p(\mathcal{O})$ denote the set of elliptic curves defined over $\mathbb{F}_p$ with the endomorphism ring $\mathcal{O}$. It is well-known that the class group $Cl(\mathcal{O})$ acts freely and transitively on $\mathcal{E}\ell\ell_p(\mathcal{O})$. We call the group action as CM-action and denote the action of an ideal class $[\mathfrak{a}] \in Cl(\mathcal{O})$ on an elliptic curve $E \in \mathcal{E}\ell\ell_p(\mathcal{O})$ by $[\mathfrak{a}]E$.

Let $p = 4\ell_1\ell_2 \cdots \ell_n - 1$ be a prime where $\ell_1, \cdots, \ell_n$ are small distinct odd primes. Let E be a supersingular elliptic curve over $\mathbb{F}_p$ such that $\mathrm{End}_p(E) = \mathbb{Z}[\pi]$, where $\mathrm{End}_p(E)$ is the endomorphism ring of E over $\mathbb{F}_p$. Note that $\mathrm{End}_p(E)$ is a commutative subring of the quaternion order $\mathrm{End}(E)$. Then the trace of Frobenius is zero, hence $E(\mathbb{F}_p) = p + 1$. Since $\pi^2 - 1 = 0 \mod \ell_i$, the ideal $\ell_i\mathcal{O}$ splits as $\ell_i\mathcal{O} = \mathfrak{l}_i\bar{\mathfrak{l}}_i$, where $\mathfrak{l}_i = (\ell_i, \pi - 1)$ and $\bar{\mathfrak{l}_i} = (\ell_i, \pi + 1)$. The group action $[\mathfrak{l}_i]E$ (resp. $[\bar{\mathfrak{l}_i}]E$) is computed via isogeny $\phi_{\mathfrak{l}_i}$ (resp. $\phi_{\bar{\mathfrak{l}_i}}$) over $\mathbb{F}_p$ (resp. $\mathbb{F}_p$) using Vélu's formulas.

Suppose Alice and Bob want to exchange a secret key. Alice chooses a vector $(e_1, \cdots, e_n) \in \mathbb{Z}^n$, where $e_i \in [-m, m]$, for a positive integer m. The vector represents an isogeny associated to the group action by the ideal class $[\mathfrak{a}] = [\mathfrak{l}_1^{e_1} \cdots \mathfrak{l}_n^{e_n}]$, where $\mathfrak{l}_i = (\ell_i, \pi - 1)$. Alice computes the public key $E_A := [\mathfrak{a}]E$ and sends E_A to Bob. Bob repeats the similar operation with his secret ideal $\mathfrak{b}$ and sends the public key $E_B := [\mathfrak{b}]E$ to Alice. Upon receiving Bob's public key, Alice computes $[\mathfrak{a}]E_B$ and Bob computes $[\mathfrak{b}]E_A$. Due to the commutativity, $[\mathfrak{a}]E_B$ and $[\mathfrak{b}]E_A$ are isomorphic to each other so that they can derive a shared secret value from the elliptic curves.

2.2 Huff curves and their arithmetic

Huff curves Huff models for elliptic curves was first introduced by Joye, Tibouchi, and Vergnaud in [19]. They proposed the group law and formula for computing Tate pairings on Huff form of elliptic curves. Let K be a finite field of characteristic not equal to 2. The Huff form of elliptic curve is given by the equation:

$$H_{a,b} : ax(y^2 - 1) = by(x^2 - 1)$$

where $a^2 \neq b^2$ and $a, b \neq 0$. The point $O = (0, 0)$ is the neutral element and $-(x, y) = (-x, -y)$. Also, every Huff curve has three points at infinity, which are also points of order 2. The curve $H_{a,b}$ can also be simplified as

$$H_c : cx(y^2 - 1) = y(x^2 - 1)$$

where $c = a/b$, $c \neq \pm 1$. The general Huff curves which contains the Huff form of elliptic curves is introduced in [26]. General Huff curves are given by the equation

$$G_{a,b} : x(ay^2 - 1) = y(bx^2 - 1)$$

where $a \neq b$ and $a, b \neq 0$. Similar to the Huff curves, the point $O = (0, 0)$ is the neutral element and $-(x, y) = (-x, -y)$. The j-invariant of the curve $G_{a,b}$ is

$$j_{G_{a,b}} = \frac{2^8(a^2 - ab + b^2)^3}{a^2b^2(a - b)^2},$$

and the j-invariant of the curve $H_{a,b}$ is

$$j_{H_{a,b}} = \frac{2^8(a^4 - a^2b^2 + b^4)^3}{a^4b^4(a^2-b^2)^2}.$$

Isomorphisms The Huff curve $H_{a,b}$ is isomorphic to a Weierstrass curve of the form

$$W_{A,B} : y^2 = x^3 + Ax^2 + Bx$$

where $A = (a^2 + b^2)$ and $B = a^2b^2$. The Huff curve $H_{a,b}$ is isomorphic to an Edwards curve of the form

$$E_d : x^2 + y^2 = 1 + dx^2y^2$$

where $d = ((a-b)/(a+b))^2$, and corresponding Montgomery curve of the form

$$M_D : y^2 = x^3 + Dx^2 + x$$

where $D = (a^2 + b^2)/ab$.

Arithmetic on Huff curves For points $P = (x_p, y_p)$ and $Q = (x_q, y_q)$ on a Huff curve $H_{a,b}$, the addition of two points $P+Q = (x_r, y_r)$ is defined as below, and doubling can be performed with exactly the same formula.

$$x_r = \frac{(x_p + x_q)(1 + y_py_q)}{(1 + x_px_q)(1 - y_py_q)}$$
$$y_r = \frac{(y_p + y_q)(1 + x_px_q)}{(1 - x_px_q)(1 + y_py_q)}$$

The above formula is same for the curve H_c. For a general Huff curve $G_{a,b}$, the unified addition is performed as below:

$$x_r = \frac{(x_p + x_q)(ay_py_q + 1)}{(bx_px_q + 1)(ay_py_q - 1)}$$
$$y_r = \frac{(y_p + y_q)(bx_px_q + 1)}{(bx_px_q - 1)(ay_py_q + 1)}$$

where $P = (x_p, y_p)$ and $Q = (x_q, y_q)$ are points on $G_{a,b}$, and $P + Q = (x_r, y_r)$.

3 w-coordinates on Huff curves

Recently, in [15] and independently in [17], they proposed a compression function on Huff curves, which allows faster elliptic curve arithmetic and isogeny computation. We shall express this compression function as w-coordinate and examine the computational cost of formulas on Huff curves when w-coordinate is used. For the simplicity of the explanation, we shall denote w-function for the compression method proposed in [15], and w_{inv}-function for the compression method proposed in [17].

3.1 Compression function w on Huff curves

In [15], they proposed a compression method for Huff curves and presented an isogeny formula. As they proposed the formulas for elliptic curve arithmetic and isogenies for Huff curve of the form $H_{a,b}$, we shall present the formulas in this setting. However, for the implementation, we apply the compression function w on H_c, for simplicity. Note that this is equivalent to the case on $H_{a,b}$, with $b = 1$. For the exact formula on H_c using w, please refer to the Appendix.

For a point $P = (x, y)$ on a Huff curve $H_{a,b}$, define the compression function w as $w(P) = xy$. Then $w(P) = w(-P)$ and $w(O) = 0$. Using this function, doubling and differential addition can be expressed as follows [15].

For $P_1, P_2 \in H_{a,b}$, let $w_1 = w(P_1)$ and $w_2 = w(P_2)$. Let $w_0 = w(2P_1), w_3 = w(P_1 + P_2)$, and $w_4 = w(P_1 - P_2)$. Then,

$$w_0 = \frac{4w_1(w_1^2 + ew_1 + 1)}{(w_1^2 - 1)^2}, w_3 w_4 = \frac{(w_1 - w_2)^2}{(w_1 w_2 - 1)^2},$$

where $e = \frac{b}{a} + \frac{a}{b}$.

For the rest of this subsection, we examine the computational cost of the doubling, differential additions, and odd-degree isogeny formula on Huff curves, in the setting of isogeny-based cryptography. We consider WZ-coordinate as projective w-coordinate on Huff curves, where $w = W/Z$. The $\mathbf{M}$ and $\mathbf{S}$ refer to a field multiplication and squaring, respectively.

Doubling Let $P = (x, y)$ be a point on a Huff curve $H_{a,b}$. Let $a = A/D$, $b = B/D$, $w = xy$, and $w = W/Z$. For $w(P) = (W : Z)$ in projective w-coordinates, the doubling of P gives $w([2]P) = (W' : Z')$, where W' and Z' are defined as:

$$\begin{aligned} W' &= 4WZ(4AB(W - Z)^2 + (A + B)^2(4WZ)) \\ Z' &= 4AB((W - Z)^2(W + Z)^2) \end{aligned}$$

The computational cost is $4\mathbf{M}+2\mathbf{S}$, when we assume that $(A + B)^2$ and $4AB$ are precomputed.

