IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 7 October 2024, accepted 28 October 2024, date of publication 5 November 2024, date of current version 13 November 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3491790

RESEARCH ARTICLE

Comprehensive Analysis and Implementation of Isogeny-Based Hash Functions

DONGHOE HEO[1], JEONGHWAN LEE[1], TAEHOON KANG[1], SUHRI KIM[2], AND SEOKHIE HONG[1]

[1]School of Cybersecurity, Korea University, Seoul 02841, South Korea
[2]School of Mathematics, Statistics, and Data Science, Sungshin Women's University, Seoul 02844, South Korea

Corresponding author: Suhri Kim (suhrikim@sungshin.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT, Ministry of Science and ICT) (No. NRF-2020R1A2C1011769).

ABSTRACT This paper analyzes the security and performance of the isogeny-based hash functions. The isogeny-based hash function was first proposed by Charles, Goren, and Lauter, and is referred to as the CGL hash function. However, Lauter and Petit demonstrated that a collision could be found if the endomorphism ring of the starting curve is known. Subsequently, isogeny hash functions that mitigate the Lauter-Petit attack have been proposed. This paper analyzes three methods that counter the Lauter-Petit attack: the methods proposed by Panny, Zaman and Min, and Larsson. In particular, we propose a new isogeny-based hash function SHH, which exploits Panny's method with Hessian curves. We then analyze the security and performance of the proposed SHH along with the methods by Zaman and Min (SCH) and Larsson (SLH). More specifically, the security was analyzed in the context of the Lauter-Petit attack, collision resistance, and fault tolerance. The analysis in this paper shows that all three algorithms can counter the Lauter-Petit attack. However, for SCH, we demonstrated that collision pairs could be found with carelessly chosen parameters. This paper also provides guidelines for selecting parameters to make SCH collision-resistant. From a fault-tolerance perspective, SCH and SLH are not fault-tolerant. We also present the results of implementing the three algorithms in SageMath. The implementation results show that at the 128-bit security level, hashing a 256-bit message takes 0.130s for SCH, 0.125s for projective-SLH, and 0.162s for SHH. As can be seen from the implementation results, projective-SLH is the most efficient, while SHH performs slower due to the use of a larger finite field.

INDEX TERMS Supersingular isogeny, provably secure hash, post-quantum cryptography.

I. INTRODUCTION

Hash functions are critically used in cryptographic algorithms that require data integrity, such as digital signatures, message authentication codes, and password storage. The currently used cryptographic hash functions, mostly follow the Merkle-Damgård method, which prevents the generation of collisions by iterating the compression function [19], [20]. The Merkle-Damgård construction-based hash function is fast and guarantees collision resistance in a computational manner. However, with the discovery of collision vulnerabilities in SHA-1 [21], NIST launched a competition for a new Secure Hash Algorithm, and the need for provable hash functions was raised. A provable hash function is a hash function in which finding a collision is reducible to solving well-known computationally hard problems. Examples in this area include those based on the discrete logarithm, factorization [23], the syndrome decoding problem for linear codes [22], etc. Although provable hash functions have the drawback of being significantly slow in speed compared to dedicated hash functions such as SHA, as they have the advantage of being provable from a security analysis perspective, active research is being conducted in this area. On the other hand, since provable hash functions are designed based on mathematical

The associate editor coordinating the review of this manuscript and approving it for publication was Sedat Akleylek.

problems, it is important that these underlying problems remain unsolved in the current computing environment. With the recent advance in quantum computing, it is also crucial for provable hash functions to use underlying problems that are secure in a quantum computing environment. Therefore, the current research direction for provable hash functions is important not only in terms of efficiency but also in ensuring quantum resistance. As a result, isogeny-based hash functions, which are based on the difficulty of finding isogenies between two elliptic curves, are actively being researched.

The first application of elliptic curve isogenies to hash functions was proposed in 2007 by Charles, Goren, and Lauter. In [1], a provable hash function was introduced using expander graphs. The hash function takes an input and uses it to determine a path through the graph, with the final vertex serving as the function's output. The proposed hash function can be applied to any expander graph. In [1], the use of ℓ-isogeny graphs of supersingular elliptic curves was proposed as the main instantiation, so that the instantiation of such a hash function using an isogeny graph is known as the CGL hash function. The CGL hash computes a 2^n-isogeny to compute a hash value for an n-bit message, where each bit's corresponding 2-isogeny is computed to prevent backtracking.

The CGL hash is based on the security of the supersingular isogeny problem, which has attracted considerable attention due to its exponential complexity even in quantum computing environments. However, it has the drawback of being inefficient, which has led to research focused on optimizing its speed. In 2017, Doliskani et al. enhanced the performance of the CGL hash function by using cyclic isogenies on a Montgomery curve [2]. Inspired by the implementation technique in SIDH [24], by dividing the message m into n-bit message blocks m_i, they compute a cyclic 2^n-isogeny with $P + [m_i]Q$ as the kernel, where P and Q are full 2^n-torsion generators. This method is more efficient than the bit-by-bit computation used in the original CGL hash function. On the other hand, since elliptic curves in CGL hash function are defined over $\mathbb{F}_{p^2}$, the finite field of CGL hash is more costly compared to other cryptographic algorithms that use a base field. From this perspective, in [25], by taking advantage of the Weil restriction, they transform all the arithmetic operations over $\mathbb{F}_{p^2}$ into those over $\mathbb{F}_p$, which speeds up the computation of the hash function by about 30%.

However, in 2018, Lauter and Petit demonstrated that the collisions in the CGL hash can be found by identifying an isogeny cycle if the endomorphism ring of the starting curve is known [3], [4]. Although several works have explored methods for generating arbitrary supersingular curves [5], an efficient way to generate a supersingular curve without revealing its endomorphism ring remains unknown. As a result, the CGL hash function lacks collision resistance unless supported by a trusted authority.

In 2019, Panny proposed a method to counter the Lauter-Petit attack by using a specific portion of the supersingular isogeny graph [8]. Instead of using the entire ℓ isogeny graph, the proposed method suggests using r isogenies, where $r < \ell$. Under a few heuristics, by appropriately adjusting ℓ and r, the method in [8] prevents collisions and permits the use of arbitrary starting curves while still providing sufficient security.

While Panny proposed a variant of the CGL hash function to counter the Lauter-Petit attack, research has also been conducted on proposing an isogeny-based hash function in a different form than the CGL hash to counter the Lauter-Petit attack. In 2023, Zaman and Min introduced a single-compression hash function based on point evaluation under supersingular isogeny [6]. They overcame the Lauter-Petit attack by disclosing an image of a torsion point as a hash value, rather than revealing a codomain curve of an isogeny. Additionally, they proposed a technique for treating multiple message blocks as a single block through a preprocessing step. In 2024, Larsson proposed a supersingular isogeny hash function based on the Lattès map on an elliptic curve [7]. Larsson demonstrated that the Lauter-Petit attack can be mitigated by incorporating a certain amount of scalar multiplication into the CGL hash computation. By exploiting a simple structure of the Lattès map, Larsson also introduced application techniques such as keyed and dynamic hashing.

As stated above, extensive research has been done on isogeny-based hash functions. Isogeny-based hash functions have research value as they are provable hash functions, where collision resistance can be reduced to another hard problem. However, unlike commonly used hash functions such as the SHA family, the main computation involves isogeny operations, requiring further research on optimization. Also, although various countermeasures against the CGL hash function have been proposed in response to the Lauter-Petit attack, further research is needed to conduct clear security analyses for each variant and compare the efficiency of the algorithms.

In this paper, we analyze the security and performance of three isogeny-based hash functions that counter the Lauter-Petit attack: the method proposed by Panny, Zaman and Min, and Larsson. In this regard, we analyze Panny's method, which directly counters the Lauter-Petit attack. In the process, this paper presents a new isogeny-based hash function `SHH` built on Panny's approach that exploits the properties of Hessian curves. A Hessian curve, which has four fixed 3-torsion subgroups and a fixed kernel that generates backward isogeny, offers a structure well-suited for 'bit-by-bit' hash computation. We then analyze the security and performance of the proposed `SHH` in comparison to the methods by Zaman and Min and Larsson.

