Isogeny-based hashing despite known endomorphisms

Lorenz Panny

Department of Mathematics and Computer Science,
Technische Universiteit Eindhoven, Netherlands
lorenz@yx7.cc

Abstract. The Charles–Goren–Lauter hash function on isogeny graphs of supersingular elliptic curves was shown to be insecure under collision attacks when the endomorphism ring of the starting curve is known. Since there is no known way to generate a supersingular elliptic curve with verifiably unknown endomorphisms, the hash function can currently only be used after a trusted-setup phase. This note presents a simple modification to the construction of the hash function which, under a few heuristics, prevents said collision attack and permits the use of arbitrary starting curves, albeit with a performance impact of a factor of two.

Keywords: Isogeny-based cryptography, expander graphs, hash functions.

1 Introduction

In 2009, Charles, Goren, and Lauter described a construction of cryptographic hash functions from certain superpolynomially-sized graphs [1], essentially by mapping the input message to a walk in the graph in an injective manner and returning a representation of the walk's end node as the hash value; cf. Figure 1. For security, it is required that the graph is strongly interconnected (fast mixing), as well as that finding non-trivial cycles in the graph is hard.

As an example instantiation, [1] proposed using ℓ-isogeny graphs of supersingular elliptic curves, which provably have the fast mixing property [8], but unfortunately Petit and Lauter later showed that cycle finding is easy when the endomorphism ring of the starting node is known [7, 4]. Since there is no known method to generate a supersingular elliptic curve in a verifiably pseudorandom way such that nobody can know the endomorphism ring, this means the CGL supersingular-isogeny hash function is only usable after a setup phase in which a trusted entity (which might be a multi-party computation) takes a random walk to generate a starting curve, then makes sure to forget the path that led there.

In Section 2, we introduce and analyze a slight tweak to the construction of the hash function that seems to thwart the Lauter–Petit attack without requiring that the endomorphism ring be unknown.

* This work was supported in part by the Commission of the European Communities through the Horizon 2020 program under project number 643161 (ECRYPT-NET). Date of this document: 2019.08.14.

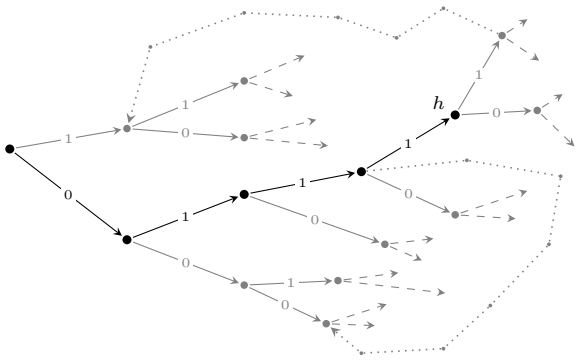

Figure 1. A graph-based hash function consuming the message `0111`. The hash value is the identifier of the end node h of the walk. The gray edges exemplify a much bigger ambient graph (containing lots of big cycles, which are supposed to be hard to find).

1.1 The Charles–Goren–Lauter hash function [1]

The construction of the supersingular-isogeny CGL hash function follows easily from the general picture and some specific properties of supersingular isogeny graphs: For prime ℓ, the ℓ-isogeny graph is a $(\ell+1)$-regular undirected graph almost everywhere (save ≤ 2 exceptional nodes). Charles, Goren, and Lauter suggest setting $\ell = 2$, such that each node has three outgoing edges, and consume one input bit at a time to deterministically select one of the two out-edges distinct from the edge that was taken to arrive at the current node.

More formally, at each point in time, the state of the hash function consists of a pair (E, x_{back}) where $E\colon y^2 = x^3+ax+b$ is a supersingular elliptic curve and x_{back} is the x-coordinate of a point of order 2 on E that leads back to the previous node on the walk and must be avoided. Consuming a bit b proceeds as follows: Find the two roots the polynomial $g(x) = (x^3 + ax + b)/(x - x_{back})$; they are the x-coordinates of the remaining points of order 2 on E. Use an arbitrary, but fixed, deterministic ordering on the two roots of $g(x)$ to assign them the labels x_0 and x_1, and let $P_i = (x_i, 0)$ be the corresponding points on E. Compute the isogeny $\varphi\colon E \to E/\langle P_b\rangle$ and update the state to $(E/\langle P_b\rangle, x(\varphi(P_{1-b})))$. Once all input bits have been consumed, the curve E from the state (or a shorter value derived from it) is returned.