Differential addition Let $P_1 = (W_1 : Z_1)$ and $P_2 = (W_2 : Z_2)$ be the points on $H_{a,b}$. Let $w_0 = w(P_1 - P_2)$ and $w_3 = w(P_1 + P_2)$. Let $w_0 = W_0/Z_0$ and $w_3 = W_3/Z_3$. Then,

$$\begin{aligned} W_3 &= Z_0(W_1 Z_2 - W_2 Z_1)^2, \\ Z_3 &= W_0(W_1 W_2 - Z_1 Z_2)^2. \end{aligned}$$

The computational cost of differential addition and doubling on Huff curves is $6\mathbf{M}+4\mathbf{S}$ using affine curve coefficients and $8\mathbf{M}+4\mathbf{S}$ using projective coordinates and projective curve coefficients.

Odd-degree isogeny formula In [15], they proposed the odd-degree isogeny formula on Huff curves by composing the isomorphism between Huff and general Huff curves, and odd-degree isogeny on general Huff curves.

Theorem 1 (Odd-degree isogeny on $H_{a,b}$ using w-function [15]). *Let P be a point on a Huff curve $H_{a,b}$ of odd order $\ell = 2s+1$. Let $\langle P \rangle = \{(0,0), \pm(\alpha_1, \beta_1), \cdots, \pm(\alpha_s, \beta_s)\}$, where $P = (\alpha_1, \beta_1)$. Let $w_i = \alpha_i \beta_i$ for $1 \leq i \leq s$. and $w = w(Q)$, where $Q = (x, y) \in H_{a,b}$. Then for ℓ-isogeny ϕ from $H_{a,b}$ to $H_{a',b'} = H_{a,b}/\langle P \rangle$ the evaluation of w, $w(\phi)$, is given by,*

$$w(\phi) = w \prod_{i=1}^{s} \frac{(w - w_i)^2}{(ww_i - 1)^2} \tag{1}$$

where

$$a' = \frac{a \prod_{i=1}^{s} (bw_i + a)}{\prod_{i=1}^{s} w_i (aw_i + b)} \quad \textit{and} \quad b' = \frac{b \prod_{i=1}^{s} (aw_i + b)}{\prod_{i=1}^{s} w_i (bw_i + a)}$$

Now to projectivize the formula, for $(\alpha_i, \beta_i) \in H_{a,b}$, let $(W_i : Z_i) = (w_i : 1)$ for $i = 1, ..., s$ where $w_i = \alpha_i \beta_i$. For an additional input point $(W : Z)$ on the curve $H_{a,b}$, the output is expressed as $(W' : Z')$ where $(W' : Z') = \phi(W : Z)$. Then, the equation (1) can be rewritten as:

$$W' = W \cdot \prod_{i=1}^{s} (WZ_i - ZW_i)^2,$$
$$Z' = Z \cdot \prod_{i=1}^{s} (WW_i - ZZ_i)^2,$$

where the computing $(WZ_i - ZW_i)$ and $(WW_i - ZZ_i)$ requires $2\mathbf{M}$. Hence for $\ell = 2s+1$-isogeny, evaluation of an isogeny costs $(4s)\mathbf{M}+2\mathbf{S}$. To compute the curve coefficients, let $a = A/D$ and $b = B/D$. Then we have,

$$A' = A \cdot \prod_{i=1}^{s} Z_i (BW_i + AZ_i)^2,$$
$$B' = B \cdot \prod_{i=1}^{s} Z_i (AW_i + BZ_i)^2,$$
$$D' = D \cdot \prod_{i=1}^{s} W_i (AW_i + BZ_i)(BW_i + AZ_i).$$

Note that only A' and B' are required when implementing isogeny-based cryptography on Huff curves. Moreover, since we use only the ratio of A and B, the term Z_i can be omitted when computing A' and B'. Therefore, recovering the curve coefficient costs $(4s)\mathbf{M}+2\mathbf{S}$.

Coefficient transformation Note that when w-function is used for elliptic curve arithmetic on Huff curves, instead of using the projective curve coefficients A, B, and D, we use $(A+B)^2$ and $4AB$ for efficient computation. Hence, after obtaining the coefficient of the image curve A' and B', $(A'+B')^2$ and $4A'B'$ must be computed in order to proceed with the elliptic curve arithmetic on the image curve. Intuitively, this requires $2\mathbf{S}$.

3.2 Compression function w_{inv} on Huff curves

Huang et al. proposed an alternate compression method for Huff curves and presented an isogeny formula on Huff curves [17]. For the simplicity of the formula, they used the Huff curve of the form H_c.

Let $P = (x, y)$ be a point on a Huff curve H_c. In [17], they defined the compression function w_{inv} as $w(P) = 1/xy$. Then $w(P) = w(-P)$ and $w(O) = \infty$. Using this function, doubling and differential addition formula are defined in [17]. Now, for $P_1, P_2 \in H_c$, let $w_1 = w(P_1)$ and $w_2 = w(P_2)$. Let $w_0 = w(2P_1), w_3 = w(P_1 + P_2)$, and $w_4 = w(P_1 - P_2)$. Then,

$$\begin{aligned} w_0 &= \frac{(w_1^2 - 1)^2}{4w_1(w_1 + c)(w_1 + 1/c)} \\ w_3 w_4 &= \frac{(w_1 w_2 - 1)^2}{(w_1 - w_2)^2}, \end{aligned}$$

For the rest of this subsection, we examine the computational cost of the doubling, differential additions, and odd-degree isogeny formula on Huff curves when w_{inv} is used for the compression.

Doubling Let $P = (x, y)$ be a point on a Huff curve H_c. Let $\hat{c} = \hat{C}/\hat{D}$, where $\hat{c} = \frac{1}{4}\left(c + \frac{1}{c} - 2\right)$. Let $w = 1/xy$, and $w = W/Z$. For $w(P) = (W : Z)$ in projective w-coordinates, the doubling of P gives $w([2]P) = (W' : Z')$, where W' and Z' are defined as:

$$\begin{aligned} W' &= \hat{D}(W - Z)^2 (W + Z)^2 \\ Z' &= 4WZ(\hat{D}(W + Z)^2 + \hat{C} \cdot 4WZ) \end{aligned}$$

The computational cost is $4\mathbf{M}$+$2\mathbf{S}$, given $\hat{C}$ and $\hat{D}$.

Differential addition Let $P_1 = (W_1 : Z_1)$ and $P_2 = (W_2 : Z_2)$ be the points on H_c. Let $w_0 = w(P_1 - P_2)$ and $w_3 = w(P_1 + P_2)$. Let $w_0 = W_0/Z_0$ and $w_3 = W_3/Z_3$. Then,

$$\begin{aligned} W_3 &= Z_0(W_1 W_2 - Z_1 Z_2)^2, \\ Z_3 &= W_0(W_1 Z_2 - Z_1 W_2)^2. \end{aligned}$$

The computational cost of differential addition and doubling on Huff curves is $6\mathbf{M}$+$4\mathbf{S}$ using affine curve coefficients and $8\mathbf{M}$+$4\mathbf{S}$ using projective coordinates and projective curve coefficients.

Odd-degree isogeny formula Using the compression function w_{inv} for the points on a Huff curve, the odd-degree isogeny formula is presented as the following theorem [17]:

Theorem 2 (Odd-degree isogeny on H_c using w_{inv}-function [17]). *Let P be a point on a Huff curve H_c of odd order $\ell = 2s+1$. Let $\langle P\rangle = \{(0,0), \pm(\alpha_1,\beta_1), \cdots, \pm(\alpha_s,\beta_s)\}$, where $P = (\alpha_1,\beta_1)$. Let $w_i = 1/\alpha_i\beta_i$ for $1 \leq i \leq s$. and $w = w(Q)$, where $Q = (x,y) \in H_c$. Then for ℓ-isogeny ϕ from H_c to $H_{c'} = H_c/\langle P\rangle$ the evaluation of w, $w(\phi)$, is given by,*

$$w(\phi) = w\prod_{i=1}^{s}\frac{(ww_i-1)^2}{(w-w_i)^2} \tag{2}$$

where

$$c' = c\prod_{i=1}^{s}\frac{(1+cw_i)^2}{(c+w_i)^2}$$

To transform the above formula by using projective coordinates and projective curve coeffcients, for $(\alpha_i,\beta_i) \in H_c$, let $(W_i : Z_i) = (w_i : 1)$ for $i = 1, ..., s$ where $w_i = 1/\alpha_i\beta_i$. For an additional input point $(W : Z)$ on the curve H_c, the output is expressed as $(W' : Z')$ where $(W' : Z') = \phi(W : Z)$. Then, the equation (2) can be rewritten as:

$$W' = W\cdot\prod_{i=1}^{s}(WW_i - ZZ_i)^2,$$
$$Z' = Z\cdot\prod_{i=1}^{s}(WZ_i - ZW_i)^2.$$

Similarly, for $\ell = 2s+1$-isogeny, evaluation of an isogeny using w_{inv}-function costs $(4s)$**M**+2**S**. To compute the curve coefficients, let $c = C/D$ and $c' = C'/D'$. Then we have,

$$C' = C\cdot\prod_{i=1}^{s}(DZ_i + CW_i)^2$$
$$D' = D\cdot\prod_{i=1}^{s}(CZ_i + DW_i)^2$$

Therefore, recovering the curve coefficient costs $(4s)$**M**+2**S**.