A. OUR CONTRIBUTIONS

The relationships between the main isogeny-based hash functions proposed to date can be illustrated in Figure 1, and these hash functions are the primary focus

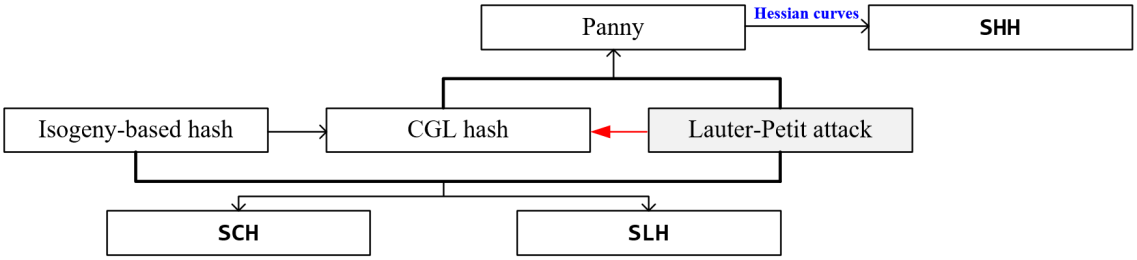


FIGURE 1. Isogeny-based hash functions.

of analysis in this paper. As shown in Figure 1, ttSCH and SLH is an isogeny-based hash function that counters the Lauter-Petit attack, but different from the CGL hash function. Panny's method can be considered a way of applying a countermeasure to the CGL hash function to counter the Lauter-Petit attack, and the SHH in this paper instantiated that idea through Hessian curves. Within this context, the contributions of this paper are summarized as follows:

- This paper proposes a new supersingular isogeny-based hash function, SHH, which leverages 3-isogenies on Hessian curves. There is currently no specific instantiation available for Panny's method. As Panny's method is notable for directly countering the Lauter-Petit attack, to compare with other isogeny-based hash functions, we employed Hessian curves, which allow for an efficient instantiation of Panny's method. Using Hessian curves, once a finite field $\mathbb{F}_{p^2}$ is established, isogeny computations on Hessian curves can be performed using only finite field operations. This eliminates the need for point arithmetic on elliptic curves, resulting in a simplified structure.
- This paper evaluates the security of isogeny-based hash functions in the context of the Lauter-Petit attack, collision resistance, and fault tolerance. We analyze that all three algorithms – SCH proposed in [6], SLH in [7], and our SHH – counter the Lauter-Petit attack. However, we identified a collision vulnerability in SCH, and analyzed the security of the point evaluation technique used in its design. Our analysis shows that the point evaluation technique can remain secure if a prime with an appropriate cofactor is selected depending on the required security level. Lastly, SHH is fault-tolerant because it uses supersingular j-invariants as hash outputs. Details are provided in Section IV.
- This paper presents the implementation result of our SHH, SCH, and SLH. In this regard, we propose parameters for each security level to implement SCH, SLH, and SHH. To the best of our knowledge, we were the first to implement SLH. As a result, the projective implementation of SLH demonstrates the best performance among the hash functions analyzed in this work, making it particularly well-suited for applications such as keyed and dynamic hashing. Details are presented in Section V.

B. ORGANIZATION

The rest of this paper is structured as follows: in Section II, we provide the mathematical background for isogenies on elliptic curves and the CGL hash function. In Section III, we review supersingular hash functions that counter the Lauter-Petit attack and introduce our hash function using Hessian 3-isogenies. Section IV presents security analysis for those hash functions, and we provide implementation techniques and results in Section V. Finally, we summarize our conclusion for this work in Section VI.

II. ISOGENIES ON ELLIPTIC CURVES

This paper considers a prime p satisfying $p \equiv 3 \mod 4$. Unless stated otherwise, K denotes the finite field $\mathbb{F}_{p^2}$ of size p^2. The mathematical backgrounds for an isogeny-based hash are outlined in Section II-A. The CGL hash function with a supersingular isogeny graph is explained in Section II-B. Section II-C introduces two examples of isogenies: Montgomery 2-isogeny and Hessian 3-isogeny.

A. ELLIPTIC CURVES AND ISOGENIES

1) ELLIPTIC CURVES

Given a prime p, an elliptic curve E over $\mathbb{F}_{p^n}$ has a group $E(\mathbb{F}_{p^n})$ of order $p^n + 1 - t_n$, where t_n is the trace of the Frobenius endomorphism and satisfies $|t_n| \leq 2\sqrt{p^n}$. If $p|t_n$, then $E/\mathbb{F}_{p^n}$ is called *supersingular*; otherwise, it is called *ordinary*. Since supersingular elliptic curves are all defined over $\mathbb{F}_{p^2}$, we assume $n = 2$ in the remainder of this paper. Two elliptic curves E and E' are *isomorphic* if there exists an $\bar{\mathbb{F}}_p$-isomorphism between them. This implies that they share the same j-invariant, meaning $j(E) = j(E')$.

2) ISOGENIES

Let E and E' be two elliptic curves over a finite field K with $|E(K)| = |E'(K)|$. Then, E and E' are said to be *isogenous* if there exists an *isogeny* $\phi : E \to E'$, where the identity of $E(K)$ is mapped onto the identity of $E'(K)$. An isogeny can be classified as either separable or inseparable, but our focus is on separable isogenies. Separable isogenies correspond one-to-one with subgroups of the group of points on a domain curve E. Given a subgroup $H \subset E(K)$, we can compute

an isogeny $\phi : E \to E'$ with the kernel H by applying the Vélu's formulas. In isogeny-based cryptography, given a finite subgroup H, the Vélu's formulas are used to perform two types of operations: *codomain computation* and *point evaluation*. Codomain computation involves determining the codomain curve E' through an isogeny $\phi : E \to E'$ with $\ker(\phi) = H$. Point evaluation involves computing $\phi(P) \in E'(K)$ for a point $P \in E(K)$.

3) SUPERSINGULAR ISOGENY GRAPHS

For given two primes p and ℓ, we can define the supersingular ℓ-isogeny graph $G(p, \ell)$ to have vertex set V, the set of j-invariants of supersingular curves over $\mathbb{F}_{p^2}$, and to have edge set E, the set of ℓ-isogenies between supersingular curves over $\mathbb{F}_{p^2}$ up to isomorphism. Then, for a prime $p > 3$ we have

$$|V| = \left\lfloor \frac{p}{12} \right\rfloor + \begin{cases} 0 & \text{if } p \equiv 1 \mod 12, \\ 1 & \text{if } p \equiv 5, 7 \mod 12, \\ 2 & \text{if } p \equiv 11 \mod 12. \end{cases}$$

Since there are $\ell + 1$ isogenies of degree ℓ, $G(p, \ell)$ is $(\ell + 1)$-regular graph. Supersingular isogeny graph has the Ramanujan property, which implies that a random walk on the graph mixes very fast. The output of a random walk on $G(p, \ell)$ tends to the uniform distribution after $O(\log |V|) = O(\log p)$ steps. Consequently, supersingular isogeny graphs are suitable for constructing hash functions.

B. THE CGL HASH FUNCTION

The CGL hash function is constructed with a supersingular isogeny graph $G(p, 2)$. Since each j-invariant has three outgoing 2-isogenies, hash functions can be naturally formulated using $G(p, 2)$. Elliptic curves $E_i : y^2 = f_i(x)$ have three 2-torsion points denoted as

$$\left(x_0^{(i)}, y_0^{(i)}\right), \left(x_1^{(i)}, y_1^{(i)}\right), \left(x_{back}^{(i)}, y_{back}^{(i)}\right).$$

To construct a hash function, we aim to compute an isogeny chain $E_0 \to E_1 \to \cdots \to E_n$, and a pair $(x_{back}^{(i)}, y_{back}^{(i)})$ represents backward 2-isogeny $E_i \to E_{i-1}$. The remaining two pairs are generators of kernels required to compute a forward 2-isogeny depending on the input message bit m_i. When the i-th bit of the message is b, the following 2-isogeny operation is applied iteratively to update the hash states.

$$\phi_i : E_i \to E_{i+1} = E_i \Big/ \left\langle (x_b^{(i)}, y_b^{(i)}) \right\rangle,$$
$$\left(x_{back}^{(i+1)}, y_{back}^{(i+1)}\right) := \phi_i \left(x_{1-b}^{(i)}, y_{1-b}^{(i)}\right).$$

Upon completing this operation, the hash state is updated as follows:

$$\left(E_i, \left(x_{back}^{(i)}, y_{back}^{(i)}\right)\right) \to \left(E_{i+1}, \left(x_{back}^{(i+1)}, y_{back}^{(i+1)}\right)\right)$$

The CGL hash function receives the n-bit message $m = (m_{n-1} \cdots m_1 m_0)_2$ as input, computes $\phi_0, \phi_1, \ldots, \phi_{n-1}$ sequentially, and the j-invariant $j(E_n)$ of E_n is the hash output.

