Faster Cryptographic Hash Function From Supersingular Isogeny Graphs

Javad Doliskani, Geovandro C. C. F. Pereira and
Paulo S. L. M. Barreto

Abstract. We propose a variant of the CGL hash [5] that is significantly faster than the original algorithm, and prove that it is preimage and collision resistant. For $n = \log p$ where p is the characteristic of the finite field, the performance ratio between CGL and the new proposal is $(5.7n + 110)/(13.5 \log n + 46.4)$. This gives an exponential speed up as the size of p increases. Assuming the best quantum preimage attack on the hash has complexity $O(p^{\frac{1}{4}})$, we attain a concrete speed-up for a 256-bit quantum preimage security level by a factor 33.5. For a 384-bit quantum preimage security level, the speed-up is by a factor 47.8.

Keywords. Cryptographic hash functions, Supersingular elliptic curves, Isogeny graphs, Expander graphs.

2010 Mathematics Subject Classification. 94A60, 14K02, 11Y16.

1 Introduction

A provably secure hash function is a hash function in which finding collisions is efficiently reducible from a computationally hard problem. The first proposals for provably secure hash functions were based on number theoretic problems such as integer factorization and discrete logarithm which are widely believed to be hard. The Very Smooth Hash (VSH) proposed by Contini et al. [8] is a provably secure hash algorithm based on an assumption related to integer factorization. The idea behind VSH is similar to the one introduced in the earlier work of Chaum [6] on undeniable signatures. A variant of VSH, called VSH-DL, is based on a problem related to the discrete logarithm problem. VSH is very fast and can be used in schemes like the Cramer-Shoup signature [11] to improve the performance without sacrificing any security.

Security of the schemes based on these classical number theoretic problems, however, is threatened by the emergence of quantum computers. A quantum com-

This work was partially supported by NSERC, CryptoWorks21, and Public Works and Government Services Canada.

puter can perform the *Fourier Transform* on an exponential number of amplitudes in polynomial time [7, 24]. This leads to polynomial time quantum algorithms for phase estimation and order-finding, and consequently factoring and computing discrete logarithms [29, 31].

Modern provably secure hash functions are based on less standard assumptions but are believed to resist quantum attacks. Inspired by Ajtai's seminal work [1] on average-case to worst-case reduction of standard lattice problems, Micciancio [21] proposed an efficient hash function whose security is based on certain approximation problems on ideal lattices. A more efficient variant of Micciancio's hash function, called SWIFFT, was later proposed by Lyubashevsky et al. [18, 19]. SWIFFT is very efficient, with performance comparable to SHA-256 and has good statistical properties. It, however, cannot be used as a pseudorandom function since it preserves addition [19, §4.3].

A class of provably secure hash functions are based on expander graphs. An expander graph is, informally, a graph with low degree and high connectivity. The use of expander graphs for hashing started with the works of Zémor and Tillich [35, 40, 41] on particular expander graphs called Cayley graphs. In 2009, Charles et al. [5] proposed an expander hash, called CGL, which is based on the isogeny graph of supersingular elliptic curves over finite fields. Supersingular isogeny graphs are excellent expander graphs with asymptotically optimal expansion constant [26]. The security of CGL is based on the hardness of computing isogenies of large degree between supersingular elliptic curves. Since the introduction of CGL, supersingular isogeny problems have attracted considerable attention in cryptography, and the best known attacks on them have exponential complexity. The main drawback of CGL is efficiency. For a finite field of characteristic p, the algorithm requires roughly $2 \log p$ modular multiplications per bit of the input. This makes CGL far less efficient than other provably secure hash algorithms.

Our contributions. We exploit primes of the form $p = 2^n f \pm 1$, where $f > 0$ is a small integer, as the characteristic of the finite field. Instead of consuming a bit of the input at a time, we use a block of length $n \approx \log p$ bits at once to generate the kernel of a cyclic smooth isogeny of degree 2^n. The isogeny is then computed efficiently using the technique of [13] to get the next curve in the graph. We show that this does not sacrifice any security and reduces the complexity of the original CGL hash from $5.7 \log p + 110$ to $13.5 \log \log p + 46.4$ modular multiplications per bit of the input. As the size of the prime p increases, this gives an exponential speed up.

Organization of the paper. In Section 2 we review some background on elliptic curves, isogenies and the CGL hash. Our new hash algorithm and the proofs of its

preimage and collision resistance are given in Section 3. In Section 4 we perform a detailed operation count on CGL and the new hash algorithm, and compare the runtime complexities.

2 Preliminaries

Let $\mathbb{F}_q$ be a finite field of q elements where $q = p^n$ for some prime $p > 3$ and integer $n \geq 1$. An elliptic curve $E/\mathbb{F}_q$ is an abelian variety of dimension 1 which also has genus 1; that is, a nonsingular projective curve of genus 1 which is also an abelian group. A morphism $E_1 \to E_2$ of elliptic curves that preserves the group structure is called an isogeny. An isogeny from an elliptic curve E to itself is called an endomorphism. The set of all such endomorphisms, denoted by $\mathrm{End}(E)$, form a ring under addition and composition.

Any isogeny $\phi : E_1 \to E_2$ induces an inclusion $\phi^* : \mathbb{F}_q(E_2) \hookrightarrow \mathbb{F}_q(E_1)$ of function fields. We say that ϕ is separable if ϕ^* is separable. Also the degree of ϕ, denoted by $\deg(\phi)$, is defined to be the degree of ϕ^*. We will call an isogeny of degree m an m-isogeny. For a separable isogeny ϕ we have $\deg(\phi) = |\ker \phi|$ [37]. For any integer m, the multiplication-by-m endomorphism $[m] : E \to E$ is separable. The kernel of $[m]$, denoted by $E[m]$, is the m-torsion subgroup of E. It can be shown that $E[m] \cong \mathbb{Z}/m\mathbb{Z} \oplus \mathbb{Z}/m\mathbb{Z}$ for any m such that $p \nmid m$. We have $E[p] = 0$ or $\mathbb{Z}/p\mathbb{Z}$. The curve E is called ordinary if $E[p] = \mathbb{Z}/p\mathbb{Z}$, and supersingular otherwise. This is equivalent to saying that $\mathrm{End}(E)$ is an order in an imaginary quadratic extension or a quaternion algebra over $\mathbb{Q}$ [30, §V.3].

