Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

Quantum-Resistant Cryptographic Hash Functions: Leveraging Ramanujan Graphs and Genetic Algorithms for Enhanced Security and Efficiency

Krishnaprabha R

Department of Mathematics, Amrita School of Physical Science, Amrita Vishwa Vidyapeetham, India
Email: krishnaprabha.kpr@gmail.com

This paper presents a novel approach to constructing cryptographic hash functions by lever- aging the spectral properties of Ramanujan graphs and the optimization capabilities of Genetic Algorithms (GAs). Ramanujan graphs, known for their sparse structure and large spectral gap, provide a robust mathematical foundation for creating collision-resistant hash functions. We utilize Ramanujan graphs constructed by Pizer and Lubotzky-Phillips-Sarnak (LPS) to encode the difficulty of computing isogenies between super singular elliptic curves, a problem known to be resistant to both classical and quantum attacks. Furthermore, a Genetic Algorithm framework is employed to optimize the hash function, enhancing its efficiency and adaptability for diverse cryptographic applications. The proposed method is rigorously analyzed for collision resistance, pre-image resistance, and efficiency. Experimental results demonstrate its feasibility for practical implementation in post-quantum secure communication and block chain systems. This study advances the design of cryptographic hash functions, offering a scalable and quantum-resistant alternative to existing solutions.

Keywords: Hash function, Rmanujan graphs, Pizer's graphs, LPS graphs, Genetic Algorithm.

1. Introduction

With the rapid advancement of quantum computing, classical cryptographic systems face unprecedented challenges. Quantum algorithms, such as Shor's algorithm [18], threaten the foundational assumptions underpinning many widely-used cryptographic schemes, including those ensuring collision resistance in hash functions [16,5]. A cryptographic hash function is a cornerstone of secure systems, playing critical roles in data integrity, digital signatures, authentication protocols, and block chain technologies. The primary requirement for such functions is their ability to resist collisions—where two different inputs produce the same hash output—a property that becomes increasingly vulnerable in the quantum era.

Current hash function designs, while robust in classical settings, often fail to meet the combined requirements of scalability, efficiency, and security in post-quantum contexts. For instance, many classical hash functions either lack the structural complexity needed to withstand quantum attacks or introduce significant computational overhead when adapted for post-quantum security. These limitations present a critical research gap: the need for hash functions that are not only resistant to quantum attacks but also efficient and scalable enough for practical applications, such as secure communication and block chain systems.

Additionally, existing methods for enhancing collision resistance often rely on computational assumptions that may weaken with advances in quantum algorithms. This underscores the necessity of exploring alternative mathematical structures that inherently exhibit strong security properties. Among such structures, Ramanujan graphs have emerged as a promising candidate due to their exceptional spectral properties, which contribute to cryptographic robustness. However, the integration of these graphs into hash function design and their potential for enhancing quantum resistance remain underexplored.

This study addresses these challenges by leveraging Ramanujan graphs to design a novel cryptographic hash function that aligns with post-quantum security requirements. By linking the security of the proposed hash function to the difficulty of computing isogenies between super singular elliptic curves, we create a robust framework that not only fills the identified research gap but also sets a foundation for future advancements in quantum-resistant cryptography.

A class of cryptographic hash functions known as Cayley hash functions is built from Cayley graphs and offers desirable characteristics including natural parallelism and a reduction of security to a clear-cut, well-defined mathematical issue. Cayley hash functions might serve as a solid basis for post-quantum cryptography as this problem is inherently resistant to quantum period finding methods due to its involvement of non-Abelian groups. There have been four distinct parameter settings for Cayley hash functions, and pre image algorithms have been identified for each of them.

For the most part, Cayley graphs are explicit Ramanujan graphs. Pizer created an explicit Ramanujan network, the only non-Cayley graph known to exist, based on the super singular elliptic curve isogeny theory. We concentrate on Ramanujan graphs made from Cayley graphs, even if Pizer's Ramanujan graphs have some cryptographic uses.

Assume that X = (V,E) is a finite graph that is (undirected) and has vertex set V and edge set E .If there are k edges incident on each vertex in a graph, then that graph is considered k-regular. Let $F\subset V$, the set of all edges connecting F to $V \setminus F$ is known as the border ∂F. The expanding constant of X is defined by Here, we'd want to use the well-known mathematical structures of elliptic curves and graphs to build a safe hash function. In fact, we'll go through a technique for allocating super singular elliptic curves with the characteristic p to an expander graph's vertices and isogenies between those elliptic curves to the edges that intersect those vertices.

$$h(X) = \min\left\{\frac{|\partial F|}{|F|} : F \subseteq V \text{ such that } 0 < |F| < \frac{|V|}{2}\right\}.$$

The objective is to build a collision-resistant, efficiently calculable hash function. We suggest generating cryptographic hash functions from expander graphs that can be proven. The hash function returns the finishing vertex and uses the input as a set of instructions for traversing a graph. Any expander graph may be constructed using our method, however we present two families of ideal expander graphs here so that you can compare their effectiveness and collision resistance. The two families are, respectively, Pizer's and Lubotzky-Phillips-Sarnak's (LPS) Ramanujan graphs. Since Ramanujan graphs are the best expander graphs, they offer superior mixing capabilities. Computing collisions is at least as difficult as computing isogenies between super singular elliptic curves when building a hash function from the Ramanujan network of super singular elliptic curves overFp2 with l-isogenies, l a prime distinct from p. The best method now available solves what is thought to be a highly challenging issue in square-root time.