Remark 1: It is possible to compute a 2^n-isogeny $E_0 \to E_n$ by using the 2^n-torsion point $P + [m]Q$, where P and Q are generators of $E_0[2^n]$. In [2], the authors proposed a variant of the CGL hash function that used this approach to significantly enhance the performance. However, this method may result in incorrect hash values for hexadecimal inputs. For instance, SHAKE256 produces distinct outputs for the inputs '0×01' and '0×0001'. Conversely, the isogeny with kernel $\langle P + Q \rangle$ could produce the same hash values for both inputs, effectively introducing a collision. While this issue warrants further investigation, this work does not delve into it. Instead, we adopt a 'bit-by-bit' approach, which mitigates the identified collision risks by ensuring that each bit of the input is processed separately.

C. SIMPLE REPRESENTATIONS OF SOME ISOGENIES

1) MONTGOMERY CURVES AND 2-ISOGENIES

Let K be a field with a characteristic not equal to 2 or 3. Let $a, b \in K$ such that $b \neq 0$ and $a^2 \neq 4$. The equation $M_{a,b} : by^2 = x^3 + ax^2 + x$ defines a Montgomery curve over K. Then, for given 2-torsion point $P \neq (0, 0)$ on $M_{a,b}$,

$$\phi : M_{a,b} \to M_{a',b'} : b'y^2 = x^3 + a'x^2 + x$$
$$(x, y) \mapsto (f(x), yf'(x)),$$

where

$$f(x) = x \cdot \frac{xx_P - 1}{x - x_P}, \tag{1}$$
$$a' = 2(1 - 2x_P^2),$$
$$b' = x_P b \tag{2}$$

is a 2-isogeny with $\ker(\phi) = \langle P \rangle$ [10, Proposition 2]. In this case, the kernel of dual of ϕ is generated by $(0, 0)$ on $M_{a',b'}$. Furthermore, two 2-torsion points $P = (x_P, 0)$ on $M_{a,b}$ can be computed by solving the quadratic equation $x^2 + ax + 1 = 0$, where $x_p = (-a \pm \sqrt{a^2 - 4})/2$. Thus, 2-isogenies on Montgomery curve $M_{a,b}$ can be simply represented by Montgomery coefficient a. Note that $M_{a,b}$ is denoted as M_a for convenience, since $M_{a,1}$ and $M_{a,b}$ are $\bar{K}$-isomorphic for any b.

2) HESSIAN CURVES AND 3-ISOGENIES

A Hessian curve over K is given by the cubic equation $H_d : x^3 + y^3 + 1 = dxy$, where $d \in K$ and $d^3 \neq 27$. There are four fixed subgroups of $H_d(K)$ of order 3, which generate four 3-isogenies on H_d. These 3-torsion subgroups are defined as follows:

$$G_i = \{(1 : -1 : 0), (-w^i, 0), (0, -w^i)\} \quad (i = 0, 1, 2),$$
$$G_b = \{(1 : -1 : 0), (1 : -w : 0), (1 : -w^2 : 0)\},$$

where $w \in K$ satisfies the equation $w^2 + w + 1 = 0$. From these subgroups, 3-isogenies on H_d can be defined both in affine and projective coordinates, as shown

in [11, Theorem 2].

$$\begin{aligned}\phi_i : H_d \to H_{d'} &= H_d \,/\, G_i \quad (i = 0, 1, 2)\\ (x, y)&\\ &\mapsto \left(m\frac{w^{2i}x + x^2y + w^iy^2}{xy}, m\frac{w^{2i}y + y^2x + w^ix^2}{xy}\right)\\ (X : Y : Z)&\\ &\mapsto (m(w^{2i}XZ^2 + X^2Y + w^iZY^2)\\ &\quad : m(w^{2i}YZ^2 + Y^2X + w^iZX^2) : XYZ),\end{aligned}$$

with $m = 1/\sqrt[3]{w^id^2 + 3w^{2i}d + 9}$. Then, the coefficient of the curve $H_{d'}$ is given by $d' = m(d + 6w^i)$. In this case, the dual of ϕ_i is generated as follows:

$$\begin{aligned}\hat{\phi}_i : H_{d'} \to H_{d''} &= H_{d'} \,/\, G_b \quad (i = 0, 1, 2)\\ (x, y)&\\ &\mapsto \left(m\frac{1 - wx^3 - \frac{d'xy(1-w)}{3}}{xy}, m\frac{1 - wy^3 - \frac{d'xy(1-w)}{3}}{xy}\right)\\ (X : Y : Z)&\\ &\mapsto (m(-wX^3 + Z^3 - d'XYZ(-w + 1)/3\\ &\quad : m(-wY^3 + Z^3 - d'XYZ(-w + 1)/3 : XYZ),\end{aligned}$$

with $m = \sqrt[3]{(-6w - 1)/((d')^3 - 27)}$. Then, the coefficient of the curve $H_{d''}$ is given by $d'' = d'm(w + 2)$. Since $\hat{\phi}_i$ is the dual of ϕ_i up to an isomorphism, d and d'' may not be exactly the same value, but $j(H_d) = j(H_{d''})$.

Remark 2: In the isogeny formulas of Montgomery (resp. Hessian) curves, the dual of 2-isogeny (resp. 3-isogeny) has the fixed kernel. Consider an isogeny Φ of degree 2^n (resp. 3^n), where $\Phi = \phi_n \circ \cdots \circ \phi_1$ and each $\phi_i : E_{i-1} \to E_i$ is constructed so that $\langle(0, 0)\rangle$ (resp. G_b) is not the kernel.

One may think that $\hat{\Phi}$ can be computed by continuously exploiting isogeny with $\langle(0, 0)\rangle$ (resp. G_b) as the kernel to E_n n times. For each i, when the isogeny $E_i \to E'_{i-1} = E_i/\langle(0, 0)\rangle$ (resp. E_i/G_b) is computed, E_{i-1} and E'_{i-1} are clearly isomorphic.

However, the kernels representing the backward isogeny in these curves may differ. Thus, by continuously applying this process, when E''_{i-2} is obtained from $E'_{i-1} \to E''_{i-2} = E'_{i-1}/\langle(0, 0)\rangle$ (resp. E'_{i-1}/G_b), it can frequently be observed that $j(E_{i-2}) \neq j(E''_{i-2})$. Therefore, to the best of our knowledge, the only definitive information obtainable using the fixed kernel is $j(E_{n-1})$ in this scenario.

III. SUPERSINGULAR ISOGENY HASH FUNCTIONS

First, we present two supersingular isogeny hash functions: SCH and SLH in Section III-A and Section III-B. Then, we review Panny's approach which mitigates the Lauter-Petit attack, and propose our Hessian hash function, SHH, in Section III-C.

A. SCH: HASHING WITH POINT EVALUATION

Zaman and Min proposed a hash function based on the point evaluation under the supersingular isogeny [6]. Their hash function combines bits of information from multiple message blocks, treating them as a single message block for hashing, and this hash function is referred to as a *single compression hash* (SCH). Given a point $P \in E(\mathbb{F}_{p^2})$ on the starting curve E and a message m, 2^n-isogeny $\Phi = \phi_{n-1} \circ \cdots \circ \phi_1 \circ \phi_0$ and $\Phi(P)$ are computed by the equations (1) and (2).

In the i-th step of computing the hash value, the CGL determines the kernel of ϕ_i on each bit of the message, whereas the SCH determines the kernel of ϕ_i by mixing information obtained from multiple message blocks with the evaluated point $(\phi_{i-1} \circ \cdots \phi_1 \circ \phi_0)(P)$. This process is outlined in Algorithm 1.