Remark 1. In the typical setting $p \equiv 3 \pmod 4$ and $E_0\colon y^2 = x^3 + x$, the initial state needs to be chosen with caution: Since E_0 admits the nontrivial automorphism $\iota\colon (x, y) \mapsto (-x, \sqrt{-1}\cdot y)$ of order 4, there is (up to isomorphism) only one curve 2-isogenous to E_0 besides itself, which leads to easy collisions if not addressed properly. Luckily, it seems that initializing the hashing process by taking the isogeny $E_0 \to E_0/\langle(\sqrt{-1}, 0)\rangle$ before starting to consume input bits gets around this issue.

1.2 The Lauter–Petit attack [7, 4]

We now outline the collision attack against the CGL hash function when the starting curve has known endomorphism ring $\mathcal{O} = \mathrm{End}(E_0)$. Recall that $\mathcal{O}$ is an order in a quaternion algebra since E_0 is supersingular.

First, notice that hash collisions correspond to cycles in the ℓ-isogeny graph containing E_0, which in turn correspond to endomorphisms of E_0 of ℓ-power norm. Finding an endomorphism of ℓ-power norm consists of solving a norm equation coming from the known description of $\mathcal{O}$, but in general the kernel subgroup of such an endomorphism (so as to iteratively compute the curves on the corresponding cycle) is impossible to write down without passing to prohibitively large extension fields.

This is where the second observation comes into play: Isogenies $E_0 \to E$ correspond (up to isomorphism of E) to left-ideals of $\mathcal{O}$, and (since principal ideals correspond to endomorphisms) the isogeny codomains corresponding to two ideals are isomorphic if and only if the ideals are equivalent. Algorithmically speaking, this allows one to replace the ideal of norm ℓ^n by an equivalent ideal of powersmooth norm, which keeps the extension degrees under control and thereby makes computing the codomain curve feasible [5].

Now finally, for any factorization $\varphi = \psi' \circ \psi$ of a cyclic isogeny $\varphi\colon E_0 \to E$, it is easy to compute the ideal corresponding to ψ from the degree of ψ and the ideal corresponding to φ. In particular, this implies that knowing the ideal $\mathfrak{a} \subseteq \mathcal{O}$ of a cyclic ℓ^n-isogeny $\varphi = \psi_n \circ \cdots \circ \psi_1$, where each ψ_i has degree ℓ, one can efficiently compute the ideals corresponding to the partial isogenies ψ_1, $\psi_2 \circ \psi_1$, etc., up to $\psi_n \circ \cdots \circ \psi_1$. Concretely, the left ideal corresponding to the partial isogeny $\psi_k \circ \cdots \circ \psi_1$ simply equals the sum $\mathfrak{a} + \mathcal{O}\ell^k$.

With these ingredients, the Lauter–Petit collision attack proceeds as follows: Solve a norm equation to find an endomorphism $\alpha \in \mathcal{O}$ of norm ℓ^n for some n. For each k from 1 to $n-1$, compute the ideal $\mathcal{O}\alpha + \mathcal{O}\ell^k$ corresponding to the first k steps in the ℓ-isogeny cycle given by α. Using an algorithm of Kohel, Petit, Lauter, and Tignol [6], transform each of these ideals into an equivalent ideal of powersmooth norm and compute the corresponding codomain curve. At this point, one has a list of curves $(E_0, E_1, \ldots, E_{n-1}, E_n = E_0)$ such that each pair (E_i, E_{i+1}) is connected by an ℓ-isogeny. In the original CGL hash function as described above, this cycle corresponds to a collision, by choosing the input bits in such a way that the edges taken are exactly the ℓ-isogenies on the cycle.

2 Hardened variant of the hash function

Our main contribution is the following simple idea:

> *Use only a fraction of the available edges at each step.*

The remainder of this note will be devoted to the analysis of this approach.

As a starting point, note that the Lauter–Petit attack outlined above relies crucially on the assumption that all (or a significant portion) of the ℓ-isogeny

cycles in the graph yield a collision. This makes use of the fact that (for $\ell = 2$), except at the starting node, there are exactly two outgoing edges at each step, corresponding to consuming the input bits 0 and 1. In other words, essentially all paths starting at E_0 are reachable by twiddling with the input bits.

