Implementation Aspects of Supersingular Isogeny-Based Cryptographic Hash Function

Miraz Uz Zaman(✉), Aaron Hutchinson, and Manki Min

Louisiana Tech University, 71272 Ruston, LA, USA
{muz001,aaronh,mankimin}@latech.edu

Abstract. Supersingular isogeny-based cryptosystems are considered an attractive candidate for the post-quantum cryptographic world due to their smaller key sizes. Based on the traversal in a supersingular isogeny graph (expander graph), Charles, Goren, and Lauter proposed a cryptographic hash function also known as CGL hash. In this paper, we present our study on the implementation-related aspects of the compact variation of the standard CGL hash function using different forms of elliptic curves (Weierstrass, Montgomery, and Legendre). Moreover, we show that some redundant computations in the original propositions of the CGL hash function can be avoided by utilizing the unique characteristics of the different forms of the elliptic curve. We also compared the running time and the total number of collisions through the experiments with the implemented algorithms.

Keywords: Post-quantum cryptography · Supersingular isogeny · Hash · CGL · Elliptic curves · Weierstrass · Montgomery · Legendre

1 Introduction

The cryptographic secure hash function is one of the widely used cryptographic tools which appears in almost all information security applications. Some long-established applications of secured hash functions are commitment schemes, digital signature schemes, message authentication codes, Blockchain, and password encryption. A cryptographic hash function converts an arbitrary length of input bit string to fixed-size data. One of the fundamental properties of a good hash function is its low computational time. Moreover, a secure hash function should be collision-free and preimage resistant, and the output distribution should be uniform. According to [20], a fully-fledged quantum computer can quickly break the currently standardized hash functions. However, based on a US National Academy report [15], we are roughly a decade away from such a fully-fledged quantum computer. Therefore, a post-quantum secured hash function is required

Supported by Louisiana Board of Regents GR301278.

Z. J. Haas et al. (Eds.): WiCON 2022, LNICST 464, pp. 14–27, 2023.
https://doi.org/10.1007/978-3-031-27041-3_2

to resist cryptanalysis for both classical and quantum computers. Moreover, the early deployment of such a post-quantum secured hash function will smooth the transition from pre-quantum to post-quantum cryptography.

In 2016, NIST initiated a process to standardize the post-quantum public-key cryptographic algorithms. After multiple rounds of evaluation and inspection, a handful of algorithms were nominated for the final round. SIKE [6] an Isogeny-based key encapsulation scheme advanced to the final round as an alternative candidate. The SIKE is mainly based on the idea of pseudo-random walks on a supersingular isogeny graph. Charles, Goren, & Lauter et al. first proposed walking on the supersingular isogeny graph (expander graph) to generate a hash function [1] in 2009. The hash function is also known as the acronym *CGL*. In CGL, each bit in the input strings of the hash function is used as the direction to traverse around the graph. Both CGL's and SIKE's security relies on the hardness of finding an isogeny between two supersingular elliptic curves. The best-known attack for both of these cryptosystems have exponential quantum complexity [7, §5]; hence they are quantum secure.

CGL hash function can also be shown as a sequence of 2-isogenies computation on supersingular elliptic curves over the finite field $\mathbb{F}_p^2$. The isogeny computation is mainly carried out through Vélu's [19] formula, where the elliptic curves are expressed in the Weierstrass form. Other than the Weierstrass form, an elliptic curve can also have some other well-known forms [8, §II.B], i.e., Montgomery form, Legendre form, Edward curves, etc. Each form shows a wide variety of computational costs for arithmetic operations and isogeny computations in the prime field. By utilizing unique non-trivial characteristics in the different forms of elliptic curves, some of the redundant computations can be saved from the original proposal of the CGL hash. In this paper, we showed some efficient and compact implementations of CGL on three widely used forms of elliptic curves. To the best of our knowledge, there is one such work [21] on the efficient algorithm of CGL previously. However, the work is only limited to the Weierstrass form. We also compared running times and collisions of our implementations for different prime fields with an extensive number of bit strings.

In Sect. 2 we refresh some basics, i.e., elliptic curve, isogenies, and the CGL hash. Some compact implementations of CGL hash on different forms of elliptic curves are discussed in Sect. 3. We present the theoretical and experimental results in Sect. 4 where we compare the running time and the total number of collisions for a comprehensive number of bit strings.