If computing a collision requires solving another well-known hard problem, such as factoring or discrete log, we refer to it as a provable hash. First section of this paper is about hash functions and elliptic curves. We will provide an overview of the construction of a hash function using expander graphs in the section after this. Then describe how collision-resistant hash functions are built using Pizer's Ramanujan graphs.

The endomorphism ring calculation problem and the path finding problem are the same for super singular elliptic curves. Ramanujan graphs are the best expander graphs because of their mixing properties. Building verified collision-resistant hash functions using expander graphs is made much easier by the mixing characteristics of these networks. The output of hash functions closely mimics the uniform distribution and, as a result, is almost similar to random bits, according to the rapid mixing characteristic. The most useful property of expander graphs for cryptography is their ability to quickly approximate the uniform distribution using random walks. Numerous techniques for computing collision-resistant hash functions use walking directions as the input, and the hash function's output is the last vertex.

In the last part of this paper we will show the role of Artificial Intelligence algorithm such as Genetic Algorithm (GA) [10,6] for improving the performance and security of hash functions. The paper is structured to comprehensively address the design and optimization of a quantum-resistant cryptographic hash function. The Abstract presents a concise summary, including the research problem, methodology, and findings. The Introduction provides the background, articulates the problem, identifies the research gap, and lists the contributions and paper organization. The Review of Literature surveys relevant works on cryptographic hash functions, Ramanujan graphs, and Genetic Algorithms, identifying limitations in existing methods. The Methodology outlines the construction of the hash function using Ramanujan graphs for collision resistance and GAs for optimization, detailing design principles, traversal mechanisms, and security analysis.T he Construction of Cryptographic Hash Functions Using Genetic Algorithm section elaborates on the integration framework, algorithm details, and implementation aspects. The Results and Discussion present performance evaluation, security analysis, and a comparative study, highlighting the method's efficiency, robustness, and practical implications. Finally, the Conclusion summarizes the findings, discusses limitations, and suggests directions for future research. Review of Literature The review of literature should provide a comprehensive overview of existing research relevant to cryptographic hash functions, Ramanujan graphs, Genetic Algorithms(GAs), and their combined applications [15,1,8].

Cryptographic Hash Functions

Cryptographic hash functions are foundational elements in secure communication, digital signatures, and block chain technology. A robust hash function must satisfy three essential properties:

Collision Resistance: It should be computationally infeasible to find two distinct inputs producing the same hash output.

Pre-image Resistance :Given a hash output, it must be computationally difficult to retrieve the original input.

Second Pre-image Resistance: It must be challenging to find a second input that maps to the same hash as a given input.

Traditional hash functions, such as SHA-256 and SHA-3, have proven effective in classical settings but are vulnerable to attacks leveraging quantum computing, such as Grover's algorithm, which reduces their security strength by half. Consequently, there is a growing need for hash functions that are resistant to quantum attacks, prompting research into alternative methods that incorporate hard mathematical problems, such as those based on isogeny graphs and sparse Ramanujan graphs.

Ramanujan Graphs in Cryptography

Ramanujan graphs, introduced by Lubotzky, Phillips, and Sarnak (LPS), have been extensively studied for their unique spectral properties, including their large spectral gap, which ensures strong connectivity and sparse structure.

Applications in Cryptography: These graphs have been utilized in cryptographic protocols due to their inherent hardness properties, making them suitable for constructing secure systems.

Super singular Isogeny Graphs: Ramanujan graphs derived from super singular elliptic curves have gained prominence in post-quantum cryptography. The difficulty of computing isogenies between these curves underpins the security of protocols such as Super singular Isogeny Diffie Hellman (SIDH).

Several studies, such as those by Pizer and the LPS group, have explored the construction of Ramanujan graphs, providing a basis for leveraging their mathematical properties in secure hash function design. Despite their potential, the integration of these graphs into cryptographic hash functions remains underexplored.

Genetic Algorithms in Optimization

Genetic Algorithms, inspired by the process of natural selection, are widely used for solving complex optimization problems.

Key Components: Encoding solutions, fitness evaluation ,crossover, mutation, and selection.