Coefficient transformation When w_{inv}-function is used for elliptic curve arithmetic on Huff curves, instead of using the projective curve coefficients C and D, we use $(C-D)^2$ and $4CD$ for efficient computation. Hence, after obtaining the coefficient of the image curve, C' and D', $(C'-D')^2$ and $4C'D'$ must be computed in order to proceed with elliptic curve arithmetic on the image curve. Intuitively, this requires 2**S**.

Remark 1. Note that when w-function is used for H_c, the formula for doubling, differential addition, and odd-degree isogenies is almost the reciprocal of the case when w_{inv}-function is used for H_c. Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be the points on H_c, and $w(P) = xy$ for $P = (x, y) \in H_c$. Let $w_1 = w(P_1)$, $w_2 = w(P_2)$, $w_0 = w(2P_1), w_3 = w(P_1 + P_2)$, and $w_4 = w(P_1 - P_2)$. Then,

$$w_0 = \frac{4w_1(w_1 + c)(w_1 + 1/c)}{(w_1^2 - 1)^2}$$
$$w_3 w_4 = \frac{(w_1 - w_2)^2}{(w_1 w_2 - 1)^2},$$

which is almost the reciprocal of the case when w_{inv}-function is used. Hence every computational cost for elliptic curve arithmetic and isogeny is the same when H_c with w is used and when H_c with w_{inv} is used.

3.3 Relationship with Montgomery curves

In this section, we shall examine the relationship between w-coordinate on a Huff curve and x-coordinate of a corresponding Montgomery curve. As denoted in Section 2, a Huff curve H_c is isomorphic to a Montgomery curve M_D for $D = (c^2 + 1)/c$, and Weierstrass curve $W_{A,B}$ for $A = c^2 + 1$ and $B = c^2$, given by the following maps:

$$\begin{aligned} M_D &\longrightarrow & W_{A,B} &\longrightarrow H_c \\ (x, y) &\longrightarrow & (cx, \sqrt{c^3}y) & \\ & & (x', y') &\longrightarrow \left(\frac{(x' + c^2)}{y'}, \frac{c(x' + 1)}{y'} \right) \end{aligned}$$

When XZ-only Montgomery arithmetic and WZ-only Huff arithmetic is used, switching between Huff curves and Montgomery curves is simple. A Montgomery point $(X_M : Z_M)$ can be transformed to the corresponding Huff WZ-coordinates as follows, when w_{inv}-function is used as a compression method :

$$(X_M : Z_M) \longrightarrow (W_{inv} : Z_{inv}) = (X_M : Z_M)$$

When w-function is used as a compression method, then the transformation is as follows:

$$(X_M : Z_M) \longrightarrow (W : Z) = (Z_M : X_M)$$

4 Square-root Vélu formula for Huff curves

4.1 Square-root Vélu formula

Recently, Bernstein et al. proposed an efficient algorithm that computes ℓ-isogeny in $\widetilde{O}(\sqrt{\ell})$ field operations [3]. The conventional Vélu formula computes ℓ-isogeny in $\tilde{O}(\ell)$ field operations. The high-level view of the Vélu formula can be considered as evaluation of polynomials over K whose roots are values of a function from a cyclic group to K. Let G be a cyclic group with generator P. Then for a finite subset S of $\mathbb{Z}$, define a polynomial

$$h_S(X) = \prod_{s \in S} (X - f([s]P)) \tag{3}$$

where $[s]P$ denotes the sum of s copies of P. In isogeny-based setting, let $E(K)$ be an elliptic curve, $P \in E(K)$. Then $G = \langle P \rangle$ is a kernel of an ℓ-isogeny $\phi : E \to E'$, and $f([s]P)$ can be considered as the x-coordinate of $[s]P$, for a scalar multiplication $[s]P$.

Let M_a be a Montgomery curve, $P \in M_a$ be a point of prime order $\ell \neq 2$. The isogeny $\phi : M_a \to M_{a'}$ with kernel $\langle P \rangle$ is given by the equation below, expressed in terms of equation (3):

$$\phi(X) = \frac{X^\ell \cdot h_S(1/X)^2}{h_S(X)^2}$$

where $a' = 2(1+d)/(1-d)$ for $d = ((a+2)/(a-2))^\ell \cdot (h_S(1)/h_S(-1))^8$, and $S = \{1, 3, \ldots, \ell - 2\}$. Now, $\phi(X)$ can be evaluated in $\tilde{O}(\sqrt{\ell})$ field operations if h_S is evaluated in $\tilde{O}(\sqrt{\ell})$ field operations.

The key for evaluating h_S in $\tilde{O}(\sqrt{\ell})$ field operations is to decompose the set S into smaller set I and J, having size similar to $\sqrt{S}$, satisfying certain conditions. In [3], I and J are chosen so that most of the elements in S is represented as elements of $(I + J) \cup (I - J)$. For details on the conditions of the set and algorithms, please refer to [3]. Hence, the problem of evaluating a polynomial whose roots are $[s]P$ for $s \in P$ is transformed to the problem of evaluating a polynomial, whose roots are $[i]P$ and $[j]P$ for $i \in I$ and $j \in J$, respectively. Then, by computing the resultant of polynomials relating to the set I and J, we can obtain the evaluation of h_S. To do this, we need to find the relations between the x-coordinate of $[i]P, [j]P, [i+j]P$, and $[i-j]P$ for $i \in I$ and $j \in J$. Below, Lemma 1 states the existence of biquadratic polynomials of an elliptic curve E that shows the relationship between points $P, Q, P+Q$, and $P-Q$ for $P, Q \in E$.

Lemma 1 (Biquadratic relations on x-coordinates [3]). *Let q be a prime power. Let $E(\mathbb{F}_q)$ be an elliptic curve. There exist biquadratic polynomials F_0, F_1, and F_2 in $\mathbb{F}_q[X_1, X_2]$ such that*

$$(X - x(P+Q))(X - x(P-Q)) = X^2 + \frac{F_1(x(P), x(Q))}{F_0(x(P), x(Q))} X + \frac{F_2(x(P), x(Q))}{F_0(x(P), x(Q))}$$

for all $P, Q \in E$ such that $O \notin \{P, Q, P+Q, P-Q\}$. The $x(P)$ denotes the x-coordinate of a point P.

If E is defined by affine Montgomery equation $By^2 = x^3 + Ax^2 + x$, then the polynomials F_0, F_1, and F_2 are defined as follows [3].

$$
\begin{aligned}
F_0(X_1, X_2) &= (X_1 - X_2)^2 \\
F_1(X_1, X_2) &= -2((X_1X_2 + 1)(X_1 + X_2) + 2AX_1X_2) \\
F_2(X_1, X_2) &= (X_1X_2 - 1)^2
\end{aligned}
$$

To use the square-root formula, we define the following biquadratic polynomials specifically for Huff curves of the form H_c. Similarly, the relationship between the w-coordinates of points $P, Q, P+Q$, and $P-Q$ on Huff curves can be written as follows:

$$
(W - w(P+Q))(W - w(P-Q)) = W^2 + \frac{G_1(w(P), w(Q))}{G_0(w(P), w(Q))}W + \frac{G_2(w(P), w(Q))}{G_0(w(P), w(Q))}
$$

For the curve H_c using the w-function, then the polynomials G_0, G_1, and G_2 are defined as follows:

$$
\begin{aligned}
G_0(W_1, W_2) &= (W_1W_2 - 1)^2 \\
G_1(W_1, W_2) &= -2((W_1W_2 + 1)(W_1 + W_2) + 2\bar{C}W_1W_2 + 4W_1W_2) \\
G_2(W_1, W_2) &= (W_1 - W_2)^2
\end{aligned}
$$

where $\bar{C} = c + \frac{1}{c} - 2$. When w_{inv}-function is used for compression, then the polynomials G_0, G_1, and G_2 are defined as follows:

$$
\begin{aligned}
G_0(W_1, W_2) &= (W_1 - W_2)^2 \\
G_1(W_1, W_2) &= -2((W_1W_2 + 1)(W_1 + W_2) + 2\bar{C}W_1W_2 + 4W_1W_2) \\
G_2(W_1, W_2) &= (W_1W_2 - 1)^2
\end{aligned}
$$

where $\bar{C} = c + \frac{1}{c} - 2$. Using this biquadratic polynomials, the square-root formula for Huff curves directly follows [3]. The following proposition states the isogeny formula on Huff curves H_c using w- and w_{inv}-function, expressed in terms of equation (3).