2 Definitions and Background

2.1 Elliptic Curve and Isogeny

An elliptic curve is a non-singular projective curve of genus 1 over a finite field $\mathbb{F}_q$ of q elements where $q = p^k$ with some prime p. The elliptic curve E over $\mathbb{F}_q$ can be represented as the following affine equation,

$$E/\mathbb{F}_q : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6,$$

where $a_1, a_2, a_3, a_4, a_6 \in \mathbb{F}_q$. If a p-torsion subgroup of an elliptic curve $E[p]$ is non-trivial then the curve is *supersingular* otherwise it's *ordinary* [16, §V.3]. As the CGL is primarily based on supersingular isogeny graphs, in this paper, we interchangeably use the elliptic curve and supersingular elliptic curve. Let E and $\tilde{E}$ be an elliptic curve over $\mathbb{F}_q$. An isogeny $\phi : E \rightarrow \tilde{E}$ is a surjective morphism of curves over $\mathbb{F}_q$ so that $\phi(O_E) = O_{\tilde{E}}$ The rational function ϕ is also a group homomorphism which maps $E(\mathbb{F}_q) \rightarrow \tilde{E}(\mathbb{F}_q)$. Here, the elliptic curves $E/\mathbb{F}_q$ and $\tilde{E}/\mathbb{F}_q$ are isogenous as there exists an isogeny ϕ between them. The existence of ϕ also ensures that there is presence of dual isogeny $\hat{\phi}$ so that $\hat{\phi} : \tilde{E}/\mathbb{F}_q \rightarrow E/\mathbb{F}_q$ [16, §III.6.1]. This also creates an equivalence relation on the finite set of isomorphism classes of elliptic curves defined over $\mathbb{F}_q$. The equivalent class of the elliptic curve under this relation is defined as the isogeny class. For a seperable and non-constant isogeny, the degree of isogeny $deg\phi$ is equal to the cardinality of the kernel i.e.,. $deg\phi = |ker(\phi)|$. Therefore an isogeny can be identified by its kernel. For example, *l-isogeny* would have the kernel of size l. Vélu's formula uses the knowledge of the size of the kernel to compute the isogeny. Two isogenous curves fall into the same isogeny class, i.e., either both are supersingular or both are ordinary.

2.2 Isogeny Graph

In this discussion we will fix field $\mathbb{F}_q = \mathbb{F}_{p^2}$, as all the supersingular elliptic curve in characteristics p are on the same isogeny class and can be defined in either $\mathbb{F}_p$ or $\mathbb{F}_{p^2}$ [11]. The isomorphic classes of the elliptic curve can be represented through j-invariant, which is a unique a priori element of $\mathbb{F}_{p^2}$. A graph of l-isogeny graph can be defined by a set of elliptic curves in $\mathbb{F}_{p^2}$ as the vertices and edges are degrees l-isogeny between them. Such a graph consists of both ordinary and supersingular components. Because of the isogenous relation, supersingular and ordinary components of isogeny graphs are disconnected.

Pizer [12,13] shows that the supersingular component has the property of the Ramanujan graph [10]. Ramanujan graph is a highly connected regular graph also known as expander graph. The details on the Ramanujan graph can be found on [2,10]. Because of the strong connectivity found in Ramanujan graphs, there is only one supersingular component in the isogeny graph which will be the main interest of this paper. The vertices in the isogeny graph are represented as j-invariants, which can be computed from the curve equation, and the total number of vertices is close to $p/12$. CGL hash function employs $l = 2$ isogeny graph.

Figure 1 shows such a 2-isogeny graph $\mathbb{F}_{211^2}$. There are 18 vertices in the graph, which corresponds to supersingular j-invariants, and 2-isogenies refer to the edges between them. All of the 17 vertices show the expected behavior of 2-isogenies except the node with j-invariants of 40. However, it can be shown that as p expands, the total number of nodes with such an exceptional behavior stays small.

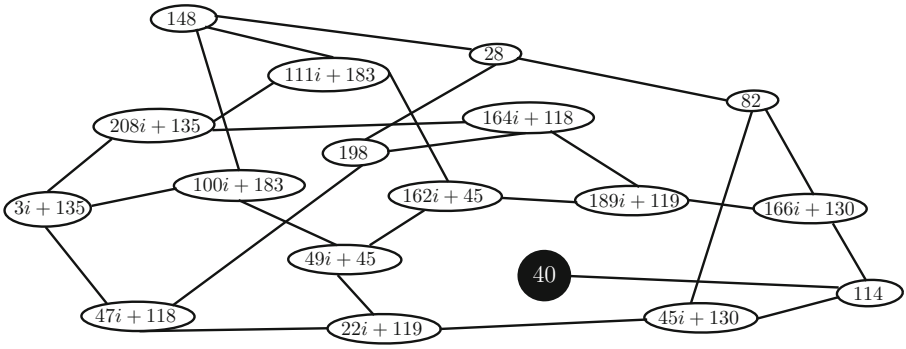


Fig. 1. The 2-isogeny graph for $\mathbb{F}_{211^2}$

2.3 CGL Hash

The rapid mixing property for a random walk in an expander graph showed strong pseudo-random behavior. Such a behavior has been widely used in different cryptographic constructions [4,9]. The principle idea of the CGL hash function is to use the same pseudo-random behavior of supersingular isogeny graph to produce a collision-free hash function. In CGL, the input of the hash function is used as the direction to traverse around the supersingular isogeny graph from the fixed starting vertex with backtracking avoided, and the end vertex is the output of the hash function. As we can see that changing the starting vertex or elliptic curve yields a different hash function, we can define a family of hash functions based on the starting curve. A 2-isogeny supersingular graph is ideally a 3-regular graph. As the CGL hash function avoids backtracking, there are two edges to follow from each vertex to the next vertex. Based on a uniform convention, these two edges will be assigned to the bit 1 and 0 leading to the next vertex.