Applications in Cryptography: GAs have been employed to optimize cryptographic primitives, such as key generation and cipher design, due to their ability to navigate large solution spaces efficiently.

Benefits in Hash Functions: GAs can enhance hash function efficiency by optimizing parameters like traversal paths and encoding mechanisms to reduce computational overhead while maintaining security properties.

While GAs have been applied in various cryptographic contexts, their application in enhancing graph based hash functions remains a novel area of exploration.

Intersection of Ramanujan Graphs and GAs in Hash Functions

The integration of Ramanujan graphs with GAs offers a promising avenue for creating quantum- resistant hash functions.

Sparse and Efficient Structures: Ramanujan graphs provide a scalable structure for hash generation, while GAs optimize the parameters to ensure collision resistance and efficiency.

Hardness Properties: By encoding the isogeny problem in super singular elliptic curves within the hash function, the construction achieves robustness against quantum attacks.

Recent research highlights the growing interest in hybrid approaches that combine the strengths of mathematical structures and optimization algorithms. However, no prior work explicitly integrates Ramanujan graphs and GAs to design cryptographic hash functions, making this study a significant contribution to the field.

Gaps in Existing Literature

Lack of scalable, efficient, and quantum-resistant hash functions based on hard mathematical problems.

Limited exploration of Ramanujan graphs in practical cryptographic applications.

Under utilization of Genetic Algorithms in optimizing graph-based cryptographic constructions.

Methodology

This section outlines the approach used to construct the proposed cryptographic hash function by integrating Ramanujan graphs and Genetic Algorithms (GAs). The methodology is divided into distinct stages to ensure clarity and reproducibility.

Design Principles

The construction of the hash function is based on the following principles:

Spectral Properties of Ramanujan Graphs: Ramanujan graphs are employed for their sparse structure and large spectral gap, ensuring strong connectivity and inherent collision resistance.

Hard Mathematical Problems: The difficulty of computing isogenies between super singular elliptic curves forms the backbone of the collision-resistant property.

Optimization via Genetic Algorithms: GAs are utilized to refine the traversal mechanisms and parameter settings, ensuring computational efficiency and adaptability.

Selection of Ramanujan Graphs

Construction of Graphs: Utilize Ramanujan graphs constructed by Pizer and Lubotzky-Phillips-Sarnak (LPS) for their well-documented properties and mathematical robustness. Choose appropriate graph parameters such as vertex count and spectral gap to balance scalability and efficiency. Vertex Encoding:

Represent input data as vertices in the Ramanujan graph.

Map the input using a deterministic hashing scheme to ensure uniform distribution across the graph.

Traversal-Based Hash Function Construction

Graph Traversal Algorithm:

Define a traversal algorithm that selects a path in the graph based on the encoded input.

Traverse the graph to produce a sequence of vertices, with the sequence determined by the input data.

Hash Output Generation:

Aggregate the vertex sequence into a fixed-length hash value using a predefined aggregation function (e.g., XOR or modular arithmetic).

Ensure that the final hash value maintains the desired cryptographic properties, such as fixed length and uniform distribution.

Integration of Genetic Algorithms

Encoding Mechanism:

Encode traversal paths as chromosomes in the Genetic Algorithm.

Represent traversal parameters, such as starting points and step sizes, as genes.

Fitness Function: Define a fitness function to evaluate the quality of each hash function instance. Criteria include:

Collision resistance: Measure the difficulty of finding two inputs with the same hash.

Pre-image resistance: Assess the infeasibility of reversing the hash.

Efficiency: Evaluate the computational cost of the hash generation process.

GA Operations:

Selection: Select the best-performing traversal strategies based on fitness scores.

Crossover: Combine the genes of high-fitness individuals to explore new traversal strategies.

Mutation: Introduce random variations to maintain diversity and avoid local optima.

Optimization Process:

Iterate the GA over multiple generations until convergence on an optimized traversal mechanism.

Finalize the traversal strategy and parameters for the hash function.

Security Analysis

Collision Resistance: Analyze the hash function's resistance to collisions by evaluating the uniqueness of hash outputs for random inputs.

Pre-image and Second Pre-image Resistance: Test the function against pre-image attacks by attempting to reverse engineer the input from the hash output.

Quantum Resistance: Assess the computational complexity of attacks leveraging quantum algorithms, ensuring robustness against Grover's algorithm and quantum isogeny attacks.

Implementation and Testing

Implementation:

Develop a prototype implementation of the hash function using programming tools like Python or C++.

Integrate the Ramanujan graph-based framework with the GA optimization module.

Testing Scenarios:

Test the hash function with datasets of varying sizes and characteristics.

Benchmark performance against standard cryptographic hash functions such as SHA-256 and SHA-3.

Performance Evaluation

Efficiency Metrics: Measure computational time, memory usage, and scalability.

Security Metrics: Quantify the resistance to collision, pre-image, and quantum attacks.