Algorithm 1 Single Compression Hash [6]

Input: Message $m = m_0||m_1|| \cdots ||m_{k-1}$;
$M_{a_0}/\mathbb{F}_{p^2}$, where $p = 2^{e_2}3^{e_3}5^{e_5}7^{e_7} - 1$;
$P \in M_{a_0}(\mathbb{F}_{p^2})$ for the starting curve M_{a_0} over $\mathbb{F}_{p^2}$
Output: x-coordinate of $\Phi(P)$, where Φ is 2^{e_2}-isogeny

for $i = 0$ to $e_2 - 1$ **do**
  $\mathrm{Re}(x_P), \mathrm{Im}(x_P) \leftarrow x_P$
  $x'_P \leftarrow (\mathrm{Re}(x_P) + \mathrm{Im}(x_P))\%2^{e_2}$
  $i' \leftarrow i\%k$
  $m_{i'} \leftarrow (m_{i'} + x'_P)\%2^{e_2}$
  $j \leftarrow x'_P\%(|2^{e_2} - 3^{e_3}| + |5^{e_5} - 7^{e_7}| + 1)$
  $b \leftarrow 0$
  for $l = 1$ to k **do**
    $b \leftarrow b \oplus m_l[j]$
  end for
  $C \leftarrow \sqrt{A^2 - 4}$
  $t_p \leftarrow -A + C$
  $t_m \leftarrow -A - C$
  if $b \leftarrow 1$ **then**
    $t \leftarrow \min\{t_p, t_m\}$
  else $\{b \leftarrow 0\}$
    $t \leftarrow \max\{t_p, t_m\}$
  end if
  $A \leftarrow 2 - t^2$
  $x_P \leftarrow \frac{x_P(x_Pt-2)}{2x_P-t}$
end for
return x_P

Hard computational problems of the SCH are defined as follows:

Problem 1: Given $P' = (x_{P'}, -) \in E'/\mathbb{F}_{p^2}$, with the order of the point P' is trivially given, find E'.

Problem 2: Given $P' = (x_{P'}, -) \in E'/\mathbb{F}_{p^2}$, find a sequence of 2-isogeny path from a starting curve E to the curve E' where E' is unknown.

Zaman and Min stated that these problems would not be solved quickly for sufficiently large prime p. Obviously, the isogeny path of degree 2^n can be effectively hidden if these problems are established. However, additional conditions on the detailed parameters are necessary to ensure the computational complexities of problems. Moreover, if the number of message blocks increases, it becomes possible

to find collision pairs for Algorithm 1. These issues are discussed in Section IV.

B. SLH: HASHING WITH LATTÈS MAPS

Larsson proposed a supersingular isogeny hash using Lattès maps on the projective line $\mathbb{P}$ to hide an isogeny path [7]. In the context of isogeny on elliptic curves, Lattès maps can be defined as follows.

Let $\iota : E \to E$ be a rational map, such as multiplication-by-N map $[N]$, and let $\pi : E \to \mathbb{P}$ be a morphism of varieties, such as the projection map $x(\cdot)$ to the x-coordinate. Then a Lattès maps associated with (ι, π) is a rational map $\psi : \mathbb{P} \to \mathbb{P}$ with the compositions $\psi = \pi \circ \iota \circ \pi^{-1}$. To incorporate Lattès maps into the CGL hash, Larsson sets $\iota = [2]$ and $\pi = x(\cdot)$. Then, Lattès map $\psi = x \circ [2] \circ x^{-1}$ is uniquely defined by

$$\psi(t) = x([2](t, -)), \quad t \in \mathbb{P} \text{ and } (t, -) \in E(\mathbb{F}_{p^2}).$$

For the construction of a chain of *supersingular Lattès hash* (SLH) on a Montgomery curve, the following mapping is applied.

$$\begin{aligned}
&\phi_i : M_{a_i} \to M_{a_{i+1}}, \\
&\quad \text{where } a_{i+1} = 2 - \left(-a_i + (-1)^{m_i} \cdot \sqrt{a_i^2 - 4}\right)^2, \\
&\psi_i = x \circ [2] \circ x^{-1} : \mathbb{P} \to M_{a_i} \to M_{a_i} \to \mathbb{P}, \\
&z_i := \psi_i(z_{i-1} + j(M_{a_i})), \quad 1 \le i \le n-1.
\end{aligned}$$

Here, $m = (m_{n-1} \cdots m_1 m_0)_2$ represents the binary message, and z_0 is initialized as $z_0 = \psi_0(j(M_{a_0}))$. The full computation process for SLH is detailed in Algorithm 2.

Algorithm 2 Supersingular Lattès Hash

Input: Message $m = (m_{n-1} \cdots m_1 m_0)_2$;
$M_{a_0}/\mathbb{F}_{p^2}$, where $p = 2^n f - 1$
Output: Hash value $\mathbf{z}_n$

1: $j_0 \leftarrow j(M_{a_0}) = 256(a_0^2 - 3)^3(a_0^2 - 4)^{-1}$
2: $\mathbf{z}_0 \leftarrow \texttt{xDBL\_affine}(j_0, a)$
3: **for** $i = 1$ to n **do**
4: $\quad a_i \leftarrow 2 - \left(-a_{i-1} + (-1)^{m_{i-1}} \cdot \sqrt{a_{i-1}^2 - 4}\right)^2$
5: $\quad t \leftarrow \mathbf{z}_{i-1} + j(M_{a_i})$
6: $\quad \mathbf{z}_i \leftarrow \texttt{xDBL\_affine}(t, a_i)$
7: **end for**
8: **return** $\mathbf{z}_n$

In Algorithm 2, $\texttt{xDBL\_affine}(t, a_i)$ computes $[2](t, -)$ where $(t, -) \in M_{a_i}(\mathbb{F}_{p^2})$ as follows.

$$\mathbf{z}_i = \frac{(t^2 - 1)^2}{4t(t^2 + a_i t + 1)}$$

C. SHH: HASHING WITH HESSIAN 3-ISOGENIES

A CGL-like hash function with a starting curve whose endomorphism ring is known is vulnerable to collision attack. Since those endomorphisms are divided into valid inputs, which correspond to some messages, the fact that the endomorphism ring of a starting curve is known is a threat to a CGL-like hash function.

However, Panny presented the method for constructing isogeny hash, avoiding the known endomorphism problem [8]. Panny exploits a subset of the full ℓ-isogeny graph, increasing the probability that an endomorphism on the starting curve cannot be mapped to valid inputs. Let C be the length of a cycle derived from the KLPT algorithm [9], and let r be the number of edges to be used as valid path at each vertex in $G(p, \ell)$. Then, the probability that a cycle splits valid isogeny paths is roughly computed as $C(r/\ell)^C$. Assuming the KLPT algorithm runs in time greater than $C \approx \log_\ell p$, we can estimate the cost of finding collision as follows:

$$\begin{aligned}
\frac{\text{Time for running KLPT}}{\text{Prob. of finding a valid cycle}} &\ge \frac{C}{C(r/\ell)^C} \\
&\approx \left(\frac{\ell}{r}\right)^{\log_\ell p} \\
&= (\ell^{1-\log_\ell r})^{\log_\ell p} = p^{1-\log_\ell r}.
\end{aligned}$$

Thus, the inequality $p^{1-\log_\ell r} \ge 2^\lambda$ must hold to ensure a sufficient security level λ. If we consider the generic birthday attacks as well, the prime p and (r, ℓ) should be chosen to satisfy the inequality $\log_2 p \ge \max\{2\lambda, \lambda/(1 - \log_\ell r)\}$.

In this section, we propose a *supersingular Hessian hash* (SHH) that uses 3-isogenies on a Hessian curve with a subset of 3-isogeny graphs. As denoted in Section II-C, a Hessian curve H_d has four subgroups of order 3. One of those 3-torsion subgroups, G_b, is the kernel of dual of other 3-isogenies derived from other 3-torsion subgroups G_i ($i = 0, 1, 2$). We exploit two of G_i to correspond input bits 0 and 1 to those two 3-isogenies. The computation of SHH is similar to that of the CGL hash function, and the details are provided in Algorithm 3.