Let's consider, a starting vertex or an elliptic curve E_1 which is defined as $y^2 = f_1(x)$. Now if we can factor the cubic $f(x)$ into $(x - x_1)(x - x_2)(x - x_3)$, 3 non-trivial 2 torsion points of E_1 can be represented as $\{(x_1, 0), (x_2, 0), (x_3, 0)\}$. Now, consider the computation of an isogeny ϕ that results an isogenous curve E_2 or $y^2 = f_2(x)$ where $ker\phi = \langle(x_1, 0)\rangle$. The other two torsion points $(x_2, 0)$ and $(x_3, 0)$ will map to the same points on curve E_2 i.e., $\phi(x_2, 0) = \phi(x_3, 0)$. The subgroup generated by kernel $\langle(x_2, 0)\rangle$ or $\langle(x_3, 0)\rangle$ on E_2 is the dual isogeny $\hat{\phi}$ return to the curve E_1.

In CGL, the dual isogeny is called the backtracking isogeny. So backtracking can be averted if new cubic $f_2(x)$ can factored out by $(x - \hat{x_2})$ where $\hat{x_2}$ is the x-coordinate of $\phi(x_2, 0)$. The remaining part of the factor is a quadratic term, and the two quadratic roots will be the x-abscissa of the other two torsion points. CGL hash proposes that a convention for ordering the point can be fixed to choose one torsion point with respect to the input bit. The chosen point will determine the next walk in the graph. Consequently, for the n-bit input string,

the CGL hash takes a walk of n-steps. The output of the hash function is the j-invariant of the last elliptic curve. The flowchart of the CGL hash function is shown in Fig. 2.

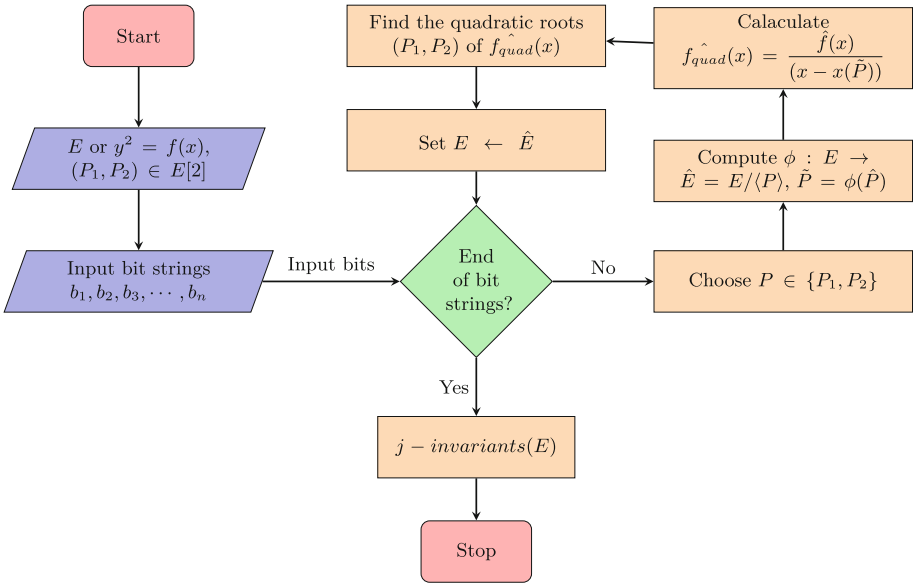


Fig. 2. Flowchart of CGL Hash

3 Compact CGL Hash Algorithms

3.1 Short Weierstrass Form

Any elliptic curve E over a field $\mathbb{F}_{p^2}$ of prime p, with $p > 3$ can be transformed into short Weierstrass form,

$$E : y^2 = x^3 + \alpha x + \beta,$$

where $(\alpha, \beta) \in K$ and $4\alpha^3 + 27\beta^2 \neq 0$. Later condition ensures that there are 3-distinct roots. The j-invariant of the elliptic curve E can be defined by,

$$j(E) = 1728\frac{4\alpha^3}{4\alpha^3 + 27\beta^2}.$$

2 Isogenies. Let γ be a root of $x^3 + \alpha x^2 + \beta$. An algebric polynomial division by $(x - \gamma)$ shows that,

$$\frac{x^3 + \alpha x + \beta}{(x - \gamma)} = (x^2 + \gamma x + (\alpha + \gamma^2))$$