Comparison with Existing Methods: Compare the proposed hash function with traditional and post-quantum hash functions to highlight its advantages.

2. Preliminaries

EllipticCurve

Let Fq be a finite field where q=pa, p a large prime. [13] Short Weierstrass form of an elliptic curve E over Fq is the set of points E (Fq) ={ (x, y), x, y ∈ Fq : y2 = x3 + ax + b } ∪ { 0E }, a, b ∈ Fq ,0E = 0 = (0 : 1 : 0) the point at infinity.

The j-invariant of an elliptic curve given by the Weierstrass equation y2=x3+ax+b is given by the

4a3

formula j(E)=17284a3+27b2.   For the elliptic curve E0 : y2 = x3 + x the j-invariant is j(E0)=1728. Isomorphic curves have the same j-invariant; over an algebraically closed field, two curves with the same j-invariant are isomorphic. Let E: y2= x3+ax+b be an elliptic curve. Let P1= (x1, y1) and P2 = (x2, y2) be two points on E different from the point at infinity, then we define the binary operation ⊕ on E as P ⊕ 0 = 0⊕P=P for any point P ∈ E. If x1 = x2 and y1 = -y2, thenP1 ⊕ P2 = 0;

Otherwise set

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, & \text{if } P \neq Q \\ \\ \frac{3x_1^2 + a}{2y_1}, & \text{if } P = Q \end{cases}$$

The point(P1⊕P2)=(x3,y3)is defined by

x3=λ2−x1−x2

y3=−λx3−y1+λx1

Under the binary operation ⊕ the set of points on the elliptic curve E forms an abelian group[14]. Over finite fields, they form finite Abelian groups.

Let E/Fq be an elliptic curve, and suppose that #E(Fq) = n is relatively prime to q. Then E(Fq) is either cyclic or isomorphic to the direct product of two cyclic groups.

Theorem: Let E/Fq be an elliptic curve, with q=pr for some prime p. Then E(Fq) is cyclic if

One of the following conditions hold:1.#E(Fq) =pk for some integer k. 2. E/Fqis super singular and q≥5 is prime with q=1mod 4 .An isogeny between two elliptic curves E and E′ is a surjective morphism [3]. Isogeny may not be injective. If E=E′, we call an isogeny an endomorphism The collection of all endomorphisms on E along with the zero map is called the endomorphism ring, denoted by End(E) An example for an endomorphism on the elliptic curve E is the multiplication-by- m map ie [m] : E →E defined by P→[m]P, P∈E(F).

Degree of an isogeny

Let ϕ: E1→ E2 be an isogeny then the degree of ϕ is the maximum of the degrees of the polynomials f1(x,y) and f2(x,y) .If the degree of an isogeny is l then it is known as an l-isogeny

Dual isogeny of the l isogeny ϕ : E1 →E2 is an l isogeny ϕˆ : E2→E1 such that ϕ ◦ ϕˆ = ϕˆ◦ ϕ = [l] where [l] is the multiplication by l map.

Elliptic curves related by an isogeny of degree n are said to be n-isogenous. Two elements j1, j2 of a field k are n-isogenous over k if there are n-isogenous elliptic curves E1; E2 over k with j(E1) = j1 and j(E2) = j2.

Ramanujan Graphs

In a graph G = (V,E), two points xi, xj ∈ V are adjacent or neighbors if { xi, xj } ∈ E. If all the vertices of G are pair wise adjacent, then we say G is complete [9]. A complete graph with n vertices is denoted as Kn. The degree d(v) of a vertex v is the number of vertices in G that are adjacent to v.Adjacency matrix associated with a graph G is a square matrix AG the entries ai,j are given by

$$a_{i,j} = \begin{cases} 1 \text{ if } \{x_i, x_j\} \in E \\ 0 \text{ other wise} \end{cases}$$

The adjacency matrix of a graph is to calculate the number of walks of different length connecting two vertices in the graph. A vertex v ∈ V is incident with an edge { xi, xj } E if v=vi or v=vj. A walk on a graph is an alternating series of vertices and edges, beginning and ending with a vertex, in which each edge is incident with the vertex immediately preceding it and the vertex immediately following it.

A walk between two vertices u and v is called a u-v walk. The length of a walk is the number of edges it has. In a walk, we count repeated edges as many times as they appear. The eigen value of a graph is the eigen value of the corresponding adjacency matrix. Since the diagonal of the adjacency matrix is all zeros the sum of all eigen values of a graph is always 0.

An expander graph [7] is a graph in which every subset of the vertices has many neighbors. An undirected graph G=(V, E) is c−expanding if for every S ⊆V, |S|≤(|V|)/((2c)), and |Γ(S)|≥(c–1)|S|, where Γ(S) is the set of neighbors of S (not including S). The most effective graphs for preserving communication secrecy are the Ramanujan and Caley graphs. Since it is difficult to locate pathways in these networks, these graphs are crucial in cryptography. For super singular elliptic curves, the path finding issue and the endomorphism ring calculation problem are identical. The mixing qualities of Ramanujan graphs make them ideal expander graphs.