Proposition 1 (Square-root formula on Huff curves). *Let H_c be an elliptic curve over $\mathbb{F}_q$ in Huff form, and let P be a point of prime order $\ell \neq 2$ in H_c. Let w be a compression function for point on H_c. For $Q \in H_c$ let $w(Q) = W$. Then the evaluation of $w(\phi(Q))$ where $\phi : H_c \to H_{c'}$, a quotient isogeny with kernel $\langle P \rangle$, is given as:*

$$
w(\phi(W)) = \frac{W^\ell h_S(W)^2}{h_S(1/W)^2}
$$

where $S = \{1, 3, \ldots, \ell - 2\}$ and $c' = c^\ell \cdot h_S(-c)^2 / h_S(-1/c)^2$.

Now, let w_{inv} be a compression function for point on H_c. For $Q \in H_c$ let $w_{inv}(Q) = W$. Then the evaluation of $w_{inv}(\phi(Q))$ where $\phi : H_c \to H_{c'}$, a *quotient isogeny with kernel $\langle P\rangle$, is given as:*

$$w_{inv}(\phi(W)) = \frac{W^\ell h_S(1/W)^2}{h_S(W)^2}$$

where $S = \{1, 3, \ldots, \ell-2\}$ and $c' = c^\ell \cdot h_S(-1/c)^2/h_S(-c)^2$.

4.2 The Huff-Edwards hybrid model and computational costs

Note that computing the coefficient of the image of Montgomery curves using the square-root Velu formula exploits the relationship between Montgomery curves and Edwards curves [3]. This not only enhances the performance of computation but also the formula for computing the coefficient of the image curve follows the expression of the square-root Vélu formula for evaluating an isogeny. Hence, Montgomery curves only need to compute $h_S(1), h_S(-1)$, while Huff curves need to compute $h_S(-c), h_S(-1/c)$ for $c \in \mathbb{F}_p$. However, as stated in Section 3, the cost of the transformation between Montgomery curves and Huff curves is free so that a similar idea can also be applied to Huff curves. The computation of the coefficient of the image curve in Huff form can further be optimized using the corresponding Edwards curves.

Let P be a point of prime order $\ell \neq 2$ in a Huff curve H_c. Let w_{inv} be a compression function for point on H_c, and $\hat{c} = \frac{1}{4}\left(c + \frac{1}{c} - 2\right)$, and $\phi : H_c \to H_{c'}$, a quotient isogeny with kernel $\langle P\rangle$. Then $\hat{c}' = \frac{1}{4}\left(c' + \frac{1}{c'} - 2\right)$ can be computed as $\hat{c}' = d/(1-d)$, where

$$d = \left(\frac{\hat{c}+1}{\hat{c}}\right)^\ell \left(\frac{h_S(1)}{h_S(-1)}\right)^8.$$

Note that when Edwards curves are exploited for recovering the coefficient of the image curve in the Huff-Edwards hybrid model, the performance gain not only comes from computing $h_S(1), h_S(-1)$ instead of $h_S(-c), h_S(-1/c)$, but also from the fact that no more coefficient transformation is required. The isogeny formula on Huff curves computes c' of the image curve $H_{c'}$, expressed in terms of c of the domain curve H_c. Therefore, coefficient transformation is required afterwards as $\hat{c}' = \frac{1}{4}\left(c' + \frac{1}{c'} - 2\right)$ is used for elliptic curve arithmetic, and c' must be kept to proceed with isogeny computations. On the other hand, when Edwards curves are exploited to recover the coefficient of the Huff curve, the formula directly uses $\hat{c}$ and $\hat{c}'$ so that coefficient transformation is not required nor c is kept to proceed with isogeny computation.

Lastly, We compare the computational costs of biquadratic polynomials for Montgomery curves and Huff curves. Computing ℓ-isogeny consists of isogeny evaluation and recovering the coefficient of the image curve. In Table 1, ℓ `eval` refers to isogeny evaluation and ℓ `coeff` refers to computing the coefficient of the image curve. The Huff-Edwards in Table 1 refers to the case when Huff curves are used to evaluate isogeny, and Edwards curves are used to compute the coefficients.

Table 1: Computational costs of biquadratic polynomials

		Montgomery	Huff	Huff-Edwards
ℓ `eval`	Computation	$h_S(Q), h_S(1/Q)$	$h_S(Q), h_S(-1/Q)$	$h_S(Q), h_S(-1/Q)$
	Cost	**7M+2S**	**7M+3S**	**7M+2S**
ℓ `coeff`	Computation	$h_S(1), h_S(-1)$	$h_S(-c), h_S(-1/c)$	$h_S(1), h_S(-1)$
	Cost	**3M+2S**	**7M+3S**	**3M+2S**

4.3 Recovering the curve coefficient

From the biquadratic polynomials, we were able to derive the formula for recovering the coefficient of the Huff curve from the w-coordinate of the points P, Q and $P - Q$ on a Huff curve, which was the missing formula in [15] and in [17].

When implementing SIDH-based cryptography, $P_A - Q_A$ and $P_B - Q_B$ are also considered as a public key for faster kernel computation using the Montgomery ladder. Hence $\phi_A(P_B - Q_B)$ and $\phi_B(P_A - Q_A)$ are also computed and exchanged to compute the shared secret key efficiently. This can be thought of as an increase in the public key size. But using the fact that the coefficient a of the Montgomery curve M_a relates to the x-coordinates of P, Q, and $P - Q$ for $P, Q \in M_a$, sending the coefficient of the image curve is omitted [12]. Therefore, $(\phi_A(P_B), \phi_A(Q_B), \phi_A(P_B - Q_B))$ and $(\phi_B(P_A), \phi_B(Q_A), \phi_B(P_A - Q_B))$ are exchanged during the protocol, and upon the receipt of the public key, the coefficient is recovered using the relationship, which costs $4\mathbf{M}+1\mathbf{S}+1\mathbf{I}$. The $\mathbf{I}$ denotes the field inversion.

For Huff curves, similar relationship can be obtained. Let H_c be a Huff curve using w as a compression function. For P, Q, and $P - Q$ in H_c, let $w(P) = w_p, w(Q) = w_q$, and $w(P - Q) = w_{pq}$. Then the following holds:

$$\begin{aligned} w(P+Q) + w(P-Q) &= \frac{2((w_p w_q + 1)(w_p + w_q) + 2\bar{c} w_p w_q + 4 w_p w_q)}{(w_p w_q - 1)^2} \\ \frac{(w_p - w_q)^2}{w_{pq}(w_p w_q - 1)^2} + w_{pq} &= \frac{2((w_p w_q + 1)(w_p + w_q) + 2\bar{c} w_p w_q + 4 w_p w_q)}{(w_p w_q - 1)^2} \end{aligned}$$

so that

$$\begin{aligned} \bar{c} &= \frac{(w_p - w_q)^2 + w_{pq}^2 (w_p w_q - 1)^2 - 2 w_{pq}((w_p w_q + 1)(w_p + w_q) + 4 w_p w_q)}{4 w_{pq} w_p w_q} \\ &= \frac{((w_p - w_q) - (w_{pq}(w_p w_q - 1)))^2 - 4 w_{pq}(w_p + w_p w_q^2 + 2 w_p w_q)}{4 w_{pq} w_p w_q} \end{aligned} \tag{4}$$

where $\bar{c} = c + \frac{1}{c} - 2$. The computational cost is 3**M**+1**S**+1**I**. Similar relationship can be obtain for w_{inv}-function, which is as follows.

$$w_{inv}(P+Q) + w_{inv}(P-Q) = \frac{2((w_p w_q + 1)(w_p + w_q) + 2\bar{c} w_p w_q + 4 w_p w_q)}{(w_p - w_q)^2}$$
$$\frac{(w_p w_q - 1)^2}{w_{pq}(w_p - w_q)^2} + w_{pq} = \frac{2((w_p w_q + 1)(w_p + w_q) + 2\bar{c} w_p w_q + 4 w_p w_q)}{(w_p - w_q)^2}$$

so that

$$\begin{aligned}\bar{c} &= \frac{(w_p w_q - 1)^2 + w_{pq}^2 (w_p - w_q)^2 - (2w_{pq}((w_p w_q + 1)(w_p + w_q) + 4 w_p w_q)}{4 w_{pq} w_p w_q} \\ &= \frac{((w_p w_q - 1) - (w_{pq}(w_p - w_q)))^2 - 4 w_{pq}(w_p + w_p w_q^2 + 2 w_p w_q)}{4 w_{pq} w_p w_q} \qquad (5)\end{aligned}$$

where $\bar{c} = c + \frac{1}{c} - 2$. The computational cost is 5**M**+1**S**+1**I**.

Summarizing the section, Table 2 denotes the computational cost of the building blocks of isogeny-based cryptography on Montgomery curves and on Huff curves. The middle rule in Table 2 divides the functions into two groups – the upper half is the functions that are commonly used in SIDH and CSIDH-based cryptography, and the lower half is the functions that are explicitly used in SIDH-based cryptography.