We will now analyze a variant of this construction, where instead of setting $\ell = 2$ and consuming one bit per step, we shall leave ℓ variable and suppose that we use the input to select one of r outgoing cyclic ℓ-isogenies at each step. By counting cyclic subgroups of $E[\ell] \cong \mathbb{Z}/\ell \times \mathbb{Z}/\ell$, one sees that (even for composite degrees ℓ) the number of cyclic ℓ-isogenies emerging from any given node is lower bounded by $\ell + 1$. Thus the probability that a given cycle of length C yields a collision can very roughly be estimated as $\leq (r/\ell)^C$, as all edges along the way need to be 'valid', but see below.

Note that there are two opposing trends when increasing ℓ: On one hand, the quotient r/ℓ shrinks, which decreases the chance that a given cycle leads to a collision, but on the other hand, the expected cycle length in the graph shrinks as well (since there are more edges), *in*creasing the chance. Luckily, we will see that it is possible to balance these effects.

Finally, note that all of this requires the heuristic assumption that the cycles obtained from the Lauter–Petit attack are more or less random: If, for instance, it turns out that one can pick the endomorphism α in a smart way to increase the probability of finding a collision, the estimates below are clearly void.

2.1 Analysis

We now estimate the chance that a 'random' ℓ-isogeny cycle yields a collision.

Expected cycle lengths. As hinted above, we require an estimate on minimum length of cycles in the ℓ-isogeny graph. By basic counting, there are $(\ell+1) \cdot \ell^{k-1}$ paths of length k. Therefore, by the birthday paradox, it can be expected that two such paths meet once approximately a square-root fraction of the $\approx p/12$ nodes has been covered, that is, after roughly $\frac{1}{2} \log_\ell p$ steps. The concatenation of two such paths thus gives rise to a cycle of length $\log_\ell p$.

Remark 2. For some curves, there are evident endomorphisms of very small ℓ-power norm: For instance, whenever $\mathrm{End}(E_0)$ contains a square root of -1, such as in the typical case $p \equiv 3 \pmod 4$ and $j(E_0) = 1728$, the endomorphism $1 + \sqrt{-1}$ of E_0 has norm 2. In such cases, care needs to be taken to avoid collisions coming from these special, exceptionally short cycles, for instance by forcing the first steps away from 'dangerous' edges.

Chance of 'good' cycles. Naïvely, the chance that a cycle yields a collision looks like $(r/\ell)^C$, corresponding to a chance of r/ℓ each that an edge on the cycle can be taken by the hash function. However, in reality, this is slightly more complicated: The cycle can be traversed in both directions, and any way of splitting the cycle into two distinct paths to the same node needs to be considered.

Although it is not difficult to compute this exactly, it is fairly obvious that the increase in the attacker's success probability due to this degree of freedom is upper bounded by the cycle length C, which will be a good enough estimate in the following.

Longer cycles? An attacker may opt to try out cycles (much) longer than the expected minimum length $\log_\ell p$, but since the success probability is exponentially small in the cycle length, this strategy seems to be inferior to minimizing cycle lengths and will thus be disregarded in the following.

Incremental cycle finding? Our estimates below make use of the assumption that the best an attacker can do is pick a random endomorphism, corresponding to an isogeny cycle, and hoping that it leads to a collision. However, it is possible to modify the KLPT algorithm to adaptively tweak the chosen endomorphism when the isogeny walk gets stuck due to a disallowed edge, as to obtain another continuation of the path that has already been started. One may hope that repeating this corrective step sufficiently often may lead to an admissible closed ℓ-isogeny walk. We now estimate the effectiveness of this approach, for concreteness focusing on a suborder $\mathcal{O}$ of $\mathrm{End}(E_0)$ with norm form $a^2 + b^2 + pc^2 + pd^2$. Such a subring exists in the common case $E_0 \colon y^2 = x^3 + x \,/\mathbb{F}_{p^2}$ with $p \equiv 3 \pmod 4$.