Two curves $E_1/\mathbb{F}_q$ and $E_2/\mathbb{F}_q$ are called isogenous if there exists an isogeny between them. For any isogeny $\phi : E_1 \to E_2$ there exists an isogeny $\hat{\phi} : E_2 \to E_1$ such that $\phi \circ \hat{\phi} = [m]$ where $m = \deg(\phi)$. Therefore, being isogenous is an equivalence relation between curves defined over $\mathbb{F}_q$. Two isogenous curves are either both ordinary or both supersingular. This means the isogeny classes of ordinary and supersingular curves are disjoint. As a consequence of Tate's isogeny theorem [34], E_1 and E_2 are $\mathbb{F}_q$-isogenous if and only if $|E_1(K)| = |E_2(K)|$ for any finite extension $K/\mathbb{F}_q$. This implies that all curves in the same isogeny class have the same number of $\mathbb{F}_q$-rational points.

2.1 Isogeny graphs

It can be shown that every supersingular elliptic curve can be defined over $\mathbb{F}_{p^2}$, that is its j-invariant is in $\mathbb{F}_{p^2}$. For a prime $\ell \neq p$, the set of isomorphism classes of elliptic curves over $\mathbb{F}_{p^2}$ and the degree-ℓ isogenies between them form a graph called the graph of ℓ-isogenies. The graph consists of ordinary and supersingular

components that, according to above remarks, are disconnected. The ordinary components, which we will not discuss here, are called isogeny volcanoes [32].

There is only one supersingular component in the isogeny graph, which we denote by G_ℓ [17]. The nodes in G_ℓ are usually represented by the j-invariants. In this paper, we interchangeably use curves and j-invariants to refer to the vertices of G_ℓ. For $p = 3$ we have $|G_\ell| = 1$ and for $p \geq 5$ we have $|G_\ell| \approx [p/12]$. We consider the edges of G_ℓ to be isomorphism classes of ℓ-isogenies: two ℓ-isogenies $\phi : E_1 \to E_2$ and $\psi : E_1' \to E_2'$ are isomorphic if there are isomorphisms $\theta_1 : E_1 \to E_1'$ and $\theta_2 : E_2 \to E_2'$ such that $\theta_2\phi = \psi\theta_1$. Another way to look at the edges in G_ℓ is through the modular polynomial $\Phi_\ell(x, y) \in \mathbb{Z}[x, y]$ [38, §69]. The modular polynomial is symmetric in the sense that $\Phi_\ell(x, y) = \Phi_\ell(y, x)$, and is of degree $\ell + 1$ in both x, y. It is well known that there is an ℓ-isogeny between two curves E_1, E_2 with j-invariants j_1, j_2 if and only if $\Phi_\ell(j_1, j_2) = 0$. Therefore, the neighbors of each $E \in G_\ell$ are exactly the curves with j-invariants a root of the univariate polynomial $\Phi_\ell(x, j(E))$. Since all the j-invariants are in $\mathbb{F}_{p^2}$, we see that G_ℓ is an $(\ell + 1)$-regular graph.

2.2 Computational problems

In this subsection, we review the hard problems [5] that the security of our hash will be based on. Let n, which is the main security parameter, be a positive integer and let p be a prime of size $\approx n$ bits. For a prime $\ell \neq p$, denote by G_ℓ the graph of supersingular elliptic curves over $\mathbb{F}_{p^2}$.

Problem 2.1. Given a curve $E \in G_\ell$, find an endomorphism $\phi \in \mathrm{End}(E) \setminus \mathbb{Z}$ of degree ℓ^{rn} for some integer $r > 0$. By ϕ not being in $\mathbb{Z}$ we mean when rn is even, ϕ is not $\psi \circ [\ell^{rn/2}]$ for some automorphism ψ of E.

Problem 2.2. Given curves $E_1, E_2 \in G_\ell$, find an isogeny $\phi : E_1 \to E_2$ of degree ℓ^{rn} for some integer $r > 0$.

Problem 2.1 can be reduced to Problem 2.2 by taking a random walk ϕ of length rn from a curve E_1 to a curve E_2 in G_ℓ, and using the solver for Problem 2.2 to find another path $\psi : E_1 \to E_2$ of length sn. These two paths will be distinct with high probability. So the composition $\hat{\psi} \circ \phi$ is an endomorphism of E_1.

Attacks. Problem 2.2 is known as the *Supersingular Isogeny Problem*, and was first introduced in [16]. As noted in [5], a variation of the Pollard-rho attack would give an algorithm of complexity $O(\sqrt{p}\log^2 p)$ for this problem.

Another attack is known as the *claw finding* attack. The claw finding problem is as follows. Given sets X, Y, Z and functions $f : X \to Y$ and $g : Z \to Y$ as

an oracle, find at least one pair $(x, y) \in X \times Z$ such that $f(x) = g(z)$. A naive algorithm can solve this in time $O(|X| + |Z|)$. Therefore, setting X and Z to be all the isogenies of length $n/2$ starting from E_1 and E_2, respectively, we get an attack of complexity $O(\sqrt{p})$ on Problem 2.2. Using a quantum computer, the claw finding problem can be solved in time $O(\sqrt[3]{|X||Z|})$ which is optimal for black-box claw algorithm [33, 42]. This gives a quantum attack of complexity $O(\sqrt[3]{p})$.

The best known attack on Problem 2.2 is due to Biasse et al. [4]. Given curves E_1, E_2 over $\mathbb{F}_{p^2}$, the idea is to generate random isogenies $E_1 \to E_1'$ and $E_2 \to E_2'$ until E_1' and E_2' are both defined over $\mathbb{F}_p$. Using Grover's algorithm, this can be done in $O(p^{1/4})$ quantum operations. Computing an isogeny between E_1' and E_2' can then be done in subexponential time. The total complexity of the algorithm is thus $O(p^{1/4})$. However, this attack does not compute an isogeny of degree ℓ^{nr} for some integer r. So it is unclear that this algorithm actually solves Problem 2.2.

Another computational problem related to supersingular isogeny graphs is the *endomorphism ring problem* which is: given $E \in G_\ell$, compute the endomorphism ring $\mathrm{End}(E)$. In a recent work by Petit and Lauter [25], it is shown that the endomorphism ring problem is polynomially equivalent to Problem 2.2 under some plausible heuristic assumptions. Petit and Lauter also give an algorithm that can efficiently compute an endomorphism for a special j-invariant in the isogeny graph. This leads to a backdoor attack on the CGL hash which can easily be detected if a collision is produced. Later, Eisentraeger et al. [15] showed that the endomorphism ring problem reduces to Problem 2.2 if in addition to a chain of ℓ-isogenies, the representation of the ℓ-power isogeny by a left ideal in a maximal order is given.