In Table 2, `DBLADD` refers to the differential addition and doubling in projective coordinates, and `DBL` refers to the doubling. ℓ`-isog eval` refers to the evaluation of an ℓ-isogeny and ℓ`-isog coeff` refers to the computation of the coefficient of the ℓ-isogenous image curve, where $\ell = 2s + 1$. `CoeffTrans` refers to the cost of transforming the coefficient for efficient elliptic curve arithmetic, which only occurs on Huff curves. The `TPL` refers to tripling of a point, and `3-isogeny` and `4-isogeny` are the combined computational cost of isogeny evaluation and coefficient computation. Lastly, `get_coeff` refers to recovering of the curve coefficient using points P, Q and $P - Q$ on an elliptic curve.

Also, `Mont` refers to Montgomery curve, and `Mont-Edwards Hybrid` refers to the hybrid method proposed in [22], where Montgomery curves are used for elliptic curve arithmetic and isogeny evaluation, and Edwards curves are used for computing the coefficient of the image curve. `Huff-Edwards Hybrid` refers to the hybrid method where Huff curves are used for elliptic curve arithmetic and isogeny evaluation, and Edwards curves are used for computing the coefficient of the image curve. As the hybrid methods are used for implementing CSIDH-based cryptography, the computational cost of the lower half of the table is omitted. The function $w(\ell)$ refers to $w(\ell) = (h-1)\mathbf{M} + (t-1)\mathbf{S}$. In $w(\ell)$, h denotes the hamming weight of ℓ and t is the bit length of ℓ.

As shown in Table 2, except for the ℓ`-isogeny coeff`, the computational cost of the lower-level functions is the same for Montgomery curves and Huff curves. Also, as the compression function w and w_{inv} are reciprocals of each other, the formula of the lower-level functions are almost reciprocals of each other so that w-function and w_{inv}-function induce the same computational cost. Hence, when implementing CSIDH-based cryptography on Huff curves, the compression function is free of one's choice. On the other hand, for SIDH-based cryptography, w-function is preferred as `get_coeff` is slightly efficient than w_{inv}-function.

Table 2: Computational cost of building-blocks of isogeny-based cryptography on Huff curves and Montgomery curves

	`Mont` [7, 10]	`Mont-Edwards Hybrid` [22]	w	w_{inv}	`Huff-Edwards Hybrid`
`DBLADD`	$6\mathbf{M} + 4\mathbf{S}$	$6\mathbf{M} + 4\mathbf{S}$	$6\mathbf{M} + 4\mathbf{S}$	$6\mathbf{M} + 4\mathbf{S}$	$6\mathbf{M} + 4\mathbf{S}$
`DBL`	$4\mathbf{M} + 2\mathbf{S}$	$4\mathbf{M} + 2\mathbf{S}$	$4\mathbf{M} + 2\mathbf{S}$	$4\mathbf{M} + 2\mathbf{S}$	$4\mathbf{M} + 2\mathbf{S}$
ℓ-`isog eval`	$4s\mathbf{M} + 2\mathbf{S}$	$4s\mathbf{M} + 2\mathbf{S}$	$4s\mathbf{M} + 2\mathbf{S}$	$4s\mathbf{M} + 2\mathbf{S}$	$4s\mathbf{M} + 2\mathbf{S}$
ℓ-`isog coeff`	$(6s-2)\mathbf{M} + 3\mathbf{S}$	$(2s)\mathbf{M} + 6\mathbf{S} + 2w(\ell)$	$4s\mathbf{M} + 2\mathbf{S}$	$4s\mathbf{M} + 2\mathbf{S}$	$(2s)\mathbf{M} + 6\mathbf{S} + 2w(\ell)$
`CoeffTrans`	-	-	$2\mathbf{S}$	$2\mathbf{S}$	-
`TPL`	$7\mathbf{M} + 5\mathbf{S}$	-	$7\mathbf{M} + 5\mathbf{S}$	$7\mathbf{M} + 5\mathbf{S}$	-
`3-isogeny`	$6\mathbf{M} + 5\mathbf{S}$	-	$6\mathbf{M} + 5\mathbf{S}$	$6\mathbf{M} + 5\mathbf{S}$	-
`4-isogeny`	$6\mathbf{M} + 6\mathbf{S}$	-	$6\mathbf{M} + 6\mathbf{S}$	$6\mathbf{M} + 6\mathbf{S}$	-
j-invariant	$3\mathbf{M} + 4\mathbf{S} + 1\mathbf{I}$	-	$3\mathbf{M} + 4\mathbf{S} + 1\mathbf{I}$	$3\mathbf{M} + 4\mathbf{S} + 1\mathbf{I}$	-
`get_coeff`	$4\mathbf{M} + 1\mathbf{S} + 1\mathbf{I}$	-	$3\mathbf{M} + 1\mathbf{S} + 1\mathbf{I}$	$5\mathbf{M} + 1\mathbf{S} + 1\mathbf{I}$	-

Remark 2. On Huff curves, the `CoeffTrans` can be omitted when division polynomial is used to represent the curve coefficient in terms of the kernel points. This can be easily done for 3- and 4- isogenies. For general higher degree isogenies, as representing the coefficient of the curve using kernel point is difficult, extra `CoeffTrans` operation is required to proceed with the elliptic curve arithmetic further or Huff-Edwards hybrid can be used to omit the transformations.

5 Implementation

In this section, we provide the performance result of isogeny-based cryptography. To evaluate the performance, the algorithms are implemented in the C language. For implementing SIDH, we use the field arithmetic of Round 3 version of SIKE, submitted to NIST. For implementing CSIDH, we use the field arithmetic implemented in [7]. All the cycle counts were obtained on one core of an Intel Core i7-7700 at 3.60 GHz, running Ubuntu 16.04 LTS. For the compilation, we used gcc version 9.3.0 with an optimization level -O3.

5.1 Implementation of SIDH

We first present the parameter settings for SIDH implementation. Then we present the implementation result with analysis. For implementing SIDH, we used the Huff curve of the form H_c with w as compression function, as w-function is more efficient than w_{inv} for recovering the coefficient of the curve after the first round of the protocol.

Parameter Settings The prime used in SIDH-based cryptography is of the form $p = \ell_A^{e_A}\ell_B^{e_B}f \pm 1$. In this section, we present two implementations on Huff curves when $\{\ell_A, \ell_B\} = \{2, 3\}$ and $\{\ell_A, \ell_B\} = \{3, 5\}$. The former is the general choice of ℓ_A and ℓ_B for implementing SIDH-based cryptography. As an extra coefficient transformation is required for Huff curves for higher degree isogenies, the latter is to examine the performance change caused by this.

For $\{\ell_A, \ell_B\} = \{2, 3\}$, we used the 751-bit prime proposed in [2], which is as follows:

$$p_{751} = 2^{372} \cdot 3^{239} - 1$$

For $\{\ell_A, \ell_B\} = \{3, 5\}$, we used the 621-bit prime of the form:

$$p_{621} = 2^{67} \cdot 3^{175} \cdot 5^{119} - 1,$$

and $3^{175} \approx 2^{277.368}$ and $5^{119} \approx 2^{276.309}$.

Over finite field $\mathbb{F}_{p_a^2} = \mathbb{F}_{p_a}(i)$ for $i^2 = -1$ and $a \in \{751, 621\}$, we used the supersingular Montgomery curve of the form as the base curve:

$$M : y^2 = x^3 + 6x^2 + x,$$

which is isomorphic to a Huff curve of the form:

$$H_{c_a} : c_a x(y^2 - 1) = y(x^2 - 1).$$

For $a = 751$, then $c_{751} = 3 + \sqrt{8} \in \mathbb{F}_{p_{751}^2}$ and for $a = 621$, then $c_{621} = 3 + \sqrt{8} \in \mathbb{F}_{p_{621}^2}$.

For p_{621}, the generator points for the Huff curve are the points P_A, Q_A and P_B, Q_B such that $P_A, Q_A \in E[3^{175}]$ and both points have exact order 3^{175}, $P_B, Q_B \in E[5^{119}]$ and both points have an exact order 5^{119}. To select such a point, we first search for the points on the following Weierstrass curve:

$$W : y^2 = x^3 + (c_{621}^2 + 1)x^2 + c_{621}^2 x,$$

which is isomorphic to the Huff curve $H_{c_{621}}$. When (P_A, Q_A) and (P_B, Q_B) are found, we compute the Weil paring $e(P_A, Q_A) \in E[3^{175}]$ and $e(P_B, Q_B) \in E[5^{119}]$ to check that the result has order 3^{175} and 5^{119}, respectively. When the points are found, we transform the points on W to points on $H_{c_{621}}$, and express in w-coordinate. The generator points on Montgomery curves are found in a similar manner.