Therefore, the elliptic curve E can be expressed as,

$$E : y^2 = (x - \gamma)(x^2 + \gamma x + (\alpha + \gamma^2)).$$

So, the 3 torsion points of order 2 are $\{(\gamma, 0), (\frac{-\gamma \pm \sqrt{-4\alpha - 3\gamma^2}}{2}, 0)\}$. According to [18, Theorem 6.13], if $(x_0, 0)$ is a torsion point of order 2 applying 2-isogeny with $ker(\phi) = \langle (x_0, 0) \rangle$ produce the following mapping,

$$\phi : E \to \tilde{E} : y^2 = x^3 + \alpha' x + \beta'$$

$$(x, y) \mapsto \left(\frac{x^2 - x_0 x + t}{x - x_0}, \frac{(x - x_0)^2 - t}{(x - x_0)^2} y \right),$$

where, $t = 3x_0^2 + \alpha$, $\alpha' = \alpha - 5t$, $w = x_0 t$, and $\beta' = \beta - 7w$.

CGL Hash Algorithm for Short Weierstrass Form. The four input parameter of the Algorithm algspswei are α, β, γ over a field of $\mathbb{F}_{p^2}$, and n-bit message $M = (b_1, b_2, \cdots, b_n)$. Here, α, β forms a supersingular elliptic curve E in weierstrass form $y^2 = x^3 + \alpha x + \beta$ over a field of $\mathbb{F}_{p^2}$, γ is one of the root of $x^3 + \alpha x + \beta$. We calculate x-abscissa of other two torsion points (x_0, x_1) in step 2 and step 3 of Algorithm algspswei. From steps 4 to 7, we set a uniform convention to define x_0 and x_1 to navigate the next walk in the graph. The convention is simply to choose the maximum of two values from step 3 as x_0 and the minimum value as x_1 for bit 1 and vice versa for bit 0. Steps 9 and 10 is utilizing Vélu's formula to calculate the next isogenous elliptic curve Weierstrass parameter α and β. The x abscissa of the dual isogeny or the backtracking point in the new isogenous curve is calculated in step 11. The output of the algorithm is the j-invaraint of the last elliptic curve which is calculated in step 13.

Algorithm 1. CGL hashing using Weierstrass Curve

Input: α, β, γ, M
Output: j-invariant
1: **for** *bit* in M **do**
2: $\delta \leftarrow \sqrt{-4\alpha - 3\gamma^2}$
3: $(x_0, x_1) \leftarrow (\frac{-\gamma + \delta}{2}, \frac{-\gamma - \delta}{2})$
4: **if** $bit \leftarrow 1$ **then**
5: $(x_0, x_1) \leftarrow \max(x_0, x_1), \min(x_0, x_1)$
6: **else** $\{bit \leftarrow 0\}$
7: $(x_0, x_1) \leftarrow \min(x_0, x_1), \max(x_0, x_1)$
8: **end if**
9: $t \leftarrow 3x_0^2 + \alpha$, $w \leftarrow x_0 t$
10: $\alpha \leftarrow \alpha - 5t$, $\beta \leftarrow \beta - 7w$
11: $\gamma \leftarrow \frac{x_1^2 - x_0 x_1 + t}{x_1 - x_0}$
12: **end for**
13: **return** j-invariant $\leftarrow 1728 \frac{4\alpha^3}{4\alpha^3 + 27\beta^2}$

3.2 Montgomery Form

An elliptic curve in Montgomery form over a field $\mathbb{F}_{p^2}$ can be defined by equation

$$E_{(A,B)} : By^2 = x^3 + Ax^2 + x = x(x^2 + Ax + 1),$$

where, $A, B \in \mathbb{F}_{p^2}$ and $B(A^2 - 4) = 0$. The parameter A primarily controls the geometry of $E_{(A,B)}$. The $j-$invariants of the curve can be expressed as,

$$j(E_{(A,B)}) = \frac{256(A^2 - 3)^3}{A^2 - 4}.$$

2 Isogenies. From the curve equation $E_{(A,B)}$, it can be easily verified that a K-rational point $Q = E(0, 0)$ is order of 2. If $P = (x_P, 0)$ is another K-rational point of order 2, then $x_P^2 + Ax_P + 1 = 0$. So, the other two rational points of order 2 are $(\frac{-A \pm \sqrt{A^2-4}}{2}, 0)$. According to [14, §4.2], applying 2 isogeny with $ker(\phi) = \langle P \rangle$ generates the following mapping,