2.3 The CGL hash

In this subsection, we review the original hash construction proposed in [5]. Let us first recall some definitions. For a family of hash functions $\mathcal{H} = \{h : \{0,1\}^{L(n)} \to S\}$, where $L(n) = \mathrm{poly}(n)$, we always assume that

- $2^{L(n)} > |S|$, and
- any $h \in \mathcal{H}$ is efficiently computable.

A hash function is called collision resistant if it is computationally infeasible to find two messages that hash to the same value for any member $h \in \mathcal{H}$.

Definition 2.3. A hash function h is called *provable collision resistant* if there exists a computational hard problem that is polynomially reducible to any algorithm that can find collisions in h.

A hash function is called preimage resistant if given an output y of the hash, it is computationally infeasible to find a message that hashes to y.

Definition 2.4. A hash function h is called *provable preimage resistant* if there exists a computational hard problem that is polynomially reducible to any algorithm that can find preimages of h.

Let $G_\ell = G_\ell(\mathbb{F}_{p^2})$ be the graph of ℓ-isogenies over $\mathbb{F}_{p^2}$. For simplicity we only consider the case $\ell = 2$, i.e., the graph of 2-isogenies. The whole scheme can be easily generalized for any prime ℓ. Let $E \in G_\ell$ be a fixed starting curve. Since G_ℓ is 3-regular, there are three isogenies from E to the neighboring curves. One of these isogenies is ignored once and for all. Given an n-bit message $M = b_1 b_2 \dots b_n$, the process starts by choosing an isogeny from E according to the bit b_1 to arrive at a curve E_1. If we don't allow backtracking, then there are two isogenies out of E_1, one of which can be chosen according to b_2. Continuing the same process, the message M determines a unique path of length n in G_ℓ. Note that it is required to make a convention for the ordering of the isogenies at each curve so that the hash is well defined. That is, the same output is produced for the same messages.

The output of the hash is the j-invariant of the curve at the end of the path. The j-invariants are of the form $ax + b$ where x is a generator of the extension $\mathbb{F}_{p^2}/\mathbb{F}_p$. As suggested in [5], the output of the hash can be represented in $\log p$ bits by applying a linear congruential operator to the resulting j-invariant.

From this scheme, we see that selecting a different starting curve $E \in G_\ell$ gives a different hash function. This way, we get a family of hash functions $\mathcal{H} = \{h_j\}_{j \in G_\ell}$ indexed by the supersingular j-invariants. Assume the hashes accept inputs of length a multiple of n. Then the above hash family is provable collision and preimage resistant.

Theorem 2.5 ([5, Theorem 1]). *If there is polynomial-time algorithm for finding collisions in the hash family $\mathcal{H} = \{h_j\}_{j \in G_\ell}$, then there is a polynomial-time algorithm for Problem* 2.1 *when $\ell = 2$.*

Theorem 2.6 ([5, Theorem 2]). *If there is a polynomial-time algorithm for finding preimages in the hash family $\mathcal{H} = \{h_j\}_{j \in G_\ell}$, then there is a polynomial-time algorithm for Problem* 2.2 *when $\ell = 2$.*

By a polynomial-time algorithm we mean an algorithm that runs in $O(\text{poly}(\log p))$ time.

Algorithm 1 $h(E, m, T)$

Input:
- An n-bit message m,
- A supersingular curve $E \in G_\ell$ as the starting vertex,
- The generator T of the backtrack 2-isogeny from E

Output:
- $E' \in G_\ell$
- The generator T' of the kernel of the backtrack 2-isogeny from E'

Obtain generators P, Q of $E[2^n]$ deterministically (e.g., Alg. 3.1 of [39])
If $T =$ NULL then go to Step 8
if $2^{n-1}P = T$ **then**
    Swap P and Q
else if $2^{n-1}Q \neq T$ **then**
    $Q := P + Q$
end if ▷ Now the basis P, Q never leads to backtrack
Compute $R = P + mQ$
Compute an isogeny $\phi : E \to E'$ with kernel $\langle R \rangle$
Obtain the 2-torsion point T' on E' that backtracks on ϕ
return (E', T')

3 The new hash algorithm

In this section, we propose a new hash algorithm based on supersingular isogeny graphs G_ℓ. For simplicity, we assume $\ell = 2$, but the scheme can easily be generalized for any prime $\ell \geq 2$. To be able to compute smooth degree isogenies, i.e. isogenies of degree 2^n, using the optimal strategy technique of [13], we choose $p = 2^n f \pm 1$ where f is small. Then we can assume that the curves in G_ℓ have order $(p \mp 1)^2 = (2^n f)^2$. This follows from the fact that the group of $\mathbb{F}_{p^2}$-rational points on a supersingular elliptic curve over $\mathbb{F}_{p^2}$ is of the form $(\mathbb{Z}/(p \mp 1)\mathbb{Z})^2$. From this group structure we see that for each $E \in G_\ell$, the whole 2^n-torsion subgroup $E[2^n]$ is contained in $E(\mathbb{F}_{p^2})$. Let $P, Q \in E[2^n]$ denote a set of generators of the 2^n-torsion. Given any n-bit message m, we obtain a hash of m as follows.

First, we compute $R = P + mQ$ which determines a cyclic subgroup $H = \langle R \rangle \subset E$ of order 2^n. Then we compute an isogeny $E \to E'$ with kernel H, which is also of degree 2^n, and return the j-invariant of E' as the hash. This way, taking E as the starting vertex, we have mapped an n-bit message to a vertex $E' \in G_\ell$.

The function $h(E, m)$ computed using Algorithm 1 is a compression function: it accepts a j-invariant and a message m, and returns a j-invariant. Therefore, we

Algorithm 2 $H(E, m)$

Input: A message m, a supersingular curve $E \in G_\ell$ as the starting vertex
Output: A supersingular curve $E' \in G_\ell$

1: Pad the message m to get $m = m_1 \| m_2 \| \dots \| m_k$ where each block m_i is n bits
2: $T_1 = \text{NULL}$
3: $E_1 := E$
4: **for** $i = 1$ to k **do**
5: $\quad (E_1, T_1) := h(E_1, m_i, T_1)$
6: **end for**
7: **return** E'

can apply the Merkle-Damgård construction [12,20] to hash messages of arbitrary length using h.