The mixing features of these graphs are highly helpful for building verifiable collision-resistant hash functions from expander graphs. According to the quick mixing property, the output of hash functions closely resembles the uniform distribution and, as a result, is essentially identical to random bits. Expander graphs' ability to swiftly approximate the uniform distribution using random walks is their most valuable feature for cryptography. Several methods for calculating collision-resistant hash functions employ the input as walking directions around a graph, and the output of the hash function is the last vertex.[2] Any expander graph may use these constructions, however locating cycles in such a graph is a challenging challenge. We can maintain a short path length in a Ramanujan graph. The difficulty of reversing a walk on a graph is largely used in the construction of cryptographic hash functions.

Path-finding problem:“ For a fixed starting and ending vertices representing the start and end points of a walk on the graph of a fixed length, find a path between them. A hash function can be defined by using the input to the function as directions for walking around the graph: the output is the label for the ending vertex of the walk. Finding collisions for the hash function is equivalent to finding cycles in the graph, and finding pre-images is equivalent to path-finding in the graph.n Backtracking is not allowed in these walks. Charles–Goren–Lauter in

2006 proposed two hash functions based on the hardness of finding paths in Lubotzky–Phillips–Sarnak (LPS) graphs and the next one is based on Super singular Isogeny Graphs (Pizer). [2] In 2008 Petit–Lauter–Quisquater breaks the hash function based on LPS graphs."

As a result, Super singular Isogeny graphs have lately attracted the attention of scholars. We transmit the images of the elements in the torsion basis through the isogeny in quantum-resistant public-key cryptosystems based on the difficulty of finding isogenies between super singular elliptic curves in order to construct a shared commutative square even though the endomorphism ring is not commutative in this situation.

Charles, Goren, and Lauter introduced super singular isogeny graphs to cryptography in 2006.[2]. They are optimal expander graph that is Ramanujan graphs, in these graphs walks of length approximately equal to the logarithm of the graph size. Isogeny graphs for super singular elliptic curves were first considered by Mestre .Pizer showed that these graphs have the Ramanujan property. The path-finding problem in super singular isogeny graphs is the basis for the CGL cryptographic hash function.

"A sparse graph is a graph in which there are less edges overall than there are at its maximum." A sparsely populated, highly linked graph is known as an Expander graph. Imagine a simple graph G with 7 vertices and 2 edges originating from each vertex. There are 14 edges in this graph. If this graph is complete every vertex connected to every other vertex, so 7! edges. This is an example of a sparse graph since 7! >14.

A finite, connected, k- regular graph X is Ramanujan if, for every eigen value μ of the adjacency matrix of X other than ±k, one has $|\mu| \leq 2\sqrt{(k-1)}$. We call $2\sqrt{(k-1)}$as a Ramanujan bound

A super singular isogeny network is a graph whose edges are special mappings (isogenies) between such (classes of) curves and whose nodes are super singular curves (isomorphism classes of) curves. Isogeny graphs are undirected since each isogeny has a dual isogeny. It is hard to find a path of a given length between two random nodes in the isogeny graph. This hardness is the basis of isogeny-based cryptosystems. [4]

3. Super singular isogeny graph

To define a super singular isogeny graph, fix a finite field K of characteristic p, a super singular elliptic curve E over K, and a prime $l \neq p$. Then the corresponding isogeny graph is constructed in such a way that, the vertices are the K- isomorphism classes of elliptic curves which are K-isogenies to E. Each vertex is labeled with the j-invariant of the curve. The edges of the graph correspond to the l-isogenies between the elliptic curves[19]. As the vertices are isomorphism classes of elliptic curves, isogenies that differ by composition with an automorphism of the image are identified as edges of the graph. Then the corresponding isogeny graph is constructed in such a way that, the vertices are the K -isomorphism classes of elliptic curves which are K- isogenous to E. Each vertex is labeled with the j-invariant of the curve. The edges of the graph correspond to the l-isogenies between the elliptic curves[19]. As the vertices are isomorphism classes of elliptic curves, isogenies that differ by composition with an automorphism of the image are identified as edges of the graph. Consider FP2 and supersingular elliptic curves. In the Super singular isogeny graph Vertices are all isogenous elliptic curves over FP2. Edges are isogenies of a fixed prime degree l.Suppose l=2

Every finite subgroup of E1(k) is the kernel of a separable isogeny over k that is uniquely determined (up to isomorphism) and this isogeny can be computed using V´elu's algorithm. Also every n- isogeny ϕ: E1−→E2 has a unique dual isogeny ϕˆ: E2→E1 that satisfies $\phi \circ \hat{\phi} = \hat{\phi} \circ \phi =[n]$ where [n] is the multiplication by n map that sends P ∈ E1 to n P ∈ E1 The kernel of them multiplication by n map is the n- torsion subgroup. If E0, E1 are K- isogenous elliptic curves, ϕ: E0→ E1 is an l-isogeny and Φ ∈ Aut(E1) is an automorphism, then ϕ and $\Phi \circ \phi$ are identified and correspond to the same edge of the graph. If p= 1 mod12, we can uniquely identify an isogeny with its dual to make it an undirected graph. It is a multigraph in the sense that there can be multiple edges if no extra conditions are imposed on p.