$$\phi : E \to \tilde{E} : by^2 = x^3 + ax^2 + x = x(x^2 + ax + 1)$$

$$(x, y) \mapsto (g(x), yg'(x)),$$

with $b = x_P B$, $a = 2(1 - 2x_P)^2 = 2 - (-A \pm \sqrt{A^2 - 4})^2$, and $g(x) = x\frac{xx_P - 1}{x - x_P}$. Moreover, [14, Corollary 1] shows that kernel of dual of ϕ is $\langle (0, 0) \rangle$. Hence, $ker(\hat{\phi}) = \langle (0, 0) \rangle$. On the other hand, the authors in [3, Eqs. (18) & (19)] outlines 2-isogeny for $ker(\phi) = \langle (0, 0) \rangle$ as, ‘

$$\phi : E \to F : By^2 = x^3 + (A + 6)x^2 + 4(2 + A)x$$

$$(x, y) \mapsto \left(\frac{(x - 1)^2}{x}, y(1 - \frac{1}{x^2}) \right).$$

The authors in [14, Remark 6] describes an isomorphism of F with the following mapping,

$$\psi : F \to G : \frac{B}{\sqrt{A^2 - 4}} y^2 = x^3 - \frac{2A}{\sqrt{A^2 - 4}} x^2 + x$$

$$(x, y) \mapsto \left(\frac{x + A + 2}{\sqrt{A^2 - 4}}, \frac{y}{\sqrt{A^2 - 4}} \right).$$

Now by applying 2-isogeny with kernel of $\langle (0, 0) \rangle$ following a simple algebric computation on coefficients of x^2 can be shown that, dual of ψ is also $\langle (0, 0) \rangle$.

$$\frac{-2\frac{-2A}{\sqrt{A^2-4}}}{\sqrt{\frac{4A^2}{A^2-4} - 4}} = \pm A$$

Hence, $ker(\hat{\psi}) = \langle (0, 0) \rangle$.

CGL Hash Algorithm for Montgomery Form. The two input parameters of the Algorithm algspsmont are the Montgomery curve parameters $A \in \mathbb{F}_p^2$ so that $y^2 = x^3 + Ax^2 + x$ supersingular elliptic curve and n-bit message M $(b_1, b_2, \cdots, b_n)$. As we know from the previous description, the dual of the kernel is always $\langle(0,0)\rangle$. So now, for the CGL hashing algorithm $(0,0)$ is the backtracking point. Hence, we don't need to calculate the backtracking point and other 2 torsion points can be assigned for bit 1 and 0 based on some uniform convention. From step 2 to 8, such a convention have been set to choose the next isogenous curve. The hash function's output will be the j-invariant of the last isogenous curve.

Algorithm 2. CGL hashing algorithm using Montgomery Curve

Input: A, B, M
Output: j-invariant
1: **for** *bit* in M **do**
2: $C \leftarrow \sqrt{A^2 - 4}$
3: $A_0 \leftarrow 2A^2 - 4 + AC$
4: $A_1 \leftarrow 2A^2 - 4 - AC$
5: **if** $bit \leftarrow 1$ **then**
6: $A \leftarrow 2 - \max(A_0, A_1)$
7: **else** $\{bit \leftarrow 0\}$
8: $A \leftarrow 2 - \min(A_0, A_1)$
9: **end if**
10: **end for**
11: **return** j-invariant $\leftarrow \frac{256(A^2-3)^3}{A^2-4}$

3.3 Legendre Form

Another interesting form of elliptic curve is Legendre Form. If $\lambda \in \mathbb{F}_p^2$ and $\lambda \neq 0, 1$, elliptic curve E in Legendre form can be represented as,

$$E_\lambda : y^2 = x(x-1)(x-\lambda).$$

Here, Legendre coefficient, λ for E is not unique. In fact each of

$$\lambda, \frac{1}{\lambda}, 1-\lambda, \frac{1}{1-\lambda}, \frac{\lambda}{\lambda-1}, \frac{\lambda-1}{\lambda}$$

yields an Isomorphic E curve. The j-invariant of E_λ is,

$$j(E_\lambda) = 2^8 \frac{(\lambda^2 - \lambda + 1)^3}{\lambda^2(\lambda-1)^2}.$$

Considering the fact that most supersingular elliptic curves will have three 2-isogenies, we can infer that those six possible Legendre coefficients for the same j-invariant can be grouped into three pairs where each pair is directly related to each of the three 2-isogenies.