Remark 3.1. In Step 1 of Algorithm 1, the generators P, Q of the 2^n-torsion should be obtained canonically so that the hash is well-defined. For example, one could use a predefined initial value as the starting index of the table T_1 (or T_2) in the entangled basis algorithm of [39].

Since the starting vertex can be any $E \in G_\ell$, Algorithm 2 gives a hash family $\mathcal{H} = \{H_j\}_{j \in G_\ell}$ indexed by the curves in G_ℓ. So a hash can be selected from the family by providing a curve $E \in G_\ell$. Note that as in the original CGL algorithm, we need to prevent backtracking.

Proposition 3.2. *For any given input message m, the isogeny path computed by Algorithm 2 contains no backtracks. In particular, the isogeny computed by Algorithm 2 is cyclic.*

Proof. We have the following:

An ℓ-power isogeny is cyclic if and only if there is no backtracking in the isogeny path.

This is essentially proved by Proposition 1 in [5]. More precisely, they prove that if an isogeny is not cyclic then it contains a backtrack. The other direction is easy to prove.

This implies that there will never be a backtracking in the cyclic isogeny computed in Step 9 of Algorithm 1. In other words, there is no backtracking for individual message blocks. So to ensure that there is no backtracking in the long isogeny corresponding to the input message m of Algorithm 2 we only need to prove that consecutive message blocks do not produce backtracking. For this, it is

enough to select the current kernel in a way that the first 2-isogeny of the current block is not dual to the last 2-isogeny of the previous block.

The construction of the basis points P, Q through Steps 1 - 7 ensures that the new P, Q are also a basis, and we always have $2^{n-1}Q = T$. This implies that $2^{n-1}R = 2^{n-1}P + m2^{n-1}Q = 2^{n-1}P + mT \neq T$, where the last inequality is because $2^{n-1}P$ is linearly independent from T. This proves the proposition. □

Note that backtrack prevention eliminates the possibility of simple attacks such as the following : compute two random isogenies $\phi_1 : E \to E_1$ and $\phi_2 : E \to E_2$ using two random message blocks m_1 and m_2. Then compute the duals $\hat{\phi}_1, \hat{\phi}_2$ corresponding to some message blocks t_1, t_2 respectively. This gives a collision $h(E, m_1 \| t_1) = h(E, m_2 \| t_2)$.

3.1 Preimage and Collision resistance

We assume $p = 2^n f \pm 1$ as above, and assume that the length of the input is kn for some integer $k \geq 1$. Let $\mathcal{H} = \{H_j\}_{j \in G_\ell}$ be the hash family computed using Algorithm 2. Following the notation of [5], given an isogeny $\phi : E \to E'$ of degree 2^n, we say that the two factorizations

$$E = E_0 \xrightarrow{\phi_1} E_1 \xrightarrow{\phi_2} E_2 \xrightarrow{\phi_3} \cdots \xrightarrow{\phi_n} E_n = E',$$

$$E = E_0 \xrightarrow{\phi'_1} E'_1 \xrightarrow{\phi'_2} E'_2 \xrightarrow{\phi'_3} \cdots \xrightarrow{\phi'_n} E_n = E'$$

are isomorphic if there exist isomorphisms $\epsilon_i : E_i \xrightarrow{\sim} E'_i$ such that $\phi'_i \circ \epsilon_{i-1} = \epsilon_i \circ \phi_i$ for all $1 \leq i \leq n$. In [5] it is proved that isomorphic isogenies correspond to the same input messages. That is indeed easy to see in their construction, since the decomposition of a long isogeny into 2-isogenies is directly determined by the bits of the message. Proving the same statement in our settings requires a bit of extra work.

Claim 3.3. *Algorithm* 2 *never produces isomorphic isogenies for inputs with different number of blocks.*

Proof. Since the isogenies are cyclic, they have unique decompositions into sequences of 2-isogenies. Since the degrees are different, the kernels have different sizes and the isogenies will not be isomorphic. □

Lemma 3.4. *Let ϕ, ϕ' be two isogenies generated by Algorithm* 2 *on inputs m, m', respectively. If ϕ and ϕ' are isomorphic, then $m = m'$.*

Proof. By Claim 3.3 m and m' must have the same number of blocks. We proceed by induction on the number of blocks. Since there is a one-to-one correspondence between n-bit integers $m \in \{0,1\}^n$ and kernels $P+mQ$ constructed in Algorithm 1, the lemma is true for a single block. Suppose m, m' have k blocks. Write $\phi = \psi_k \circ \psi$ and $\phi' = \psi'_k \circ \psi'$ where ψ, ψ' are the isogenies corresponding to the first $k-1$ blocks of m, m', and ψ_k, ψ'_k are the isogenies corresponding to the last blocks m_k, m'_k of m, m', respectively.

By the induction hypothesis ψ and ψ' correspond to the same $k-1$ blocks. Since ϕ, ϕ' are isomorphic, they must have the same cyclic kernels, that is $\ker(\psi_k \circ \psi) = \ker(\psi'_k \circ \psi')$. From this we have $\ker(\psi_k) = \ker(\psi'_k)$, or equivalently $\langle P + m_k Q\rangle = \langle P + m'_k Q\rangle$ which implies $m_k = m'_k$ by the above correspondence for single blocks. □

Theorem 3.5 (Preimage Resistance). *If there is a polynomial-time algorithm for finding preimages for the hash family $\mathcal{H}$, then there is a polynomial-time algorithm for Problem* 2.2.