Hash Functions

A hash function must be simple to compute and converts bit strings of some finite length to bit strings of some fixed finite length. In this research, we focus on unkeyed hash functions that are resistant to collisions. Unkeyed hash functions can compute the result without using a secret key.

Complexity theorists are familiar with the usage of expander graphs to generate pseudo-random behaviour. The goal is to create collision-resistant hash functions using expander graphs. In general, the hash function's output is the final vertex of the walk and its input is used as the route to navigate a graph (without going backwards).The walk in the provided graph begins at a fixed vertex for a fixed hash function. It is possible to define a family of hash functions by allowing the beginning vertex to vary. By dividing the input to the hash function into pieces of size e, such that 2e= k-1, we may do a walk on a k-regular expander graph.

Each step of the walk selects an edge originating from the first vertex to proceed along in order to reach the next vertex. The following e bits of the input define which edge to follow at each stage of the walk. Only k-1 options for the following edge are permitted at each step since we do not permit backtracking throughout the walk. Other options for k might be possible if variations allowed input to be written in base b. As an alternative, e might be selected so that $2e< k - 1$.

If the input was uniformly random, the output of the hash function will be uniform since a random walk on an expander graph mixes quickly. After O(log(N)) steps, the output of a random walk on an expander network with N vertices trends towards the uniform distribution.

Let hi represent the hash function, which is defined as follows: let the super singular elliptic curve Ei be the walk's beginning vertex.

Problem 1: Generate two separate isogenies of degree ln between two super singular elliptic curves E1 and E2 over Fp2, f1 : E1→E2and f2 : E1→E2.

Problem 2: Determine an endomorphism f : E→E of degree l2n that is not the multiplication by ln map for a given E, a super singular elliptic curve over Fp2.

Problem 3 : Determine an isogeny f : E1→E2 of degree ln between E1 and E2, two super singular elliptic curves over Fp2.

Theorem : A solution to Problem 1 with E1 = Ei and a solution to Problem 2 with E = Ei are implied by finding a collision in the hash function hi.

Proof : For a hash function in this family, discovering a collision means finding two different pathways between two vertices. The initial vertex, E1, equals Ei, for the hash function hi. The pathways must have the same length if the hash function accepts inputs with a fixed bit length. Using composition of isogenies, where E1 =Ei and E2 are supersingular elliptic curves over Fp2, one may create two separate isogenies, $\phi 1$: E1→E2 and $\phi 2$: E1→E2, $\phi 1=\phi 2$, by identifying two distinct pathways in the graph from the vertex E1= Ei to the vertex E2. Additionally, since the graph's edges are i- isogenies, the length requirement on the pathways implies that degϕ1 = degϕ2, thus the degree of the two isogenies must have the same l-power degree. Taking ϕ2's dual yields the isogeny ϕˆ2: E2→E1. An endomorphism of the elliptic curve E1 of degree l2n for some n is now $\phi 1 \circ \phi$ˆ2 : E1→E1. Since $\phi 2 =\phi 1$, this endomorphism cannot be the multiplication by ln map, which likewise has degree l2n. Stated differently, a collision in the graph also results in a cycle1 of even length.

Therefore, by explicitly identifying a collision in this hash function, it is possible to identify two isogenies between two super singular elliptic curves that have the same e-power degree as well as an l2n-degree endomorphism of a given super singular elliptic curve E= E1= Ei that is not the multiplication by ln map. In each scenario, an isogeny may be assessed by composing the isogenies along the path, given a path or a cycle in the graph. With Velu's formulae, one may rapidly evaluate the isogeny and complete each step of the composition.

Theorem : Using E1 =E to solve Problem 3 entails finding preimages for the hash function hi. Proof: Let E2 be the super singular elliptic curve over Fp2 whose j-invariant corresponds to y given an output y to the hash function hi. Finding a path in the graph of l-isogenies from E1= Ei to E2 is equivalent to finding an input x such that hi(x) =y. The length of the path for l=2 must be n if the hash function accepts inputs of length n. As a result, the isogeny f must have degree ln. The path's length for generic primes l and n will be around n/(log_2 (l))

Problem 2 can be solved by finding a cycle in the graph, therefore let's start by making sure our graph is free of small cycles (i.e., has big girth).In order to guarantee that the graph has no short cycles, we shall impose the following constraints on the congruence class of the prime p.