Suppose the attacker has started out with an endomorphism α of ℓ-power norm $a^2 + b^2 + pc^2 + pd^2$ and gets stuck after k steps in the ℓ-isogeny graph. They wish to compute a new endomorphism $\alpha' \neq \alpha$ of ℓ-power norm such that $\mathcal{O}\alpha' + \mathcal{O}\ell^k = \mathcal{O}\alpha + \mathcal{O}\ell^k$, which can be done by solving the norm equation

$$a'^2 + b'^2 + pc'^2 + pd'^2 = \ell^{e'}$$

subject to the constraints $a' \equiv a \pmod{\ell^k}$, etc., with small e'. Estimating the number of choices for (a', b', c', d', e') for a given upper bound $e' \leq M$, we get $\#a', \#b' \approx \sqrt{\ell^M}/\ell^k$, $\#c', \#d' \approx \sqrt{\ell^M/p}/\ell^k$, and $\#e' \approx M$. Hence the total number of choices is $\approx \ell^{2M-4k}M/p$. A single choice has a heuristic chance of $\approx 1/\ell^M$ to satisfy the equation, hence there should be solutions roughly when $\ell^{M-4k}M/p \geq 1$; thus we can expect $e' + \log_\ell e' \geq 4k + \log_\ell p$. Note that we will want ℓ to be very large (see Table 1), hence this implies roughly $e' \geq 4k + \log_\ell p$. Since $e' - k$ is expected to be bigger than the original expected cycle length $\log_\ell p$, it seems that this approach will only make the attacker's life (even) harder.

Expected success probability. In summary, the probability that an attacker can transform a given 'random' isogeny cycle into two colliding hash function inputs is coarsely upper bounded by

$$C \cdot \left(\frac{r}{\ell}\right)^C.$$

Translating this into the amount of work for an attacker, we recall that we can assume $C \approx \log_\ell p$ and point out that *finding* a cycle using the KLPT algorithm

surely takes time $\Omega(\log p) \geq C$, hence we can estimate the total attack cost as

$$\left(\frac{\ell}{r}\right)^C \approx \left(\frac{\ell}{r}\right)^{\log_\ell p} = \left(\ell^{1-\log_\ell r}\right)^{\log_\ell p} = p^{1-\log_\ell r}\,.$$

For any target security level λ, one therefore obtains a tradeoff curve between the size of p and the relative sizes of ℓ and r. For example, when $r = 2$, such that one bit is consumed at a time, one requires the prime p to be about 2.71λ bits long when $\ell = 3$, about 2λ bits long when $\ell = 4$, about $3\lambda/2$ bits when $\ell = 8$, and about $4\lambda/3$ bits when $\ell = 16$, to (conjecturally) achieve λ-bit security against this specific attack.

Note that for $\log_\ell r \leq 1/2$, the Petit–Lauter attack against this variant is in fact more expensive than generic birthday attacks against the hash function, hence whenever $r^2 \leq \ell$ the cycle-finding approach can be neglected. In summary, to protect the CGL hash function against both the Petit–Lauter attack as well as generic collision finding, the choice of $(p, \log_\ell r)$ should satisfy

$$\log_2 p \geq \max\{2\lambda, \lambda/(1-\log_\ell r)\} =: \pi_\lambda(\log_\ell r)\,. \tag{1}$$

Optimizing parameter choices. Clearly, larger p and smaller $\log_\ell r$ have a negative impact on the performance of the hash function. Following [3], it is beneficial to pick the pair (p, ℓ) in such a way that $\ell = 2^n$ is a big power of two and $p = 2^n \cdot f - 1$, where f is a small cofactor. With the starting curve $E_0 \colon y^2 = x^3 + x$, these choices ensure that at each step of the hash function a big chunk of the input message can be encoded injectively, but not necessarily surjectively, as a cyclic $\mathbb{F}_{p^2}$-rational subgroup of size 2^n, whose corresponding isogeny can be evaluated efficiently using the tree-based strategy of [2] using $O(n \log n)$ operations.