Also, when implementing 5-isogeny, we used the formula from [10] for isogeny evaluation. For recovering the coefficient of the image curve, we used the 2-torsion method described in [10]. The reason is that using the 2-torsion method, the cost for recovering the coefficient of the image curve is 8**M**+4**S**, while using the projectivized formula of [10] presented in [7], the cost is 10**M**+3**S**.

Implementation Results Table 3 presents the implementation results of SIDH on Montgomery curves and Huff curves. Using the prime p_{621} and p_{751}, the performance of SIDH is compared between Montgomery curves and Huff curves. The implementation using p_{621} uses 3- and 5-isogeny formula, which is presented in Section 3.2. The prime p_{751} uses 3- and 4-isogeny, and the corresponding formula on H_c using w-function is in the Appendix.

Table 3: Performance results of SIDH implementation.

	Montgomery Curve		Huff Curve	
	p_{621}	p_{751}	p_{621}	p_{751}
Isogeny degree used	3,5	2,3	3,5	2,3
Alice's Keygen	144,248,840	243,489,372	144,182,511	243,590,780
Bob's Keygen	160,539,980	273,583,953	160,668,976	273,541,121
Alice's Shared Key	122,129,993	200,242,066	122,118,345	200,231,811
Bob's Shared Key	138,590,523	233,094,860	138,760,218	232,956,506
Total	565,509,336	950,410,251	565,730,050	950,320,218
Security (Classical)	138	186	138	186

As denoted in Table 3, for p_{751}, the performance of the Montgomery-SIDH and Huff-SIDH are almost the same. This is obvious as the computational cost for the formulas for implementing isogeny-based cryptography is almost the same. For p_{621}, although the Huff curve requires to transform the curve coefficient on Bob's side, the performance of the Montgomery-SIDH and Huff-SIDH is almost the same. We shall analyze the results in detail by dividing them into key generation and shared key computation phases.

Public key generation During this phase, it is natural that there is no difference when comparing the computational cost of the two curves for Alice's side. For Bob's side on Huff curves, after calculating the coefficient of the image curve, extra coefficient transformation is required for efficient quintupling. Hence, computing the coefficient on Huff curves costs $8\mathbf{M}+4\mathbf{S}$ for a total. For Montgomery curves, 2-torsion is used for recovering the coefficient of the image curve. Hence, after evaluating isogeny at a 2-torsion point for a Montgomery curve, recovering the curve coefficient is required, and total also costs $8\mathbf{M}+4\mathbf{S}$. Therefore, the computation of 5-isogeny on both curves is almost the same.

Computing the shared key The difference in the computation between two curves occurs when calculating the curve coefficient upon the receipt of $(\phi_i(P_j), \phi_i(Q_j), \phi_i(P_j - Q_j))$ for $(i, j) \in \{(A, B), (B, A)\}$. Now, note that upon the receipt of $(\phi_i(P_j), \phi_i(Q_j), \phi_i(P_j - Q_j))$ for $(i, j) \in \{(A, B), (B, A)\}$, the Huff coefficient $\hat{c} = c + 1/c + 2$ of H_c is recovered using equation (4), not c itself. For

Alice, as $\hat{c}$ is directly used for tripling and isogeny computation, the performance on Huff curves and Montgomery curves is almost the same. However, on Bob's side in Huff curves, recovering $\hat{c}$ is not enough $-$ $\hat{c}$ is used for quintupling, but we need the actual c to compute the coefficient of the isogenous curve. For Montgomery curves, Bob uses the extra 2-torsion point on the base curve to compute the image curve's coefficient. To reduce the key size, when computing the shared key on Bob's side, we compute the 2-torsion, given the coefficient of the Montgomery curve. Hence both curves require solving quadratic equation over $\mathbb{F}_{p^2}$, which requires 1 field squaring and 1 square-root computation for both curves. Hence, the performance of Montgomery-SIDH and Huff-SIDH is almost the same.

5.2 Implementation of CSIDH

For CSIDH-based cryptography, the compression function to use is free of one's choice as the computational cost of the building blocks for CSIDH is the same for w and w_{inv}. In this paper, we used w_{inv} for implementing CSIDH. To implement CSIDH-based cryptography, we first need to check whether a supersingular curve exists over a given prime field. In this section, we examine the existence of a supersingular Huff curve over $\mathbb{F}_p$ for a prime p and present the base curve H_c for the implementation. Then we present the implementation result of CSIDH using Huff curves.

Prime field and base curve In order to implement the CSIDH-based cryptography, we need to search for a supersingular Huff curve $H_{a,b}$ over a prime field p of the form $p = f \cdot \prod \ell_i - 1$, where ℓ_is are small distinct primes. Below is the theorem proving that a supersingular Huff curve over $\mathbb{F}_p$ exists when $p \equiv 7 \mod 8$. If $p \equiv 3 \mod 8$, there is no supersingular Huff curve over $\mathbb{F}_p$.

Theorem 3. *There exists a supersingular Huff curve of the form $H_{a,b}$ over $\mathbb{F}_p$ when $p \equiv 7 \mod 8$.*

Proof. In the CSIDH setting, for every supersingular elliptic curve over $\mathbb{F}_p$, there exists a corresponding supersingular Montgomery curve over $\mathbb{F}_p$. Hence it suffices to show that for a given supersingular Montgomery curve, there exists an isomorphic Huff curve over $\mathbb{F}_p$. Now, Huff curve $H_{a,b}$ is isomorphic to a Montgomery curve of the form:

$$M : y^2 = x^3 + \frac{a^2 + b^2}{ab} x^2 + x \tag{6}$$

Then, M is supersingular if and only if $H_{a,b}$ is supersingular. Let $(a^2 + b^2)/ab = A$. If we find a supersingular Montgomery curve $y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$, then by using the equation (6), we can find the corresponding supersingular Huff curve over $\mathbb{F}_p$. Solving the equation we have,

$$a = \frac{Ab \pm \sqrt{(Ab)^2 - 4b^2}}{2} \tag{7}$$

From the above equation, $H_{a,b}$ is defined over $\mathbb{F}_p$, if and only if $Ab^2 - 4b^2$ is a square in $\mathbb{F}_p$, i.e. $A^2 - 4$ is a square in $\mathbb{F}_p$.

Now, suppose $p \equiv 7 \mod 8$ and let M be a supersingluar curve having a 2-torsion point on $\mathbb{F}_p$ except for $(0,0)$. Then the 2-torsion subgroup of M satisfy $|M[2]| = 4$. In this case, the supersingular curve M lies on the surface so that $\mathrm{End}_{\mathbb{F}_p}(M) \cong \mathbb{Z}[(1+\sqrt{-p})/2]$ [6]. Then $A^2 - 4$ is a square in $\mathbb{F}_p$, so that the corresponding $H_{a,b}$ exists over $\mathbb{F}_p$. On the other hand, if $p \equiv 3 \mod 8$, then M lies on the floor so that $A^2 - 4$ is not a square in $\mathbb{F}_p$, so that there is no supersingular Huff curve $H_{a,b}$ over $\mathbb{F}_p$.

The original implementation of CSIDH uses the prime of the form $p \equiv 3 \mod 8$. However, from Theorem 3. we use the 511-bit prime presented in [16], which works over $\mathbb{F}_p$ where

$$p = 2^4 \cdot 3^3 \cdot 5 \cdot 7 \cdot 11^2 \cdot 13 \cdots 373 - 1. \tag{8}$$

In this field, we choose a supersingular Huff curve of the form as the base curve:

$$H_c : cx(y^2 - 1) = y(x^2 - 1)$$

where $c = 3 - \sqrt{8} \in \mathbb{F}_p$.

Remark 3. For a prime p such that $p \equiv 3 \mod 8$, there exist a supersingular general Huff curve over $\mathbb{F}_p$. However, as the computational cost of elliptic curve arithmetic and isogeny evaluations is slower than the Huff curve, we omit this case.

Selecting a random point over $\mathbb{F}_p$ When implementing CSIDH, one has to select a random point on a curve over $\mathbb{F}_p$ of a certain order to compute an isogeny using Vélu's formula. For a Montgomery curve, first, a random element in $\mathbb{F}_p$ is selected, and we consider it as an x-coordinate of a given Montgomery curve. Then, by using the curve equation $y^2 = x^3 + Ax^2 + x$, $r = x^3 + Ax^2 + x$ is computed and checked whether r is a square or non-square in $\mathbb{F}_p$. The computational cost for checking whether a random point on a Montgomery curve is in $\mathbb{F}_p$ or $\mathbb{F}_{p^2} \backslash \mathbb{F}_p$ costs $2\mathbf{M} + 1\mathbf{S}$ (we omit the computational costs for computing the Legendre symbol).