Algorithm 3 Supersingular Hessian Hash

Input: Message $m = (m_{n-1} \cdots m_1 m_0)_2$;
$H_{d_0}/\mathbb{F}_{p^2}$, where $p = 3 \cdot 2^{2.71\lambda} f - 1$
Output: Hash value $j(H_{d_n})$

1: **for** $i = 0$ to $n - 1$ **do**
2: $\quad d_{i+1} \leftarrow \dfrac{d_i + 6w^{m_i}}{\sqrt[3]{d_i^2 w^{m_i} + 3 d_i w^{2m_i} + 9}}$
3: **end for**
4: **return** $j(H_{d_n})$

IV. SECURITY ANALYSIS

In this section, we present two types of collisions identified in SCH. These vulnerabilities arise due to the compression of message blocks, rather than the point evaluation technique. We then provide criteria to ensure the secure application of the point evaluation technique. Finally, we discuss the fault detection capabilities of the three hash functions.

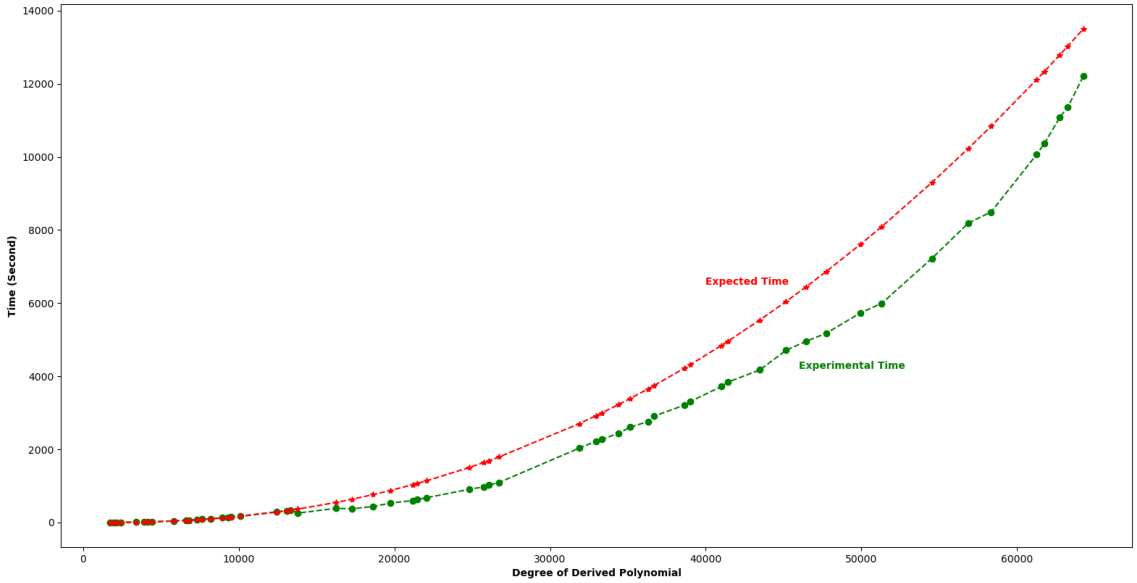


FIGURE 2. Experimental and expected time for solving polynomial derived from proposition 1.

A. COLLISIONS AGAINST SCH

Let k be the number of message blocks. This paper presents two types of collisions in SCH: the first arises from many message blocks ($k > e_2$), while the second results from a fixed b-sequence ($k \leq e_2$).

1) CASE 1. $K > E_2$

Algorithm 1 shows that the hash value for message $m = m_0||m_1||\cdots||m_{k-1}$ is computed by iterating the 2-isogeny e_2 times, assuming that each m_i is an e_2-bit message block. As denoted in steps 4 and 5 in Algorithm 1, k is involved when determining b. In the original CGL hash function, e_2k number of 2-isogenies are required to compute the hash value for the given message m. However, in SCH, only e_2 number of 2-isogenies are used. The SCH is designed to use only e_2 number of 2-isogenies by using k to determine b. Unfortunately, while this feature of SCH leads to improved performance, we can find collisions from this structure as k grows. Let two messages M_1, M_2 be defined as follows.

$$M_1 = m_0||m_1||\cdots||m_{e_2-1},$$
$$M_2 = m_0||m_1||\cdots||m_{e_2-1}||m_c||m_c.$$

In this case, the (e_2+1)-th and (e_2+2)-th messages of M_2 are not affected by step 5 in Algorithm 1, and are XORed together and canceled out at step 9 since $i \leq e_2$. Thus M_1 and M_2 are collision pairs yielding the same output.

2) CASE 2. $K \leq E_2$

In this scenario, we begin by fixing an isogeny path for a specific k. Fixing an isogeny path implies that the sequence of b exploited in [step 11-20, Algorithm 1] is known. Given this hypothesis, we can identify the intermediate tuples (M_{a_i}, x_{P_i}, b_i) and derive the following equations.

$$m_{r_l} = m_{r_l} + \sum_{t=0}^{q_l} x'_{P_{r_l+tq_l}} \quad (\max\{0, i-k+1\} \leq l \leq i),$$
$$b_i = \bigoplus_{l=0}^{k-1} m_l[j_i],$$

In the above equation, q_l and r_l are the quotient and remainder of l/k, respectively. Considering an unknown message of (e_2k)-bit, since there are at most e_2 independent equations, it is possible to find a number of messages m with (M_{a_i}, x_{P_i}, b_i) as the correct intermediate values.

From these discussions, we provide the computational cost and the implementation result of SCH in Section V, specifically for the case where $k = 1$, with the loop repeating as many times as the number of bits in the message block to achieve collision-resistance.

3) POINT EVALUATION TECHNIQUE

To mitigate the risk of the Lauter-Petit attack, SCH reveals the point evaluated by an isogeny $\phi : M_a \to M_{a'}$ as the hash value. More explicitly, SCH discloses the x-coordinate of the evaluated point. This makes it computationally difficult for an attacker to recover the curve coefficient of $M_{a'}$. Although SCH does not satisfy collision resistance as discussed in Section IV-A, disclosing the evaluated points instead of the image curves is a sufficiently valid approach. However, if the order of the evaluated point becomes known to an attacker, it becomes possible to recover the image curve $M_{a'}$ as follows.

Proposition 1: Given an f-torsion point P on a Montgomery curve $M_a/\mathbb{F}_{p^2}$, one can derive a polynomial equation of degree $O(f^2)$ from the condition $[f]P = O$. By solving

this equation, the Montgomery coefficient $a \in \mathbb{F}_{p^2}$ can be recovered.

According to the complexity presented in [26] and our experimental results shown in Figure 2, the complexity of solving equation in the Proposition 1 is upper bounded by $O(f^4 \log_2(f))$. This implies that Problem 1 can be solved if the order of the evaluated point is not sufficiently large. Therefore, to use the point evaluation technique, a prime $p = 2^n f - 1$ must satisfy $\log_2(f) \approx 0.25\lambda$ and $\log_2(p) \approx 2\lambda$ for security level λ. Note that if the order of point P used in SCH is not large enough, Problem 2 can also be solved by the discussion in Section IV-A and Proposition 1, which means that second pre-image resistance is also not satisfied.

Remark 3: At first glance, it may seem that SLH also uses the point evaluation technique, but upon examining the Algorithm 2, it becomes clear that this is not the case. Additionally, while a computation is performed on a point with the intermediate value as the x-coordinate, the order of the point is not disclosed, making this discussion inapplicable.

B. FAULT DETECTION

As hash functions are critical for ensuring data integrity, security, and system reliability, it is important to have the ability to detect faults. The use of hash functions plays a key role in cryptography, ensuring data integrity during transmission or storage. However, the occurrence of faults in the computation of hash values may result in the compromise of data integrity, underscoring the necessity for the implementation of robust mechanisms for the detection and rectification of such faults. Environmental factors, such as voltage fluctuations and electromagnetic interference, have the potential to induce transient and permanent faults in hardware implementations, including filed-programmable gate arrays (FPGAs). The absence of effective fault detection mechanisms may lead to a significant deterioration in the reliability of cryptographic systems.