2 Isogenies. The three 2-torsion points are readily available from the elliptic curve expression, E_λ. The points in affine form are $(0,0),(1,0),(\lambda,0)$. From [5, Theorem 4 & 5], it can be shown that applying 2 isogeny when $ker(\phi) \neq (\lambda,0)$ will produce the following mapping,

$$\phi : E \to \tilde{E} : y^2 = x(x-1)(x-\lambda').$$

The relationship between λ and λ' can be shown with the following equation.

$$\lambda = \begin{cases} \frac{(\sqrt{\lambda}+1)^2}{4\sqrt{\lambda}} = \frac{\lambda+1+2\sqrt{\lambda}}{4\sqrt{\lambda}} & \text{if } ker(\phi) = \langle(0,0)\rangle \\ \frac{-(\sqrt{1-\lambda}-1)^2}{4\sqrt{1-\lambda}} = \frac{\lambda-2+2\sqrt{1-\lambda}}{4\sqrt{1-\lambda}} & \text{if } ker(\phi) = \langle(1,0)\rangle \\ \frac{(\sqrt{\lambda}+\sqrt{\lambda-1})^2}{4\sqrt{\lambda}\sqrt{\lambda-1}} = \frac{2\lambda-1+2\sqrt{\lambda}\sqrt{1-\lambda}}{4\sqrt{\lambda}\sqrt{\lambda-1}} & \text{if } ker(\phi) = \langle(\lambda,0)\rangle \end{cases}$$

CGL Hash Algorithm for Legendre Form. The input parameters of the Algorithm algspsleg are Legendre parameter $\lambda \in \mathbb{F}_p^2$ so that $y^2 = x(x-1)(x-\lambda)$ is a supersingular elliptic curve and n-bit message M $(b_1, b_2, \cdots, b_n)$. Similar to the Montgomery form algorithm, the backtracking point is fixed which is $(\lambda, 0)$. Therefore, from step 2 to 5, we set kernel point $(1,0)$ for bit 1 and $(0,0)$ for bit 0. The output of the hash function algorithm will be the j-invariant of the last isogenous curve.

Algorithm 3. CGL hashing algorithm using Legendre Curve

Input: λ, M
Output: j-invariant
1: **for** *bit* in M **do**
2: **if** $bit \leftarrow 1$ **then**
3: $\lambda \leftarrow \frac{\lambda+1+2\sqrt{\lambda}}{4\sqrt{\lambda}}$
4: **else** $\{bit \leftarrow 0\}$
5: $\lambda \leftarrow \frac{\lambda-2+2\sqrt{1-\lambda}}{4\sqrt{1-\lambda}}$
6: **end if**
7: **end for**
8: **return** j-invariant $\leftarrow 2^8 \frac{(\lambda^2-\lambda+1)^3}{\lambda^2(\lambda-1)^2}$

3.4 Yoshida et al. Proposed Method for Computing CGL Hash

Yoshida et al. proposed two compact methods for computing CGL hash in [21, §5] for the Weierstrass form of supersingular elliptic curves. Both methods mainly rely on their proposed Theorem [21, Theorem 1] which deduces a relationship between the current and following curve parameters using the non-backtracking torsion point. As both methods theoretically share the exact cost, we implemented their proposed method 1.

4 Result and Discussion

4.1 Algorithmic Cost Comparison

Tables 1 and 2 compare the computational cost for all the three different algorithms described in Sect. sectionspsAlgos. The result in the tables is based on the frequently used assumption that I = 100M, S = 0.67M, and SR = (0.67 log p + 100)M, where I, S, SR, and M represent Inversion, Squaring, Square root, and Multiplication, respectively.

The computational cost measure for calculating 2-isogeny sequences for n-bit input strings and determining the j-invariant of the last isogenous curve. The comparison among the algorithms shows that the algorithm for Montgomery and Legendre form has a lower computational cost than the algorithm for the Weierstrass form. Moreover, simulation result from the implementation of the algorithm in the following subsection also shows the same result.

Table 1. CGL Hash operation counts for different algorithms

Algorithm	Curve Model	Counts			
		Multiplication	Square	Square root	Inversion
1	Weierstrass	$4n + 1$	$3n + 2$	n	$n + 1$
2	Montgomery	$2n + 1$	$n + 1$	n	1
3	Legendre	2	3	n	$n + 1$

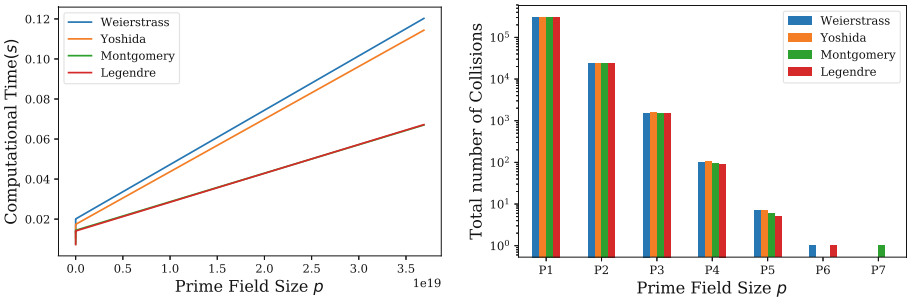
Table 2. CGL Hash costs comparison for different algorithms