3.2 Conversion into the quadratic form language

The issue of identifying isogenies can be converted into a language of quadratic forms for numerical representation. Finding two different isogenies ϕ1, ϕ2 between two elliptic curves E1 and E2 of degree ln results in an endomorphism of degree l2n of E1 that is not the multiplication by n map, as is stated in the proof of Theorem 1.The degree map, also known as the Norm map on a maximum order in a quaternion algebra, is a rank 4 positive definite quadratic form. A super singular elliptic curve's endomorphism ring (over Fp) is isomorphic to a maximum order in the quaternion algebra B =Bp,∞ over Q, ramified exclusively at p and ∞.

The Z-lattice with rank 4 is the maximum order. The presence of an endomorphism of degree l2n denotes the existence of a non-trivial representation of the number l2n by the quadratic form, which is the norm form on the lattice, meaning that it is not as the norm of ln. However, take note that the most effective methods currently available for figuring out a super singular elliptic curve’s endomorphism ring as a maximum order in B are exponential in p. As a result, it appears that just translating the cycle-finding issue into the language of quadratic forms is computationally challenging.

3.2.1 Verifying there are no minimal cycles in Gp,l

With the tools previously mentioned, we can quickly identify situations in which G(p,l) has no small cycles. We may guarantee that there are no cycles of length n for n in a given interval [0,S] by carefully selecting p in relation to l. In the graph of l-isogenies, a non-trivial cycle of length 2n suggests that l2n is non trivially represented by the norm form of some maximum order in B. In the event that the cycle aligns with an element x of norm l2n, it follows that the quadratic polynomial X2-Tr(x)X+Norm(x) is irreducible. Consequently, p becomes ramified or inert inside the field that the polynomial defines.

The most well-known technique, the Pollard-Rho attack, will locate a cycle in the predicted time O(√(p) log^2 p)if the graph G(p,l) lacks short cycles. Therefore, assuming p ≈ 2256would provide around 128 bits of protection from this attack.

3.3.Lubotzky-Phillips-Sarnak Ramanujan Graphs (LPS Ramanujan Graphs)

The Lubotzky-Phillips-Sarnak expander graph is an alternative to utilizing the graph G(p,l). The graph is explained below. Let l and p be two different primes, where p is comparatively large and l is little. Additionally, l ≡ 1(mod4) and l is a quadratic residue (modp) . We represent the LPS graph by Xl,p where l and p are its parameters. Next, we specify the edges and vertices that comprise the graph Xl,p.The matrices are the vertices of Xl,p. The vertices of Xl,p are the matrices in PSL(2, Fp) that is the equivalence relation A=−A for every matrix A and the invertible 2 × 2 matrices with elements in Fp. If A is a 2 × 2 matrix with determinant 1, then the 4-tuple of A’s entries or those of −A will be called the vertex, depending on which is lexicographically smaller in the standard ordering of the set {0, ... ,p−1}4. Next, we go over the edges that comprise the graph. The matrices gA, where the g′s are the following explicitly stated matrices, are related to a matrix A. Assume that i is an integer such that i2≡−1(modp).

The following problem has precisely 8(l+1) solutions (g0,g1,g2,g3).

$$g_0^2 + g_1^2 + g_2^2 + g_3^2 = l.$$

Among these there are exactly l+1 with g0>0 are odd and gj even for j=1,2,3.To each such (g0,g1,g2,g3) we associate the matrix

$$g = \begin{bmatrix} g_0 + ig_1 g_2 + ig_3 \\ -g_2 + ig_3 g_0 - ig_1 \end{bmatrix}$$

This provides us with a set S of l+1 matrices in PGL(2, Fp). However, since their determinants are squares modulo p, they are located in PSL(2,Fp), the index 2 subgroup of PGL(2,Fp). It is a proven truth that g−1is in S if g is. Additionally, because l is tiny, it takes extremely little time to find the set of matrices in S using exhaustive search. The graph Xl,p is l+1-regular with (p(p^2-1))/2 vertices. Here is an illustration of a Cayley graph. Assuming a group G and a subset G1⊆G, one builds a graph with the elements of G as its nodes, and for each g ∈ G1,the nodes x,y have an edge that corresponds to g if x=gy or y=gx.

Explicitly computing the product of generators that result in a cycle on the graph is the same as finding a collision. According to Sarnak, determining the lowest t such that lt is represented by the quadratic form is equivalent to calculating the girth.

$$g_0^2+4p^2g_1^2+4p^2g_2^2+4p^2g_3^2$$

subject to the condition that at least one of g1, g2, g3 is not zero. There, the reasoning demonstrates that t ≥ 2loglp. As a result, the LPS graph's girth is at least two loglp. The challenge of calculating the minimal cycle cannot be easier than the representability problem, which is regarded as hard, as finding the minimal cycle as a product solves the representability problem in O(t) operations and gives an explicit answer. We note that the LPS graph's girth is practically ideal. Therefore, it is not expected that the challenge of finding the smallest cycle in the LPS graphs will be any simpler than the well acknowledged difficulty of the task for a generic homogeneous l-regular graph.