In this context, the SCH and SLH have been designed to prevent the Lauter-Petit attack by avoiding the direct disclosure of the image curve. While both functions can guarantee sufficient randomness under certain conditions, they are unable to ensure that the hash value has been calculated correctly. This validation is crucial for ensuring robustness against both natural and malicious faults, particularly in hardware implementations where faults may have dire consequences.

Previous research has investigated fault detection schemes for promising hash functions, including the SHA family and BLAKE [12], [13], [14], [15]. Although it may appear challenging to apply such fault detection schemes to SCH and SLH, SHH possesses a distinctive property that enables it to detect faults with a high degree of certainty based solely on its output. In particular, the output of SHH is the j-invariant of the supersingular curve, and the probability of randomly generating a supersingular j-invariant is approximately $1/p$. Therefore, in the event of a fault, the output is likely to fail the supersingularity test. This distinctive feature enables the effective detection of faults in the computations of SHH alone.

Table 1 presents a summary of the results of the security analysis of SCH, SLH, and SHH, highlighting the comparative capabilities of these functions in addressing fault detection and their implications for cryptographic robustness.

TABLE 1. Resistance to known attacks.

Attack	SCH	SLH	SHH
Lauter-Petit	O	O	O
Section IV-A	X	O	O
Fault detection	X	X	O

V. IMPLEMENTATION

In this section, we provide our implementation details, computational costs, and implementation results of SCH, SLH, and SHH. Note that **M**, **S**, **I**, and **E** refer to field multiplication, squaring, inversion, and exponentiation, respectively. Generally, isogeny-based cryptography prefers to use Montgomery-friendly prime p, which is of the form $2^n f - 1$, and to use a supersingular elliptic curve $E/\mathbb{F}_{p^2}$ with $|E(\mathbb{F}_{p^2})| = (p+1)^2$. Since the binary form of $p+1$ contains many zeros, the hamming weight of p is approximately $\log_2 p$. Thus, in this work, we count exponentiations, such as $a^{\frac{p-3}{4}}$ and $a^{\frac{p-1}{2}}$, as $2\log_2 p$ multiplications over $\mathbb{F}_{p^2}$.

A. FINITE FIELD OPERATION

This section defines the finite field operations essential for implementing the algorithms presented in Section III. Specifically, SCH and SLH rely on 2-isogeny over Montgomery curves, while SHH uses 3-isogeny on Hessian curves. To support these operations, three additional finite field operations are required beyond basic arithmetic: *inverse*, *square root*, and *cube root*. These operations are vital for efficient isogeny computation and ensuring the security of the proposed hash functions. First, given $x = a + bi \in \mathbb{F}_{p^2}$, the inverse x^{-1} can be computed as follows.

$$x^{-1} = \frac{a - bi}{a^2 + b^2}.$$

Note that $a, b \in \mathbb{F}_p$ so that $(a^2 + b^2)^{-1}$ is computed as $\mathbb{F}_p$-exponentiation, $(a^2 + b^2)^{p-2}$. The square root operation is implemented following the algorithm described in [17, Algorithm 9], and the specific process is outlined in Algorithm 4.

In Algorithm 4, step 7 implies that a is an element over $\mathbb{F}_p$, which rarely occurs. Therefore, Algorithm 4 typically involves two $(\log_2 p)$-bit exponentiation. Finally, the cube root of elements in $\mathbb{F}_{p^2}$ can be computed as Proposition 2.

Proposition 2: Let p be a prime satisfying $p \equiv 4$ or $5 \bmod 9$. Then a cube root $\sqrt[3]{a}$ of a cubic residue $a \in \mathbb{F}_{p^2}$ can be computed as $\sqrt[3]{a} = a^{\frac{p^2+2}{9}}$.

TABLE 2. Computational costs of SCH, SLH, and SHH.

	$\log_2 p$	Costs	Approximations
SCH	2λ	$2\lambda(2.25\mathbf{E}_p + 7\mathbf{M} + 4\mathbf{S})$	$\lambda^2(18\lambda + 40.8)\mathbf{M}$
SLH	2λ	$2\lambda(2.5\mathbf{E}_p + 10\mathbf{M} + 10\mathbf{S})$	$\lambda^2(20\lambda + 72)\mathbf{M}$
SLH (Proj.)	2λ	$2\lambda(2\mathbf{E}_p + 10\mathbf{M} + 6\mathbf{S})$	$\lambda^2(16\lambda + 59.2)\mathbf{M}$
SHH	2.71λ	$2\lambda(1\mathbf{E}_{p^2} + 3\mathbf{M} + 1\mathbf{S})$	$\lambda^2(23.99\lambda + 16.82)\mathbf{M}$

Algorithm 4 Square Root Computation Over $\mathbb{F}_{p^2}$, With $p \equiv 3 \bmod 4$ [17]

Input: $a \in \mathbb{F}_{p^2}$; $i \in \mathbb{F}_{p^2}$ such that $i^2 = -1$
Output: $\sqrt{a} \in \mathbb{F}_{p^2}$ if it exists, False otherwise

$a_1 \leftarrow a^{\frac{p-3}{4}}$
$x_0 \leftarrow a_1 \cdot a$
$\alpha \leftarrow a_1 \cdot x_0$
if $a_0 = -1$ **then**
    return False
end if
if $\alpha = -1$ **then**
    return $i \cdot x_0$
else
    $b \leftarrow (1+\alpha)^{\frac{p-1}{2}}$
    return $b \cdot x_0$
end if

Proof: $a \in \mathbb{F}_{p^2}$ is a cubic residue in $\mathbb{F}_{p^2}$ if $a^{\frac{p^2-1}{3}} = 1$ [18, Theorem 8]. Let a be a cubic residue in $\mathbb{F}_{p^2}$ and $p^2 + 2$ be divisible by 9. Then, $\left(a^{\frac{p^2+2}{9}}\right)^3 = a \cdot a^{\frac{p^2-1}{3}} = a$. □ ■

From the above computations, we count the square root operation as $1\mathbf{SQRT} \approx 2E_p + 3M$ and the cube root operation as $1\mathbf{CBRT} = 1E_{p^2}$, where E_p represents $(\log_2 p)$-bit exponentiation and E_{p^2} represents $(2\log_2 p)$-bit exponentiation. Additionally, since $\mathbb{F}_{p^2}$-multiplication can be converted to four $\mathbb{F}_p$-multiplications and $\mathbb{F}_{p^2}$-inversion requires exponentiation over $\mathbb{F}_p$, we count an inverse operation as $1I = 0.25 \cdot (1E_p) + 1M + 2S$.

B. PROJECTIVE IMPLEMENTATION

In isogeny-based cryptography, projective coordinates are used to improve computational efficiency. Similarly, in isogeny-based hash functions, projective coordinates can be applied to avoid inversion operations. In this section, we examine the efficiency of applying projective coordinates to the SCH, SHH, and SLH.

1) SCH

While projective implementation of SCH is possible, it is not considered in this work. In SCH, the part where projective coordinates can be applied is the point evaluation process. SCH uses the x-coordinate of the evaluated point for selecting the direction of the isogeny path in the preprocessing step.

TABLE 3. Primes used in the implementations.

λ	Prime
128	$p_{\mathtt{SCH}} = 2^{224} \cdot 7 \cdot 67 \cdot 4578857 - 1$
	$p_{\mathtt{SLH}} = 2^{256} \cdot 3^2 \cdot 5 - 1$
	$p_{\mathtt{SHH}} = 2^{341} \cdot 3 \cdot 67 - 1$
192	$p_{\mathtt{SCH}} = 2^{336} \cdot 73 \cdot 1069 \cdot 1803471281 - 1$
	$p_{\mathtt{SLH}} = 2^{384} \cdot 7^2 \cdot 11 - 1$
	$p_{\mathtt{SHH}} = 2^{514} \cdot 3 \cdot 29 - 1$
256	$p_{\mathtt{SCH}} = 2^{448} \cdot 13 \cdot 41 \cdot 9431 \cdot 1834867773071 - 1$
	$p_{\mathtt{SLH}} = 2^{512} \cdot 3^5 - 1$
	$p_{\mathtt{SHH}} = 2^{687} \cdot 3 \cdot 157 - 1$

TABLE 4. The implementation results of each algorithm (second).