Algorithm	Curve Model	Costs in M
1	Weierstrass	(206 + 0.67 log p)n + 102.33
2	Montgomery	(102.67 + 0.67 log p)n + 101.67
3	Legendre	(200 + 0.67 log p)n + 104

4.2 Simulation Result

To evaluate the performance of the algorithms algspswei, algspsmont, and algspsleg we wrote a Python script using SageMath [17]. We also implemented the algorithm proposed in [21, §5.3] and compared it with our compact algorithms. All the algorithms are simulated in 64 bit processor Intel®Core™ i5-7500 CPU @ 3.40 GHz x 4 in Ubuntu 18.04.6 LTS operating system. Moreover, we identified the algorithm's running time and the total number of collisions as the comparison criteria as "low computational time" and "collision avoidance" are the two fundamental properties of the hash function. The running time and total collisions have been computed for 2^{20} number of unique 128-bit input strings with different finite (prime) fields. Here, running time means the difference between starting

and ending times of the algorithm for each input bit string. To find the central tendency of running time, we took the arithmetic mean of all the 2^{20} unique input bit strings. On the other hand, the total number of collisions is the input bit strings that end up on the same j-invariant. We used the SIKE proposed elliptic curve, or its isomorphic curve over $\mathbb{F}_p^2$, as the starting curve for all the algorithms. The starting elliptic curves are $y^2 = x^3 - 11x + 14$, $y^2 = x^3 + 6x^2 + x$, and $y^2 = x(x-1)(x-17+12\sqrt{2})$ over $\mathbb{F}_p^2$ for Weierstrass, Montgomery, and Legendre forms respectively.



(a) Computational time

(b) Total number of collisions

Fig. 3. Average computation time and total number of collisions for 2^{20} number of 128 input bit strings over different size of finite field $\mathbb{F}_{p^2}$. P1, P2, P3, P4, P5, P6, P7 in Fig. 3b are representing the prime number $2^{24} + 43, 2^{28} + 3, 2^{32} + 15, 2^{36} + 31, 2^{40} + 15, 2^{44} + 7, 2^{48} + 75$ respectively.

Figure 3 shows the simulation result for all four algorithms. The algorithm proposed in [21, §5.3] is designated as Yoshida, and the other three algorithms are represented with the corresponding curve form name. Figure 3a shows that computational time increase with the field size and Algorithm algspsmont, algspsleg have lower computational time than other two algorithms. The total number of collisions decreases exponentially with the increasing prime field sizes shown in Fig. 3b.

In another experiment, we tested the total number of collisions for different sizes of input bit strings on all the implemented algorithms. As we know, for a prime field of size p, there are approximately $\frac{P}{12}$ number of j-invariant or unique isomorphic supersingular elliptic curve. In this experiment, we choose $p \approx 2^{21}$ and checked the collisions for all the combinations of input bit strings up to the length of 18. Figure 4 shows that the number of collisions increases exponentially as the length of the input bit string expands. The more rapid increase of the number of collisions by Montgomery form curves needs more detailed analysis with different starting elliptic curves and prime fields, but one thing clear is that the collision increase like this is not acceptable behavior in a cryptographic hash function. This experiment was designed to choose the proper size of the prime field in the design of a secure hash function.

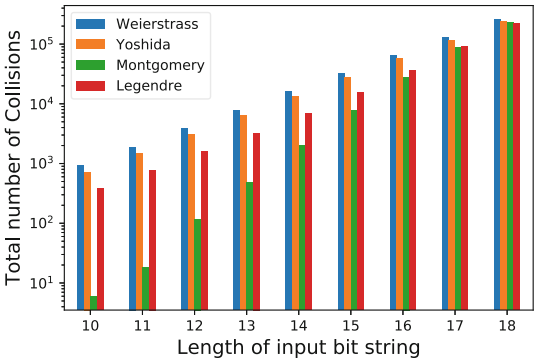


Fig. 4. Total number of collisions for different length of input bit strings

5 Conclusion

In this paper, we presented four different compact implementations of 2-isogeny sequences or a CGL hash function. Out of the four different implementations of the compact CGL function, we found that Legendre and Montgomery forms lead to the best result in computation time. This is the result of the effective use of the fixed backtracking isogeny properties of Montgomery and Legendre form curves which were one of the performance bottlenecks in the standard CGL function. On the other hand, Legendre form-based implementation's collision rates were higher than those for Montgomery form while still lower than those for the other two implementations. We believe that this could be the result of short-length cycles during the computation with Legendre form and we'll dig into the details to find more explanations and solutions.