Proof. Let $H \in \mathcal{H}$ be a hash function corresponding to an initial vertex $E \in G_\ell$. Given an output $E_1 \in G_\ell$ of H, a preimage for E_1 is a message $m = m_1 \| m_2 \| \dots \| m_k$, where each m_i is n bits. By construction, the message m corresponds to an isogeny $E \to E_1$ of degree 2^{kn}. This means finding a preimage for H is equivalent to finding a 2^{kn}-isogeny between the two given curves E, E_1. □

Remark 3.6. By Merkle-Damgård Theorem, collision resistance of the compression function implies the collision resistance of the hash function. A compression function $h(a,b)$ is said to be collision resistant if it is hard to find two pairs $(a_1, b_1) \neq (a_2, b_2)$ such that $h(a_1, b_1) = h(a_2, b_2)$. Therefore, we only need to prove that the compression function $h(E, m)$ of Algorithm 1 is collision resistant. But $h(E, m)$ is not collision resistant. In fact, we can easily find curves $E_1, E_2 \in G_\ell$ and n-bit messages m_1, m_2 such that $h(E_1, m_1) = h(E_2, m_2)$ as follows. Let $E \in G_\ell$ be any curve and let $P, Q \in E[2^n]$ be a basis generated by Algorithm 1. For any integer $0 \leq t_1 < 2^n$ we can construct an isogeny $\phi_1 : E \to E_1$ with kernel $\langle P + t_1 Q\rangle$. Now, the kernel of $\hat{\phi}_1 : E_1 \to E$ is of the form $\langle P_1 + m_1 Q_1\rangle$ for a basis $P_1, Q_1 \in E_1[2^n]$. We can efficiently find m_1 from ϕ. Repeating the process for another $t_2 \neq t_1$, we get an isogeny $\hat{\phi}_2 : E_2 \to E$ with kernel $\langle P_2 + m_2 Q_2\rangle$. Clearly, the pairs (E_1, m_1) and (E_2, m_2) give a collision in h.

This, however, does not imply that the hash $H(E, m)$ is not collision resistant. On the contrary, we prove in the following that Problem 2.1 is efficiently reducible to finding collisions in $H(E, m)$. This means, the collision resistance condition

on the compression function might not be required in some concrete instantiations of the Merkle-Damgård paradigm. In other words, the condition is sufficient but not necessary.

Theorem 3.7 (Collision Resistance). *If there is a polynomial-time algorithm for finding collisions in the hash family $\mathcal{H}$, then there is a polynomial-time algorithm for Problem* 2.1.

Proof. Let $H \in \mathcal{H}$ be a hash function corresponding to an initial vertex $E \in G_\ell$ and let $H(E, m) = j(E') = H(E, m')$ for two distinct messages m, m'. Let ϕ, ϕ' be the isogenies computed by Algorithm 2 on inputs m and m', respectively. Since m and m' are distinct, the isogenies ϕ and ϕ' are not isomorphic by Lemma 3.4. Therefore, $\hat{\phi} \circ \phi'$ is a valid solution to Problem 2.1. □

4 Complexity

In this section, we compare the running time complexity of the original CGL hash with the one proposed in Section 3. We will count the number of operations in $\mathbb{F}_p$, and denote multiplication, squaring, and inversion by **m**, **s**, and **i**, respectively. As the size of p increases, addition becomes negligible compared to multiplication, so we shall ignore addition in the subsequent complexity estimations. The costs of multiplication, squaring, and inversion in $\mathbb{F}_{p^2}$ are 3**m**, 2**m**, 1**i** + 2**m** + 2**s**, respectively. To get a more precise operation count we fix the following parameters:

- The prime $p = 2^n f - 1$, where f is a small positive integer,
- The prime $\ell = 2$ so that we work on the 2-isogeny graph G_ℓ,
- The length kn of the input message, where k is a positive integer.

We assume the curves in G_ℓ have order $(p+1)^2 = (2^n f)^2$. The assumption on the length of the input message means we have already padded the input message so that it is k blocks of size n bits.

The special shape $p = 2^n f - 1$ of our prime leads to a more precise operation count for exponentiations. This is because most of the bits in the binary representation of p are 1's. Let $a \in \mathbb{F}_{p^2}$ be any given element. We shall use the following operation counts in the next subsections:

- Testing if a is a quadratic residue amounts to raising the norm of a to the $(p+1)/4$, which takes $\approx \log p$**s**.

- Taking a square root of a using the method of [28], requires 1 inversion in $\mathbb{F}_p$, and raising to $(p+1)/4$ twice in $\mathbb{F}_p$, respectively. Therefore, $\sqrt{a}$ is computed using $\approx 2\log p\mathbf{s} + 1\mathbf{i}$.

4.1 Moving around the isogeny graph

The complexities of both hash algorithms clearly depend on the cost of walking around in G_ℓ. The standard approach is to use the Vélu formulas [36]. This involves operations such as point addition and scalar multiplication, and small degree isogeny computation and evaluation. So we need to choose a curve model that is most optimized for these operations. The three well-known models that are widely used for computations are the Weierstrass model, the Montgomery model [22], and the twisted Edwards model [2, 14].

The short Weierstrass model is written as $y^2 = x^3 + ax + b$. Using projective coordinates, one point addition in this model takes 40**m** and one doubling takes 27**m** [3]. If we assume one of the points is scaled to have $Z = 1$, then addition and doubling are done using 22**m** and 19**m**, respectively.

The Montgomery model is written as $by^2 = x^3 + ax^2 + x$. Using the X, Z coordinates, which is called the Kummer line, one differential addition in this model costs 12**m**, and one doubling costs 10**m**. If one of the points is scaled to have $Z = 1$, then a differential addition costs 13**m**, and a doubling costs 7**m**.

The twisted Edwards model is written as $ax^2+y^2 = 1+dx^2y^2$. Using projective coordinates, one addition in this model costs 32**m**, and one doubling costs 17**m**. If one of the points is scaled to have $Z = 1$, then an addition and doubling cost 20**m** and 14**m**.

Unfortunately, there is not much literature on efficient computation and evaluation of small degree isogenies. Analogues of the Vélu formulas for twisted Edwards curves are given in [23], and the ones for Montgomery curves are given in [13]. Note that since the order of curves in G_ℓ is $(2^n f)^2$, all curves have points of order 2, so any of the above models can be used for our algorithm. However, based on the above operation counts and the advice of [13], we choose to work with the Montgomery model in this paper.

Montgomery curves. As mentioned above, a Montgomery curve over $\mathbb{F}_{p^2}$ has equation

$$E_{a,b} : by^2 = x^3 + ax^2 + x$$

where $a, b \in \mathbb{F}_{p^2}$. The projective equation of $E_{a,b}$ is $bY^2Z = X^3 + aX^2Z + XZ^2$. The projection $x : E_{a,b} \setminus \{0\} \to \mathbb{P}^1$ defined by $(X : Y : Z) \mapsto (X : Z)$ is a morphism of order 2 that induces a bijection $E_{a,b}/\langle 1, -1\rangle \cong \mathbb{P}^1$. This map provides

efficient arithmetic in $E_{a,b}/\langle 1, -1\rangle$, done entirely in the X, Z coordinates. The line $\mathbb{P}^1$ can be considered as the Kummer variety of $E_{a,b}$, and is called the Kummer line of $E_{a,b}$. Since the map x takes both P and $-P$ to $x(P)$ for all $P \in E_{a,b}$, we cannot add two distinct points P, Q on the Kummer line unless the difference $P - Q$ is already known. This particular addition, that takes $P - Q$ as an input, is called differential addition.