	λ		
Algorithm	128	192	256
SCH	$1.30 \cdot 10^{-1}$	$3.15 \cdot 10^{-1}$	$6.33 \cdot 10^{-1}$
SLH	$1.59 \cdot 10^{-1}$	$4.00 \cdot 10^{-1}$	$7.97 \cdot 10^{-1}$
SLH (Proj.)	$1.25 \cdot 10^{-1}$	$3.15 \cdot 10^{-1}$	$6.24 \cdot 10^{-1}$
SHH	$1.62 \cdot 10^{-1}$	$4.47 \cdot 10^{-1}$	$9.21 \cdot 10^{-1}$
SHA3	$5.72 \cdot 10^{-7}$	$8.83 \cdot 10^{-7}$	$1.14 \cdot 10^{-6}$

Hence, implementing projective coordinates would require fundamental changes to the algorithm's structure, rather than a simple adjustment. Moreover, Algorithm 1 is confirmed to be an insecure hash function as shown in Section IV-A.

2) SHH

For SHH implementation, affine coordinates are more efficient than projective coordinates. As shown in Algorithm 3, SHH relies heavily on inversion and cube root operations, which could indicate that a projective version would be more efficient. However, as demonstrated in Proposition 2, when using $p \equiv 4$ or $5 \bmod 9$, the cube root can be computed using $(2\log_2 p)$-bit exponentiation. Meanwhile, the inverse-cube root can be computed as follows.

$$a^{-\frac{1}{3}} = a^{\frac{p^2-1}{3} - \frac{p^2+2}{9}} = a^{\frac{2p^2-5}{9}},$$

which can also be computed using $(2\log_2 p)$-bit exponentiation. Applying a projective context would require computing the cube root twice, which is inefficient. Therefore, a projective implementation of SHH is not considered.

TABLE 5. Summary of isogeny-based hash functions.

	SCH [6]	SLH [7]	SHH
Idea	• Montgomery 2-isogeny • Point evaluation	• Montgomery 2-isogeny • Lattés maps	• Hessian 3-isogeny • CGL with Panny's idea [8]
Advantages	• Fast	• Fastest (projective)	• Fault-tolerant
Limitations	• Fault vulnerability • Collisions for multiple messages	• Fault vulnerability	• Slowest

3) SLH

SLH can be effectively implemented in a projective version, unlike the other two algorithms. In SLH two inversion operations are required for the j-invariant computation and xDBL_affine. These operations can be computed efficiently by using projective coordinates. Let $\mathbf{z}_i = \mathbf{Z_i}/\mathbf{D_i}$. The projective version of SLH is illustrated in Algorithm 5.

Algorithm 5 Supersingular Lattès Hash (Projective)

Input: Message $m = (m_{n-1} \cdots m_1 m_0)_2$;
$M_{a_0}/\mathbb{F}_{p^2}$, where $p = 2^n f - 1$
Output: Hash value $\mathbf{z}_n$

$J_0 \leftarrow 256(a_0^2 - 3)^3$
$K_0 \leftarrow (a_0^2 - 4)$
$(\mathbf{Z_0}, \mathbf{D_0}) \leftarrow \mathtt{xDBL}(J_0, K_0, a_o)$
for $i = 1$ to n **do**
  $a_i \leftarrow 2 - \left(-a_{i-1} + (-1)^{m_{i-1}} \cdot \sqrt{a_{i-1}^2 - 4}\right)^2$
  $T \leftarrow \mathbf{Z}_{i-1}(a_i^2 - 4) + 256\mathbf{D}_{i-1}(a_i^2 - 3)^3$
  $U \leftarrow \mathbf{D}_{i-1}(a_i^2 - 4)$
  $(\mathbf{Z}_i, \mathbf{D}_i) \leftarrow \mathtt{xDBL}(T, U, a_i)$
end for
$\mathbf{z}_n \leftarrow \frac{\mathbf{Z}_n}{\mathbf{D}_n}$
return $\mathbf{z}_n$

In Algorithm 5, $\mathtt{xDBL}(T, U, a_i)$ computes $[2](T/U, -)$ where $(T/U, -) \in M_{a_i}(\mathbb{F}_{p^2})$ as follows.

$$\mathbf{Z}_i = (T^2 - U^2)^2,$$
$$\mathbf{D}_i = 4TU(T^2 + a_i TU + U^2).$$

C. COMPUTATIONAL COSTS AND IMPLEMENTATION RESULTS

Let λ be a security level and assume that all hash algorithms described in this work are hashing on an (2λ)-bit message. The primes are set to satisfy the following conditions:

$$p_{\mathtt{SCH}} \approx 2^{n_1} f_1 - 1,$$
$$p_{\mathtt{SLH}} \approx 2^{n_2} f_2 - 1,$$
$$p_{\mathtt{SHH}} \approx 2^{n_3} \cdot 3 f_3 - 1,$$

where $\log_2(f_1) \approx 0.25\lambda$ and f_2, f_3 are small cofactors satisfying $\log_2(p_{\mathtt{SCH}}) \approx \log_2(p_{\mathtt{SLH}}) \approx 2\lambda$ and $\log_2(p_{\mathtt{SHH}}) \approx 2.71\lambda$. Based on these primes, the computational costs of the hash functions are obtained as in Table 2.

As shown in Table 2, the projective version of SLH has the lowest computational cost, while SHH has the highest. The reason is that, unlike other algorithms that use a 2λ-bit prime, SHH uses a 2.71λ-bit prime to ensure sufficient security. We estimate the total number of $\mathbb{F}_{p^2}$-multiplications, considering the cost of finite field arithmetic, and these approximations are described in Table 2.

To evaluate the performance, we implemented the algorithms using SageMath. SageMath was used for the finite field operations. The elliptic curve arithmetic and isogeny computations were implemented manually and used commonly across all three algorithms. In each instance of implementation, the prime was selected in accordance with the required security level λ, as illustrated in Table 3.

The performances of our implementations are evaluated on the Intel Core i7-12700K at 3.60 GHz, running Ubuntu 22.04.2 LTS. Our implementation results and the comparison with SHA3 are presented in Table 4.

As denoted in Table 4, the projective implementation of SLH demonstrates the fastest performance, while SHH exhibits the slowest. These results align closely with the cost estimations presented in Table 2.

Since all algorithms use Montgomery-friendly prime, overall performance can be improved by implementing finite field arithmetic using it. However, as shown in Table 4, it does not yet show sufficient performance for actual use, thus future work will be needed. Summarizing the result, Table 5 describes the security and performance results of this paper.

Remark 4: Should a projective version of SCH *be proposed, the number of multiplications may be slightly less than that of* SLH*. This is due to the fact that the cost of point evaluation of Montgomery 2-isogeny is slightly cheaper than Montgomery point doubling. However, as per Section IV-A3, it is anticipated that projective* SLH *will ultimately prove to be the most efficient, given that* SLH *utilizes more efficient Montgomery-friendly primes.*

VI. CONCLUSION

In this paper, we analyzed the security of previously proposed isogeny-based hash functions SCH and SLH, and proposed a new supersingular isogeny hash function SHH, based on 3-isogenies on Hessian curves.

SCH utilizes a point evaluation technique and a single compression method to ensure the randomness of the results.

According to our analysis, it counters the Lauter-Petit attack by not revealing the supersingular *j*-invariant. However, our analysis also revealed that while SCH's single compression method offers the advantage of improved performance when hashing multiple blocks, it also introduces a vulnerability that could allow collisions to be found through other methods. In this paper, we provided guidelines for selecting parameters to securely use SCH.

SLH uses Lattés maps to efficiently compute hash values, enhancing the CGL hash with elliptic curve multiplications. The security analysis of SLH in this work confirmed that it is collision-resistant. Lastly, our SHH exploits the strategy proposed in [8] that prevents valid collisions by limiting the use of edges in the supersingular isogeny graph to resist the Lauter-Petit attack. SHH takes advantage of the efficiency of computing 3-isogeny chains on Hessian curves.