The following method checks whether the point $(x, y) \in H_c$ is on $\mathbb{F}_p$ or $\mathbb{F}_{p^2} \backslash \mathbb{F}_p$. Since $w = 1/xy$ for a point $(x, y) \in H_c$, $y = 1/wx$. Now, from the curve equation, the following holds:

$$\begin{aligned} cx(y^2 - 1) &= y(x^2 - 1) \\ \frac{c}{w}y - cx &= \frac{c}{w}x - y \\ \left(\frac{c}{w} + 1\right) y &= \left(\frac{1}{w} + c\right) x \\ x^2 &= (w + c)/w(1 + cw) \end{aligned}$$

Thus $x \in \mathbb{F}_p$ if $(cw + 1)(cw + w^2)$ is square in $\mathbb{F}_p$. The computational cost for checking whether a random point is in $\mathbb{F}_p$ or $\mathbb{F}_{p^2} \backslash \mathbb{F}_p$ costs $2\mathbf{M} + 1\mathbf{S}$.

Implementation results For the implementation, we used the prime field $\mathbb{F}_p$, where p is defined as in equation 8. In order to compare the performance with Montgomery curves, we use the following supersingular curve over $\mathbb{F}_p$ as a base curve.

$$M : y^2 = x^3 + x$$

The original implementation of Montgomery-CSIDH in [7] does not use the optimization method when evaluating isogenies. Hence, we modified the implementation for a fair comparison with the Huff-CSIDH. The difference in the performance between the algorithms lies purely in the computation of the coefficient of the image curve and coefficient transformation for Huff curves. Table 4 presents the performance of the group action on Montgomery curves and Huff curves. In Table 4, Montgomery-Edwards CSIDH is a method proposed in [22], which implements CSIDH using Montgomery curves but uses Edwards curves for evaluating the coefficient of the image curve. Lastly, Huff-Edwards CSIDH is a method that implements CSIDH using Huff curves, but uses Edwards curves for evaluating the coefficient of the image curve.

Table 4: Performance results of group action in using traditional Vélu formula.

	Group action
Montgomery-CSIDH	109,772,163
Montgomery-Edwards CSIDH	96,570,560
Huff-CSIDH	103,079,951
Huff-Edwards CSIDH	96,766,674

As shown in Table 4, Huff-CSIDH is 6% faster than Montgomery-CSIDH. This is because although an extra coefficient transformation is required when Huff curves are used for the implementation, recovering the Montgomery curve's coefficient is costly than on a Huff curve for odd-degree isogenies. For the hybrid implementations, Montgomery-Edwards CSIDH is almost the same as Huff-Edwards CSIDH.

Additionally, for CSIDH, it is important to optimize the odd-degree isogeny formula as isogeny computation contributes to the overall CSIDH performance. Hence we present the CSIDH implementation using the square-root Vélu formula in [3].

Table 5: Performance results of group action in CSIDH using the square-root Vélu formula.

	Group action
`sqrt-Mont`	97,549,802
`sqrt-Huff`	97,957,553

Table 5 presents the performance comparison of Montgomery-CSIDH and Huff-CSIDH, when the square-root Vélu formula is used for computing odd-degree isogenies. In Table 5, `sqrt-Mont` and `sqrt-Huff` refers to CSIDH implementation when the square-root Vélu formula is used for Montgomery curves and Huff curves, respectively. As the effect of the square-root Vélu formula becomes more conspicuous when isogeny of degree larger than 113 is used, the effect is not immediate for the current parameter setting. However, as the square-root Vélu formula exploits Edwards curves for recovering the coefficient of Montgomery curve, we can see that the performance of `sqrt-Mont` is similar to Montgomery-Edwards CSIDH in Table 4. As this is also the case for Huff curves, where Edwards curves are used to recover the coefficients of the image curve, the performance of `sqrt-Huff` is similar to Huff-Edwards CSIDH.

6 Conclusion

In this paper, we present the analysis of Huff curves' usage for implementing isogeny-based cryptography. First, we analyzed the computational cost of the lower-level functions when the compression method is used for Huff curves. Then, we proposed additional functions on Huff curves to implement isogeny-based cryptography. We presented the implementation results of SIDH and CSIDH on Huff curves.

For SIDH, we conclude that using w as a compression function on Huff curves is preferred as the computational cost of recovering the coefficient is more efficient. As the computational cost for the lower-level function is the same on Montgomery curves and Huff curves, the performance of Montgomery-SIDH and Huff-SIDH is almost the same. For CSIDH, we present the birational polynomials on Huff curves in order to exploit the square-root Vélu formula. We implemented CSIDH using the classical Vélu formula and the square-root Vélu formula and compare them with Montgomery curves. The performance of `sqrt-Mont` is almost the same as `sqrt-Huff`. To summarize, the performance of isogeny-based cryptography on Huff curves is as fast as on Montgomery curves.

Based on our analysis, the Huff curve can be quite practical for implementing isogeny-based cryptography but has limitations. First, as the points of order 2 are all at infinity on Huff curves, it is hard to construct a 2-isogeny formula using w-coordinate so that only e_A with an even number can be used to implement SIDH with Huff curves. The second is that a supersingular Huff curve exists on

$\mathbb{F}_p$, where $p \equiv 7 \mod 8$. This result is contrary to the case where supersingular Montgomery curve exists on $\mathbb{F}_p$ for both $p \equiv 3 \mod 8$ and $p \equiv 7 \mod 8$.

References

1. Azarderakhsh, R., Campagna, M., Costello, C., De Feo, L., Hess, B., Jalali, A., Jao, D., Koziel, B., LaMacchia, B., Longa, P., et al.: Supersingular isogeny key encapsulation. Submission to the NIST post-quantum standardization project, 2017
2. Azarderakhsh, R., Jao, D., Kalach, K., Koziel, B., Leonardi, C.: Key compression for isogeny-based cryptosystems. In: Proceedings of the 3rd ACM International Workshop on ASIA Public-Key Cryptography. pp. 1–10. ACM (2016)
3. Bernstein, D.J., Feo, L.D., Leroux, A., Smith, B.: Faster computation of isogenies of large prime degree. Cryptology ePrint Archive, Report 2020/341 (2020), https://eprint.iacr.org/2020/341
4. Beullens, W., Kleinjung, T., Vercauteren, F.: CSI-FiSh: efficient isogeny based signatures through class group computations. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 227–247. Springer (2019)
5. Bos, J.W., Friedberger, S.J.: Arithmetic considerations for isogeny-based cryptography. IEEE Transactions on Computers 68(7), 979–990 (July 2019)
6. Castryck, W., Decru, T.: CSIDH on the surface. In: International Conference on Post-Quantum Cryptography. pp. 111–129. Springer (2020)
7. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An efficient post-quantum commutative group action. In: Peyrin, T., Galbraith, S. (eds.) Advances in Cryptology – ASIACRYPT 2018. pp. 395–427. Springer International Publishing, Cham (2018)
8. Childs, A., Jao, D., Soukharev, V.: Constructing elliptic curve isogenies in quantum subexponential time. Journal of Mathematical Cryptology 8(1), 1–29 (2014)
9. Costello, C.: B-SIDH: supersingular isogeny Diffie-Hellman using twisted torsion. IACR Cryptol. ePrint Arch. 2019, 1145 (2019)
10. Costello, C., Hisil, H.: A simple and compact algorithm for SIDH with arbitrary degree isogenies. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology – ASIACRYPT 2017. pp. 303–329. Springer International Publishing, Cham (2017)
11. Costello, C., Longa, P., Naehrig, M.: SIDH library (2016-2018). https://github.com/Microsoft/PQCrypto-SIDH
12. Costello, C., Longa, P., Naehrig, M.: Efficient algorithms for supersingular isogeny Diffie-Hellman. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology – CRYPTO 2016. pp. 572–601. Springer, Berlin, Heidelberg (2016)
13. Couveignes, J.M.: Hard homogeneous spaces. (2006), https://eprint.iacr.org/2006/291
14. De Feo, L., Kieffer, J., Smith, B.: Towards practical key exchange from ordinary isogeny graphs. In: Peyrin, T., Galbraith, S. (eds.) Advances in Cryptology – ASIACRYPT 2018. pp. 365–394. Springer International Publishing, Cham (2018)
15. Dryło, R., Kijko, T., Wroński, M.: Efficient Montgomery-like formulas for general Huff's and Huff's elliptic curves and their applications to the isogeny-based cryptography. Cryptology ePrint Archive, Report 2020/526 (2020), https://eprint.iacr.org/2020/526
16. Heo, D., Kim, S., Yoon, K., Park, Y.H., Hong, S.: Optimized CSIDH implementation using a 2-torsion point. Cryptography 4(3), 20 (2020)