Efficient formulas for the following operations on the Kummer line were given in [22].

- Doubling: $\{x(P), a\} \mapsto x(2P)$,
- Differential addition: $\{x(P), x(Q), x(P - Q)\} \mapsto x(P + Q)$,
- Double and add: $\{x(P), x(Q), x(P - Q), a\} \mapsto \{x(2P), x(P + Q)\}$,
- Ladder: $\{x(P), a, m\} \mapsto x(mP)$.

The last operation, known as the Montgomery ladder, is done using doubling and differential addition.

Isogenies of Montgomery curves. Computing 2-isogenies between Montgomery curves can also be done entirely on the Kummer line. Efficient formulas for 2 and 4-isogenies were derived in [13]. Later, it was observed by Costello et al. [9] that computing an isogeny of degree 2^n is more efficiently done using 4-isogenies. To avoid many inversions in computing small degree isogenies, it was proposed in [10] to consider "projective" coefficients for the curve $E_{a,b}$ as well. That is to write $E_{a,b}$ as $E_{(A:B:C)} : By^2 = Cx^3 + Ax^2 + Cx$ for some $C \neq 0$, with $b = B/C$ and $a = A/C$. Like the arithmetic on the Kummer line, this leads to an isogeny arithmetic in which curves and their quadratic twists are identified by working only with the coefficients $(A : C) \in \mathbb{P}^1$. Following [9] we also use $(A_{24} : C_{24})$ to denote the Montgomery curve constant $(a - 2)/4$.

The projective versions of the 4-isogeny formulas in [13] can be written as follows [9]. Let $P = (X_4 : Z_4) \in E_{(A:C)}$ be a point of order 4 and denote by $\phi : E_{(A_{24}:C_{24})} \to E_{(A'_{24}:C'_{24})}$ the 4-isogeny with kernel $\langle P\rangle$. The target curve of ϕ is given by

$$(A'_{24} : C'_{24}) = (X_4^4 - Z_4^4 : Z_4^4),$$

and an evaluation $(X' : Z') = \phi(X : Z)$ is given by

$$\begin{aligned}(X' : Z') = (X(2X_4Z_4Z - X(X_4^2 + Z_4^2))(X_4X - Z_4Z)^2 :\\ Z(2X_4Z_4X - Z(X_4^2 + Z_4^2))(Z_4X - X_4Z)^2).\end{aligned}$$

We shall also need computing and evaluating 2-isogenies for backtrack prevention, see Algorithm 1. For a point $P = (X_2, Z_2)$ of order 2, using the formulas in [27] the target curve of the isogeny ϕ with kernel $\langle P \rangle$ is given

$$(A' + 2 : 4) = (-X_2^2 : Z_2^2).$$

An evaluation $(X' : Z') = \phi(X : Z)$ is given by

$$(X' : Z') = (X(XX_2 - ZZ_2) : Z(XZ_2 - ZX_2)).$$

The costs of point and isogeny arithmetics on Montgomery curves, taken from [10, 27], are summarized in Table 1.

operation	input	output	m
doubling	$x(P), a$	$x(2P)$	16
differential addition	$x(P), x(Q), x(P-Q)$	$x(P+Q)$	13
double and add	$x(P), x(Q), x(P-Q), a$	$x(2P), x(P+Q)$	26
ladder	$x(P), a, m$	$x(mP)$	$23n$
compute 2-isogeny	$x(P)$	A', C'	4
evaluate 2-isogeny	$x(P)$	A', C'	12
compute 4-isogeny	$x(P)$	A', C'	8
evaluate 4-isogeny	$x(Q)$	$x(\phi(Q))$	22

Table 1. Costs of different operations for Montgomery curves

4.2 Complexity of CGL

For hashing a message in the original CGL algorithm, 2-torsion points and the Vélu formulas are used. This requires obtaining two 2-torsion points at each curve by eliminating the point corresponding to the arriving isogeny, and using Vélu to compute the next curve. The 2-torsion can be computed using $f(x)$ from the equation $y^2 = f(x)$ of the curve. The polynomial $f(x)$ is cubic, but a linear factor corresponding to one of the 2-torsion points is eliminated. This means we always have a quadratic equation to factor. Therefore, hashing each bit of the message needs: 1 isogeny computation, 1 isogeny evaluation, and 1 square root computation.

Modular polynomials. Since we only need to work with j-invariants, a more efficient approach is to use the modular polynomial $\Phi_2(x, y)$. For any curve $E \in G_\ell$, the univariate polynomial $\Phi_2(x, j(E))$ is a cubic with roots the j-invariants of the curves 2-isogenous to E. Eliminating one of the linear factors corresponding to the j-invariant of the previous curve, we are left with a quadratic equation. Now,

computing a square root gives the j-invariant of the next curve. This way, isogeny computation and evaluation is avoided altogether. For each bit of the input we need to do the following:

- Evaluate the modular polynomial for the current curve. The modular polynomial is

$$\Phi_2(x,y) = x^3 + y^3 - x^2y^2 + 1488(x^2y + y^2x) - 162000(x^2 + y^2) \\ 40773375xy + 8748000000(x + y) - 157464000000000.$$

  The evaluation $\Phi_2(x, j(E))$ requires 5**m** and a few scalar multiplications.

- Obtain a quadratic equation $g(x)$ from $\Phi_2(x, j(E))$ by factoring out a linear factor. This needs only 3**m**.

- Solve the quadratic equation. This needs 2**m** and 1 square root computation. The square root computation takes $5 \log p\mathbf{m} + \log p\mathbf{s} + 1\mathbf{i}$.

In summary, we need $(5 \log p + 10)\mathbf{m} + \log p\mathbf{s} + 1\mathbf{i}$ for each bit, and hence

$$kn((5 \log p + 10)\mathbf{m} + \log p\mathbf{s} + 1\mathbf{i}) \tag{4.1}$$

for a message of length kn bits.