In future work, we will focus on further reducing the cost of traversal in supersingular isogeny graph as well as the time and collision trade-off study through comparison with other variations of CGL function. We strongly believe a cryptographic hash function's key aspect must be security and a faster hash function with poorer security cannot be accepted.

Our future work also includes the in-depth study of the relation between the six Legendre coefficients and the three 2-isogenies as mentioned in Sect. 3.3 that can lead to a more efficient design of a supersingular isogeny-based cryptographic hash function. If we can use it to avoid or mitigate the calculation of square roots which is an expensive operation in supersingular isogeny graphs, it will provide not only a better way to design a new hash function but also a better way to improve the other cryptographic functions based on supersingular isogeny.

References

1. Charles, D.X., Lauter, K.E., Goren, E.Z.: Cryptographic hash functions from expander graphs. J. Cryptol. **22**(1), 93–113 (2007). https://doi.org/10.1007/s00145-007-9002-x
2. Costache, A., Feigon, B., Lauter, K., Massierer, M., Puskás, A.: Ramanujan graphs in cryptography. In: Balakrishnan, J.S., Folsom, A., Lalín, M., Manes, M. (eds.) Research Directions in Number Theory. AWMS, vol. 19, pp. 1–40. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-19478-9_1
3. Feo, L.D., Jao, D., Plût, J.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. J. Math. Cryptol. **8**(3), 209–247 (2014). https://www.degruyter.com/document/doi/10.1515/jmc-2012-0015/html. https://doi.org/10.1515/jmc-2012-0015
4. Goldreich, O.: Candidate one-way functions based on expander graphs. In: Goldreich, O. (ed.) Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation. LNCS, vol. 6650, pp. 76–87. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22670-0_10
5. Hutchinson, A., Elliott, J.: Supersingular isogeny Diffie-Hellman with Legendre form. J. Math. Cryptol. **1**, 19 (2022). https://doi.org/10.1515/JMC.2008.008
6. Jao, D., et al.: Supersingular isogeny key encapsulation. Submission NIST Post-Quantum Standard. Proj. **152**, 154–155 (2017)
7. Jao, D., De Feo, L.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 19–34. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25405-5_2
8. Lara-Nino, C., Díaz-Pérez, A., Morales-Sandoval, M.: Elliptic curve lightweight cryptography: a survey. IEEE Access **6**, 72514–72550 (2018). https://doi.org/10.1109/ACCESS.2018.2881444
9. Lauter, K.E., Charles, D.X., Goren, E.Z.: Pseudorandom number generation with expander graphs (Mar 15 2011), uS Patent 7,907,726
10. Lubotzky, A., Phillips, R., Sarnak, P.: Ramanujan graphs. Combinatorica **8**(3), 261–277 (1988)
11. Mestre, J.F.: La méthode des graphes. Exemples et applications. In: Proceedings of the International Conference on Class Numbers and Fundamental Units of Algebraic Number Fields (Katata), pp. 217–242. CiteSeer (1986)
12. Pizer, A.K.: Ramanujan graphs. Computational perspectives on number theory, Chicago, IL. AMS/IP Stud. Adv, Math **7**, 159–178 (1995)
13. Pizer, A.K.: Ramanujan graphs and Hecke operators. Bull. Am. Math. Soc. **23**(1), 127–137 (1990)
14. Renes, J.: Computing isogenies between montgomery curves using the action of (0, 0). In: Lange, T., Steinwandt, R. (eds.) PQCrypto 2018. LNCS, vol. 10786, pp. 229–247. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-79063-3_11
15. National Academies of Sciences, Engineering Medicine : Quantum computing: progress and prospects (2019). https://doi.org/10.17226/25196
16. Silverman, J.H.: Heights and elliptic curves. In: Cornell, G., Silverman, J.H. (eds.) Arithmetic Geometry, pp. 253–265. Springer (1986). https://doi.org/10.1007/978-1-4613-8655-1_10
17. Stein, W., et al.: Sage Mathematics Software (Version 9.4.0). The Sage Development Team (2021). https://www.sagemath.org/
18. Sutherland, A.: Lecture 6: Isogeny kernels and division polynomials. In: Elliptic Curves–MIT Course No. 18.783. Cambridge MA (2019). https://math.mit.edu/classes/18.783/2019/LectureNotes6.pdf. MIT OpenCourseWare

19. Vélu, J.: Isogénies entre courbes elliptiques. Comptes-Rendus de l'Académie des Sciences, Série I **273**, 238–241 (Juillet 1971)
20. Webber, M., Elfving, V., Weidt, S., Hensinger, W.K.: The impact of hardware specifications on reaching quantum advantage in the fault tolerant regime. AVS Quant. Sci. **4**(1), 013801 (2022)
21. Yoshida, R., Takashima, K.: Simple algorithms for computing a sequence of 2-isogenies. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 52–65. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00730-9_4