17. Huang, Y., Zhang, F., Hu, Z., Liu, Z.: Optimized arithmetic operations for isogeny-based cryptography on Huff curves. In: Australasian Conference on Information Security and Privacy. pp. 23–40. Springer (2020)
18. Jao, D., De Feo, L.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In: Yang, B.Y. (ed.) Post-Quantum Cryptography. pp. 19–34. Springer, Berlin, Heidelberg (2011)
19. Joye, M., Tibouchi, M., Vergnaud, D.: Huff's model for elliptic curves. In: International Algorithmic Number Theory Symposium. pp. 234–250. Springer (2010)
20. Kim, S., Yoon, K., Kwon, J., Park, Y.H., Hong, S.: New hybrid method for isogeny-based cryptosystems using Edwards curves. IEEE Transactions on Information Theory (2019)
21. Kim, S., Yoon, K., Park, Y.H., Hong, S.: Optimized method for computing odd-degree isogenies on Edwards curves. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 273–292. Springer (2019)
22. Meyer, M., Reith, S.: A faster way to the CSIDH. In: Chakraborty, D., Iwata, T. (eds.) Progress in Cryptology – INDOCRYPT 2018. pp. 137–152. Springer International Publishing, Cham (2018)
23. Meyer, M., Reith, S., Campos, F.: On hybrid SIDH schemes using Edwards and Montgomery curve arithmetic (2017), https://eprint.iacr.org/2017/1213
24. Moody, D., Shumow, D.: Analogues of Vélu's formulas for isogenies on alternate models of elliptic curves. Mathematics of Computation 85(300), 1929–1951 (2016)
25. Stolbunov, A.: Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves. Advances in Mathematics of Communication 4(2), 215–235 (2010)
26. Wu, H., Feng, R.: Elliptic curves in Huff's model. Security and Communication Networks 17(6) (2012)

Acknowledgement

Appendix A 4-isogenies on Huff Curves

Although [15] omits the 4-isogeny formula on Huff curves using w-function, by adapting the idea from [17], we additionally present the 4-isogeny formula on Huff curves using w-function. In this section, we shall briefly state the formula for implementing SIDH using 4-isogeny.

Theorem 4 (2-isogenies for $H_{a,b}$ using w-function). *Let $\phi : H_{a,b} \to H_{a',b'}$ be a 2-isogeny with kernel $\{(0,0),(a:b:0)\}$. Let $w = xy$ for $(x,y) \in H_{a,b}$. Then the evaluation of w under ϕ is given by*

$$\phi(w) = \frac{(a^2-b^2)w}{(bw+a)(aw+b)} \tag{9}$$

where

$$a' = \sqrt{-\left(\sqrt{\frac{1}{b^2}} + \sqrt{\frac{1}{a^2}}\right)^2}, \quad b' = \sqrt{-\left(\sqrt{\frac{1}{b^2}} - \sqrt{\frac{1}{a^2}}\right)^2}$$

To derive equation (9), we adapt the method used in [17]. That is, ϕ is first derived from the below composition:

$$H_{a,b} \xrightarrow{\iota} G_{a,b} \xrightarrow{\psi} G_{\hat{a},\hat{b}} \xrightarrow{\iota^{-1}} H_{a',b'}$$

where ι denotes the transformation from a Huff curve to a general Huff curve, ψ is a 2-isogeny on general Huff curve from [24]. Then $\phi = \iota^{-1} \circ \psi \circ \iota$. Similarly, we can derive Huff 2-isogenies for the curve of the form H_c.

Theorem 5 (2-isogenies for H_c using w-function). *Let $\phi : H_c \to H_{c'}$ be a 2-isogeny with kernel $\{(0,0),(c:1:0)\}$. Let $w = xy$ for $(x,y) \in H_c$. Then the evaluation of w under ϕ is given by*

$$\phi(w) = \frac{w(c^2-1)}{(w+c)(cw+1)} \tag{10}$$

where $c' = \mid (c+1) \mid / \mid (1-c) \mid$.

As shown in equation (10), 2-isogeny on H_c using w-function is identical to setting $b = 1$ in equation (9). Also, equation (10) is just reciprocal of the 2-isogeny on H_c using w_{inv} function defined in [17]. Lastly, we state the 4-isogeny on Huff curve using w-function, directly derived from the idea presented in [17].

Theorem 6 (4-isogenies for H_c using w-function). *Let $\phi : H_c \to H_{c'}$ be a 4-isogeny with kernel P such that $w(P) = w_4$ and P has order 4 in H_c. Let $w(Q) = w = xy$ for a point $Q = (x,y) \in H_c$. Then the evaluation of w under ϕ is given by*

$$\phi(w) = \frac{w(w-w_4)^2(ww_4^2 + w - 2w_4)}{(2ww_4 - w_4^2 - 1)(ww_4 - 1)^2}$$

where $c' = (1 + \sqrt{1-w_4^4})/(1 - \sqrt{1-w_4^4})$.

Appendix B Formulas for implementing SIDH-based cryptography

In this section, we present the doubling, tripling, 3-isogeny, and 4-isogeny formula on a Huff curve of the form H_c, using w as a compression function. For corresponding formulas on H_c using w function, please refer to [17].

Doubling Let $P = (x,y)$ be a point on a Huff curve H_c. Let $c = C/D$ and $\hat{c} = \hat{C}/\hat{D}$, where $\hat{c} = \frac{1}{4}\left(c + \frac{1}{c} - 2\right)$. For $w(P) = (W : Z)$ in projective w-coordinates, the doubling of P gives $w([2]P) = (W' : Z')$, where W' and Z' are defined as:

$$W' = 4WZ(\hat{D}(W+Z)^2 + \hat{C} \cdot 4WZ)$$
$$Z' = \hat{D}(W-Z)^2(W+Z)^2$$

The computational cost is $4\mathbf{M} + 2\mathbf{S}$, given $\hat{C}$ and $\hat{D}$. Therefore, instead of using the projective curve coefficient $(C : D)$, it is efficient to use $(\hat{C} : \hat{D}) = ((C-D)^2 : 4CD)$ for implementation.

Tripling Let $P = (x, y)$ be a point on a Huff curve H_c. Let $c = C/D$ and $\hat{c} = \hat{C}/\hat{D}$, where $\hat{c} = \frac{1}{4}\left(c + \frac{1}{c} - 2\right)$. For $w(P) = (W : Z)$ in projective w-coordinates, the tripling of P gives $w([3]P) = (W' : Z')$, where W' and Z' are defined as:

$$W' = W(\hat{D}W^4 - 6\hat{D}W^2Z^2 - 16\hat{C}WZ^3 - 8\hat{D}WZ^3 - 3\hat{D}Z^4)^2$$
$$Z' = Z(3\hat{D}W^4 + 16\hat{C}W^3Z + 8\hat{D}W^3Z + 6\hat{D}W^2Z^2 - \hat{D}Z^4)^2$$

The tripling formula is the same to the case when w_{inv} is used. The computational cost is $7\mathbf{M} + 5\mathbf{S}$, given $\hat{C}$ and $\hat{D}$. For tripling it efficient to keep the projective curve coefficient as $((C - D)^2 : (C + D)^2)$.

3-isogeny Let $P = (x_3, y_3)$ be a 3-torsion point on a Huff curve H_c, $w(P) = (W_3 : Z_3)$. Let $\phi : H_c \to H_{c'}$ be a 3-isogeny generated by a kernel $\langle P \rangle$, such that $H_{c'} = H_c/\langle P \rangle$. Let $Q = (W : Z)$ be another point on H_c. Then the image $w(\phi(Q)) = (W' : Z')$ is computed as:

$$W' = W(WZ_3 - ZW_3)^2$$
$$Z' = Z(WW_3 - ZZ_3)^2$$

and

$$\hat{C} = (W_3 - Z_3)(W_3 + 3Z_3)^3$$
$$\hat{D} = (W_3 + Z_3)(W_3 - 3Z_3)^3$$

where $c' = C'/D'$ and $\hat{C} = (C' + D')^2$ and $\hat{D} = (C' - D')^2$, to continue with the tripling efficiently. The computational cost for 3-isogeny evaluation is $4\mathbf{M} + 2\mathbf{S}$ and the computational cost for computing the coefficient of the image curve is $2\mathbf{M} + 3\mathbf{S}$.

4-isogeny Let $P = (x_4, y_4)$ be a 4-torsion point on a Huff curve H_c, $w(P) = (W_4 : Z_4)$. Let $\phi : H_c \to H_{c'}$ be a 4-isogeny generated by a kernel $\langle P \rangle$, such that $H_{c'} = H_c/\langle P \rangle$. Let $Q = (W : Z)$ be another point on H_c. Then the image $w(\phi(Q)) = (W' : Z')$ is computed as:

$$W' = W(2W_4Z_4Z - W(W_4^2 + Z_4^2))(Z_4W - W_4Z)^2$$
$$Z' = Z(2W_4Z_4W - Z(W_4^2 + Z_4^2))(WW_4 - ZZ_4)^2$$

and

$$\hat{C} = 4Z_4^4 - 4W_4^4$$
$$\hat{D} = 4W_4^4$$

where $\hat{c} = \hat{C}/\hat{D} = \frac{1}{4}\left(c' + \frac{1}{c'} - 2\right)$, to continue with the doubling efficiently. The computational cost for 4-isogeny evaluation is $6\mathbf{M} + 2\mathbf{S}$ and the computational cost for computing the coefficient of the image curve is $4\mathbf{S}$.