4.3 Complexity of the new hash

For each n-bit block m of the input message Algorithm 2 performs the following:

- Obtain generators P, Q of the group $E[2^n]$ of the current curve E. This can be efficiently done using the *entangled basis* technique of [39]. An entangled basis computation takes, on average, 2 quadratic residuosity tests, 1 square root finalization and 24**m**. A quadratic residuosity test in $\mathbb{F}_{p^2}$ takes $\log p\mathbf{s}$, and the square root finalization can be done in $\log p\mathbf{s} + 1\mathbf{i}$. The total cost of basis generation is thus $24\mathbf{m} + 3 \log p\mathbf{s} + 1\mathbf{i}$.

- Compute $2^{n-1}P, 2^{n-1}Q$. The value $2^{n-1}P$ is always computed, but the value $2^{n-1}Q$ is computed 2/3 of the time on average. The total cost on average is then $\approx 1.67n$ point doublings which is $26.7n\mathbf{m}$.

- Compute $R = P + mQ$. For this, we first compute mQ using the usual Montgomery ladder which takes $23n\mathbf{m}$, and then add P at the cost of a few multiplications.

- Compute an isogeny $\phi : E \to E'$ with kernel $\langle R \rangle$. For this, we use the optimal strategy approach of [13]. Using this strategy, computing a 2^n-isogeny takes $\frac{1}{2}n \log n$ point doublings and 2-isogeny computations. As mentioned

above, we can use 4-isogenies instead of 2-isogenies. Using this strategy, obtaining the target curve E' takes $\frac{n}{2}$ 4-isogeny computations and $\frac{n}{4}\log\frac{n}{2}$ point quadrupling and 4-isogeny evaluations. According to Table 1, all these together take

$$\left(\frac{27}{2}n\log n - \frac{19}{2}n\right)\mathbf{m}.$$

- Compute the backtrack 2-torsion point. For this we only need to compute the last 4-isogeny as 2-isogenies, and then compute the image of a 2-torsion point that is not in the kernel of the last 2-isogeny. If the kernel is not $\langle(0,0)\rangle$ then we can simply use $(0,0)$ as the desired point. Otherwise, we need to take a square root to obtain a 2-torsion that is not in the kernel. The latter case happens with probability $\approx 1/3$. The total cost of this is step is then $\approx (2\log p\mathbf{s} + 1\mathbf{i})/3$.

Replacing $\log p$ by n, the total complexity of Algorithm 2 for an input of length kn bits is then

$$\left(\frac{27}{2}n\log n + 40.2n\right)k\mathbf{m} + \frac{10}{3}nk\mathbf{s} + \frac{4}{3}k\mathbf{i}. \tag{4.2}$$

Performance comparison. Comparing the complexity of the new hash algorithm, Eq. (4.2), with CGL, Eq. (4.1), we immediately see that the new algorithm is significantly faster. Asymptotically, the runtime of the new hash is quasi-linear in n while the runtime of CGL is quadratic in n. For a concrete performance comparison, we can replace the squaring and inversion operations **s** and **i** by a factor of multiplication **m**. A frequently used convention is to set **s** = 0.67**m** and **i** = 100**m**. From this, we get the estimations

$$kn(5.7n + 110)\mathbf{m}$$

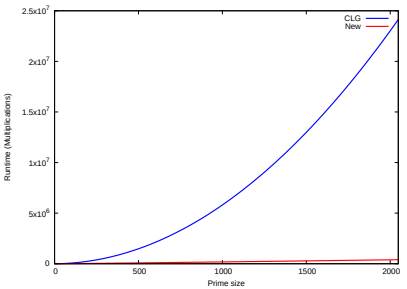
for complexity of the CGL hash algorithm, and

$$kn(13.5\log n + 42.4)\mathbf{m}$$

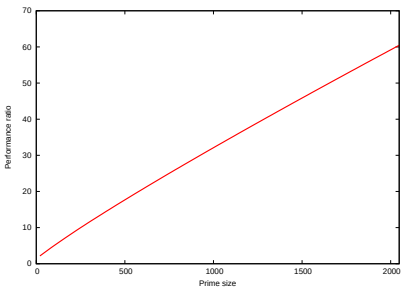
for complexity of the new hash algorithm. This leads to the performance ratio

$$\frac{5.7n + 110}{13.5\log n + 42.4} \tag{4.3}$$

which is asymptotically $c\frac{n}{\log n}$ for the constant $c = 5.7/13.5$. Figures 1a and 1b compare the performances of CGL and the new hash algorithm for different parameter sizes.



(a) Number of multiplications in $\mathbb{F}_{p^2}$ for CGL and the new hash algorithm



(b) Performance ratio between CGL and the new hash algorithm

Figure 1a shows the quadratic versus quasi-linear behaviors of the algorithms. From Figure 1b and Eq (4.3) we see that for $n = 1024$, i.e., a prime of size ≈ 1024 bits, the new algorithm is 33.5 times faster than the original CGL algorithm. Given that the best attack on the isogeny problem, and hence on both hash algorithms, has quantum complexity $O(p^{1/4})$, this corresponds to 256-bit quantum security level. For a 384-bit quantum security level, we get a performance ratio of 47.8.

Remark 4.1. Although our hash is significantly faster asymptotically, for short messages, it suffers from the computational overhead of the extra block of padding. For example, in the extreme case of hashing only one block of size $n = 1024$, the performance ratio drops to $(5.7 \times 1024 + 110)/(2 \times (13.5 \times 10 + 42.5)) \approx 16.8$.

Bibliography

[1] Miklós Ajtai, Generating hard instances of lattice problems, in: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, ACM, pp. 99–108, 1996.

[2] Daniel J Bernstein, Peter Birkner, Marc Joye, Tanja Lange and Christiane Peters, Twisted edwards curves, in: *International Conference on Cryptology in Africa*, Springer, pp. 389–405, 2008.

[3] Daniel J Bernstein and Tanja Lange, *Explicit Formulas Database*, 2007, http://www.hyperelliptic.org/EFD/index.html.

[4] Jean-François Biasse, David Jao and Anirudh Sankar, A quantum algorithm for computing isogenies between supersingular elliptic curves, in: *International Conference in Cryptology in India*, Springer, pp. 428–442, 2014.

[5] Denis X Charles, Kristin E Lauter and Eyal Z Goren, Cryptographic hash functions from expander graphs, *Journal of Cryptology* **22** (2009), 93–113.