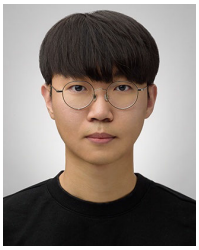
The implementation results show that isogeny-based hash functions still lack practical aspects, which is also a characteristic of provable hash functions when compared to dedicated hash functions. However, as the development of a quantum computer becomes visible, research on the efficiency of quantum-safe hashes is important in provable hash area, and this research will serve as a foundation for future developments. Furthermore, future research will be conducted to enhance performance through Montgomery-friendly implementation of finite field operations and low-level programming. There will also be investigations to determine the potential applications of these hash functions in various domains and how to apply fault detection.

REFERENCES

[1] D. X. Charles, K. E. Lauter, and E. Z. Goren, "Cryptographic hash functions from expander graphs," *J. Cryptol.*, vol. 22, no. 1, pp. 93–113, Jan. 2009.
[2] J. Doliskani, G. C. C. F. Pereira, and P. S. L. M. Barreto, "Faster cryptographic hash function from supersingular isogeny graphs," in *Proc. Int. Conf. Sel. Areas Cryptograph.* Halifax, U.K.: Springer, 2022, pp. 399–415.
[3] C. Petit and K. Lauter, "Hard and easy problems for supersingular isogeny graphs," Cryptol. ePrint Arch., Tech. Rep. 2017/962, 2017.
[4] K. Eisenträger, S. Hallgren, K. Lauter, T. Morrison, and C. Petit, "Supersingular isogeny graphs and endomorphism rings: Reductions and solutions," in *Proc. 37th Annu. Int. Conf. Theory Appl. Cryptograph.* Tel Aviv, Israel. Cham, Switzerland: Springer, 2018, pp. 329–368.
[5] J. Booher, R. Bowden, J. Doliskani, T. Boris Fouotsa, S. D. Galbraith, S. Kunzweiler, S.-P. Merz, C. Petit, B. Smith, K. E. Stange, Y. B. Ti, C. Vincent, J. F. Voloch, C. Weitkämper, and L. Zobernig, "Failing to hash into supersingular isogeny graphs," *Comput. J.*, vol. 67, no. 8, pp. 2702–2719, Aug. 2024.
[6] M. U. Zaman and M. Min, "Supersingular isogeny-based single compression cryptographic hash function," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, Dec. 2023, pp. 4479–4484.
[7] D. Larsson, "Supersingular hashing using Lattès maps," Cryptol. ePrint Arch., Tech. Rep. 2024/539, 2024.
[8] L. Panny, "Isogeny-based hashing despite known endomorphisms," Cryptol. ePrint Arch., Tech. Rep. 2019/927, 2019.
[9] D. Kohel, K. Lauter, C. Petit, and J.-P. Tignol, "On the quaternion-isogeny path problem," *LMS J. Comput. Math.*, vol. 17, no. A, pp. 418–432, 2014.
[10] J. Renes, "Computing isogenies between Montgomery curves using the action of (0, 0)," in *Proc. Int. Conf. Post-Quantum Cryptograph.*, Fort Lauderdale, FL, USA. Cham, Switzerland: Springer, 2018, pp. 229–247.
[11] F. L. P. Broon and E. Fouotsa, "Analogue of Vlu's formulas for computing isogenies over Hessian model of elliptic curves," Cryptol. ePrint Arch., Tech. Rep. 2019/1480, 2019.
[12] M. M. Kermani, S. Bayat-Sarmadi, A.-B. Ackie, and R. Azarderakhsh, "High-performance fault diagnosis schemes for efficient hash algorithm Blake," in *Proc. IEEE 10th Latin Amer. Symp. Circuits Syst. (LASCAS)*, Feb. 2019, pp. 201–204.
[13] F. Kahri, H. Mestiri, B. Bouallegue, and M. Machhout, "An efficient fault detection scheme for the secure hash algorithm SHA-512," in *Proc. Int. Conf. Green Energy Convers. Syst. (GECS)*, Mar. 2017, pp. 1–5.
[14] F. Kahri, H. Mestiri, B. Bouallegue, and M. Machhout, "High performance and reliable fault detection scheme for the secure hash algorithm," *Indian J. Sci. Technol.*, vol. 10, no. 19, pp. 1–9, Feb. 2017.
[15] A. Torres-Alvarado, L. A. Morales-Rosales, I. Algredo-Badillo, F. López-Huerta, M. Lobato-Báez, and J. C. López-Pimentel, "An SHA-3 hardware architecture against failures based on Hamming codes and triple modular redundancy," *Sensors*, vol. 22, no. 8, p. 2985, Apr. 2022.
[16] I. Gavrilan, F. Oberhansl, A. Wagner, E. Strieder, and A. Zankl, "Impeccable Keccak: Towards fault resilient SPHINCS+ implementations," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2024, no. 2, pp. 154–189, Mar. 2024.
[17] G. Adj and F. Rodríguez-Henríquez, "Square root computation over even extension fields," *IEEE Trans. Comput.*, vol. 63, no. 11, pp. 2829–2841, Nov. 2014.
[18] M. S. Mohamad, "An algorithms for finding the cube roots in finite fields," *Proc. Comput. Sci.*, vol. 179, pp. 838–844, Jan. 2021.
[19] I. B. Damgård, "A design principle for hash functions," in *Proc. Int. Cryptol. Conf.*, 1989, pp. 416–427.
[20] R. C. Merkle, "A certified digital signature," in *Proc. Conf. Theory Appl. Cryptol.*, 1989, pp. 218–238.
[21] X. Y. Wang, Y. Q. Yin, and H. B. Yu, "Finding collisions in the full SHA-1," in *Proc. Annu. Int. Cryptol. Conf.*, in Lecture Notes in Computer Science, vol. 3621, 2005, pp. 17–36.
[22] D. Augot, M. Finiasz, and N. Sendrier, "A fast provably secure cryptographic hash function," Cryptol. ePrint Arch., Tech. Rep. 2003/230, 2003.
[23] S. Contini, A. K. Lenstra, and R. Steinfeld, "VSH, an efficient and provable collision-resistant hash function," in *Proc. 24th Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, vol. 4004, Saint Petersburg, Russia. Berlin, Germany: Springer, May 2006, pp. 165–182.
[24] D. Jao and L. De Feo, "Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies," in *Proc. 4th Int. Workshop PQCrypto*. Cham, Switzerland: Springer, 2011, pp. 19–34.
[25] Y. Huang, F. Zhang, Z. Liu, and H. Zhang, "An efficient signature scheme from supersingular elliptic curve isogenies," *IEEE Access*, vol. 7, pp. 129834–129847, 2019.
[26] M.-H. Kim and S. Sutherland, "Polynomial root-finding algorithms and branched covers," *SIAM J. Comput.*, vol. 23, no. 2, pp. 415–436, Apr. 1994.

DONGHOE HEO received the B.A. degree in mathematics from Hanyang University, in 2019. He is currently pursuing the Ph.D. degree with the Graduate School of Information Security, Korea University, with a focus on post-quantum cryptography, especially isogeny-based cryptosystems, and quantum algorithms.

JEONGHWAN LEE received the B.A. degree in artificial intelligence cyber security from Korea University, in 2023, where he is currently pursuing the M.S. degree with the Graduate School of Information Security, with a focus on post-quantum cryptography, especially isogeny-based cryptosystems, and side-channel analysis.



TAEHOON KANG received the B.A. degree in mathematics from Korea University, in 2023, where he is currently pursuing the M.S. degree with the Graduate School of Information Security, with a focus on post-quantum cryptography, especially isogeny-based cryptosystems, and quantum algorithms.

SUHRI KIM received the B.A. degree in mathematics and the M.A. degree in information security from Korea University, in 2014 and 2016, respectively, and the Ph.D. degree from the Graduate School of Information Security, Korea University, in 2020. She is currently working as an Associate Professor with the School of Mathematics, Statistics, and Data Science, Sungshin Women's University. Her research interests include post-quantum cryptography and efficient computations for isogeny-based cryptosystems.

SEOKHIE HONG received the M.S. and Ph.D. degrees in mathematics from Korea University, in 1997 and 2001, respectively. He worked at Security Technologies Inc., from 2000 to 2004. Subsequently, he conducted postdoctoral research with COSIC, KU Leuven, Belgium, from 2004 to 2005, after which he joined the Graduate School of Cyber Security, Korea University. His research interests include cryptography, public and symmetric cryptosystems, hash functions, and MACs.