While the exact cost of arithmetic in $\mathbb{F}_{p^2}$ depends rather strongly on specific implementation details (e.g., word-size boundaries), we can roughly model the growth in complexity as $(\log p)^2$, which corresponds to the cost of schoolbook arithmetic. Therefore, for a fixed choice of $(\log p, \ell, r)$, we can expect the cost per bit of evaluating our modified hash function to roughly scale as

$$(\log p)^2 (n \log n)/\log r \approx (\log p)^2 (\log\log p)/\log_\ell r =: C(\log p, \log_\ell r)\,.$$

The objective is therefore to minimize this function while respecting the constraint $\log_2 p \geq \pi_\lambda(\log_\ell r)$ from (1) for a certain security level λ. Clearly, for fixed $\log_\ell r$, it is optimal to choose p as small as possible, i.e., $\log_2 p \approx \pi_\lambda(\log_\ell r)$. See Figure 2 for plots of C restricted to this choice of $\log_2 p$ for various values of λ; it is evident that $\log_\ell r = 1/2$ minimizes the cost. Based on this, Table 1 lists some parameter choices expected to work well for the case where ℓ and r are (following [3]) both powers of two, for some typical target security levels λ.

Since we only consume $\log_\ell r \approx$ half of the possible input bits at each step, but can make use of all existing optimizations otherwise, we expect the performance of our modified hash function to be very close to half the speed of the optimized CGL hash function variant from [3].

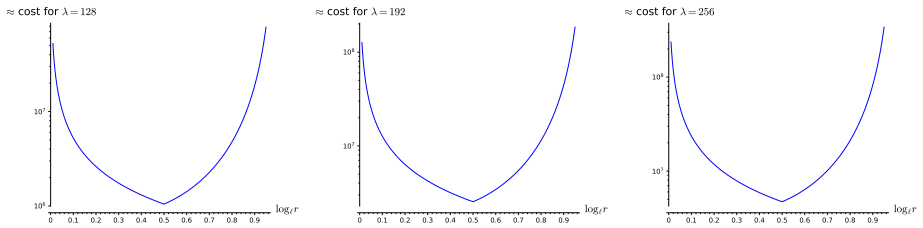

Figure 2. Plots of the cost estimate $C(2^{\pi\lambda(\log_\ell r)}, \log_\ell r)$ for $\lambda \in \{128, 192, 256\}$. The exact values are meaningless, but notice the clear-cut global minima at $\log_\ell r = 1/2$.

Table 1. Reasonable choices of (p, ℓ, r) for conjectured security levels λ.

λ	p	ℓ	r
128	$2^{256} \cdot 45 - 1$	2^{256}	2^{128}
192	$2^{391} \cdot 3 - 1$	2^{390}	2^{195}
256	$2^{512} \cdot 243 - 1$	2^{512}	2^{256}

References

[1] Denis X. Charles, Kristin E. Lauter, and Eyal Z. Goren. Cryptographic hash functions from expander graphs. *J. Cryptology*, 22(1):93–113, 2009. https://ia.cr/2006/021.

[2] Luca De Feo, David Jao, and Jérôme Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *J. Mathematical Cryptology*, 8(3):209–247, 2014. https://ia.cr/2011/506.

[3] Javad Doliskani, Geovandro C. C. F. Pereira, and Paulo S. L. M. Barreto. Faster cryptographic hash function from supersingular isogeny graphs, 2017. IACR Cryptology ePrint Archive 2017/1202. https://ia.cr/2017/1202.

[4] Kirsten Eisenträger, Sean Hallgren, Kristin E. Lauter, Travis Morrison, and Christophe Petit. Supersingular isogeny graphs and endomorphism rings: Reductions and solutions. In *EUROCRYPT*, volume 10822 of *LNCS*, pages 329–368. Springer, 2018. https://ia.cr/2018/371.

[5] Steven D. Galbraith, Christophe Petit, and Javier Silva. Identification protocols and signature schemes based on supersingular isogeny problems. In *ASIACRYPT*, volume 10624 of *LNCS*, pages 3–33. Springer, 2017. https://ia.cr/2016/1154.

[6] David Kohel, Kristin E. Lauter, Christophe Petit, and Jean-Pierre Tignol. On the quaternion ℓ-isogeny path problem, 2014. IACR Cryptology ePrint Archive 2014/505. https://ia.cr/2014/505.

[7] Christophe Petit and Kristin E. Lauter. Hard and easy problems for supersingular isogeny graphs, 2017. IACR Cryptology ePrint Archive 2017/962. https://ia.cr/2017/962.

[8] Arnold K. Pizer. Ramanujan graphs and Hecke operators. *Bull. Amer. Math. Soc. (N.S.)*, 23(1):127–137, 07 1990.