[6] David Chaum, Eugène van Heijst and Birgit Pfitzmann, Cryptographically strong undeniable signatures, unconditionally secure for the signer, in: *Annual International Cryptology Conference*, Springer, pp. 470–484, 1991.

[7] Richard Cleve, Artur Ekert, Chiara Macchiavello and Michele Mosca, Quantum algorithms revisited, in: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 454, The Royal Society, pp. 339–354, 1998.

[8] Scott Contini, Arjen K Lenstra and Ron Steinfeld, VSH, an Efficient and Provable Collision-Resistant Hash Function., in: *Eurocrypt*, 4004, Springer, pp. 165–182, 2006.

[9] Craig Costello and Huseyin Hisil, A simple and compact algorithm for SIDH with arbitrary degree isogenies, in: *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, pp. 303–329, 2017.

[10] Craig Costello, Patrick Longa and Michael Naehrig, Efficient algorithms for supersingular isogeny Diffie-Hellman, in: *Annual Cryptology Conference*, Springer, pp. 572–601, 2016.

[11] Ronald Cramer and Victor Shoup, Signature schemes based on the strong RSA assumption, *ACM Transactions on Information and System Security (TISSEC)* **3** (2000), 161–185.

[12] Ivan Bjerre Damgård, A design principle for hash functions, in: *Conference on the Theory and Application of Cryptology*, Springer, pp. 416–427, 1989.

[13] Luca De Feo, David Jao and Jérôme Plût, Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies, *Journal of Mathematical Cryptology* **8** (2014), 209–247.

[14] Harold Edwards, A normal form for elliptic curves, *Bulletin of the American Mathematical Society* **44** (2007), 393–422.

[15] Kirsten Eisentraeger, Sean Hallgren and Travis Morrison, *On the Hardness of Computing Endomorphism Rings of Supersingular Elliptic Curves*, Cryptology ePrint Archive, Report 2017/986, 2017, https://eprint.iacr.org/2017/986.

[16] Steven D Galbraith, Constructing isogenies between elliptic curves over finite fields, *LMS Journal of Computation and Mathematics* **2** (1999), 118–138.

[17] David R. Kohel, *Endomorphism rings of elliptic curves over finite fields*, Ph.D. thesis, University of California, Berkeley, 1996.

[18] Vadim Lyubashevsky and Daniele Micciancio, Generalized compact knapsacks are collision resistant, *Automata, Languages and Programming* (2006), 144–155.

[19] Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert and Alon Rosen, SWIFFT: A modest proposal for FFT hashing, *Lecture Notes in Computer Science* **5086** (2008), 54–72.

[20] Ralph C Merkle, A certified digital signature, in: *Conference on the Theory and Application of Cryptology*, Springer, pp. 218–238, 1989.

[21] Daniele Micciancio, Generalized compact knapsacks, cyclic lattices, and efficient one-way functions, *Computational Complexity* **16** (2007), 365–411.

[22] Peter L Montgomery, Speeding the Pollard and elliptic curve methods of factorization, *Mathematics of computation* **48** (1987), 243–264.

[23] Dustin Moody and Daniel Shumow, Analogues of Vélu's formulas for isogenies on alternate models of elliptic curves, *Mathematics of Computation* **85** (2016), 1929–1951.

[24] Michael A Nielsen and Isaac Chuang, *Quantum computation and quantum information*, AAPT, 2002.

[25] Christophe Petit and Kristin Lauter, *Hard and easy problems for supersingular isogeny graphs*, Cryptology ePrint Archive, Report 2017/962, 2017, https://eprint.iacr.org/2017/962.

[26] Arnold K Pizer, Ramanujan graphs and Hecke operators, *Bulletin of the American Mathematical Society* **23** (1990), 127–137.

[27] Joost Renes, Computing isogenies between Montgomery curves using the action of (0, 0), in: *International Conference on Post-Quantum Cryptography*, Springer, pp. 229–247, 2018.

[28] Michael Scott, Implementing cryptographic pairings, *Lecture Notes in Computer Science* **4575** (2007), 177.

[29] Peter W Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM review* **41** (1999), 303–332.

[30] Joseph H Silverman, *The arithmetic of elliptic curves*, 106, Springer Science & Business Media, 2009.

[31] Daniel R Simon, On the power of quantum computation, *SIAM journal on computing* **26** (1997), 1474–1483.

[32] Andrew Sutherland, Isogeny volcanoes, *The Open Book Series* **1** (2013), 507–530.

[33] Seiichiro Tani, Claw finding algorithms using quantum walk, *Theoretical Computer Science* **410** (2009), 5285–5297.

[34] John Tate, Endomorphisms of abelian varieties over finite fields, *Inventiones mathematicae* **2** (1966), 134–144.

[35] Jean-Pierre Tillich and Gilles Zémor, Group-theoretic hash functions, in: *Workshop on Algebraic Coding*, Springer, pp. 90–110, 1993.

[36] Jacques Vélu, Isogénies entre courbes elliptiques, *Comptes-Rendus de l'Académie des Sciences* **273** (1971), 238–241.

[37] Lawrence C Washington, *Elliptic curves: number theory and cryptography*, CRC press, 2008.

[38] Heinrich Weber, Lehrbuch der Algebra, vol. 3, *III, third edition (Chelsea, New York, 1961)* (1908).

[39] Gustavo H. M. Zanon, Marcos A. Simplicio Jr., Geovandro C. C. F. Pereira, Javad Doliskani and Paulo S. L. M. Barreto, *Faster isogeny-based compressed key agreement*, Cryptology ePrint Archive, Report 2017/1143, 2017, https://eprint.iacr.org/2017/1143.

[40] Gilles Zémor, Hash functions and graphs with large girths, in: *Advances in Cryptology-EUROCRYPT'91*, Springer, pp. 508–511, 1991.

[41] Gilles Zémor, Hash functions and Cayley graphs, *Designs, Codes and Cryptography* **4** (1994), 381–394.

[42] Shengyu Zhang, Promised and distributed quantum search, in: *International Computing and Combinatorics Conference*, Springer, pp. 430–439, 2005.

Author information

Javad Doliskani, Institute for Quantum Computing, University of Waterloo, Canada.
E-mail: javad.doliskani@uwaterloo.ca

Geovandro C. C. F. Pereira, Institute for Quantum Computing, University of Waterloo, Canada.
E-mail: geovandro.pereira@uwaterloo.ca

Paulo S. L. M. Barreto, University of Washington Tacoma, USA.
E-mail: pbarreto@uw.edu