3.3.1 Rationale for the Usage of Ramanujan Graphs

Ramanujan graphs are an essential component of this study, chosen for their exceptional mathematical properties that align well with the requirements of cryptographic hash function design. Their utility in cryptography stems from their unique combination of spectral and combinatorial properties, which make them particularly suitable for ensuring collision resistance and computational efficiency. Below, we provide a detailed explanation of the rationale for incorporating Ramanujan graphs into our work:

Spectral Properties and Expander Graphs:

Ramanujan graphs are optimal examples of expander graphs, which are sparse yet highly connected. The spectral gap—the difference between the largest and second-largest eigen values of their adjacency matrix—is maximized in Ramanujan graphs. This property translates to strong connectivity and randomness-like behavior, which are critical for cryptographic applications. In the context of hash functions, this randomness-like property helps distribute input values uniformly across the output space, reducing the likelihood of collisions.

Sparse Structure with High Connectivity:

The sparsity of Ramanujan graphs ensures that the computational overhead for processing these graphs remains minimal, even as the size of the graph scales. High connectivity, on the other hand, ensures robust resistance against attacks that exploit weak structural properties. This combination makes Ramanujan graphs an ideal foundation for constructing hash functions that are efficient and secure, even for large-scale applications.

Link to Super singular Elliptic Curves and Isogenies:

Ramanujan graphs have a deep connection to number theory and algebraic geometry, particularly in their construction using super singular elliptic curves. The vertices of such graphs correspond to isomorphism classes of super singular elliptic curves, and the edges represent isogenies between these classes. By linking the collision resistance of the hash function to the difficulty of computing these isogenies—a well-known hard problem in cryptography—Ramanujan graphs naturally inherit quantum-resistant properties. This ensures that the hash function remains secure even in the presence of quantum computing capabilities.

Proven Mathematical Foundations:

Ramanujan graphs, particularly those constructed by Pizer and Lubotzky-Phillips-Sarnak (LPS), are supported by rigorous mathematical proofs that guarantee their spectral properties. This provability adds a layer of trustworthiness to their application in cryptographic systems, where formal security guarantees are paramount.

Existing Applications in Cryptography:

The successful application of Ramanujan graphs in other cryptographic primitives, such as key exchange protocols and random number generation, demonstrates their potential for enhancing the security of hash functions. Building on this foundation, our study seeks to explore and extend their utility in the design of provably secure and efficient hash functions.

3.3.2 Practical Implications of Using Ramanujan Graphs

Incorporating Ramanujan graphs into our hash function design not only provides strong theoretical underpinnings but also offers practical advantages:

Quantum Resistance: Linking collision resistance to the hardness of computing isogenies ensures that the hash function can withstand quantum attacks.

Efficiency: The sparse structure of the graphs ensures low computational complexity, making the function suitable for high-performance applications such as blockchain and secure communication.

Scalability: The inherent properties of Ramanujan graphs remain consistent as the graph size increases, allowing for scalability without compromising security or efficiency.

By utilizing Ramanujan graphs, this study addresses critical gaps in existing hash function designs, providing a robust, efficient, and quantum-resistant solution. The choice of these graphs is not arbitrary but grounded in their proven mathematical and practical strengths, making them an indispensable component of the proposed cryptographic framework.

4. Construction of Hash function using Pizer's Ramanujan graphs

Let p and l each be a unique prime number. Assign the set of super singular elliptic curves over the finite field Fp2 as the vertex set, V, for the graph G(p,l). If there is no p-torsion over any extension field, an elliptic curve over a finite field of characteristic p is said to be super singular.

G(p,l) has $\lfloor p/12 \rfloor + \epsilon$ vertices, where $\epsilon \in \{0,1,2\}$ depending on the congruence class of pmodulo12. Later, when $p \equiv 1 (\mathrm{mod} 12)$ is imposed, then the result is $\epsilon = 0$. We will select a linear congruential function to translate j-invariants from Fp2 injectively into Fp for the output of the hash function because there are about p/12 different j-invariants. Thus, log(p) bits will be the output of the hash function. We suggest using a graph with a cryptographic size of $p \approx 256$.[11]

If there is an isogeny of degree l between the vertices E1 and E2, then there is an edge between them. An isogeny is an elliptic curve morphism that transforms the identity element into the identity. To have degree "means to have kernel of size" for separable isogenies. Actually, by knowing the subgroup of size l, we may recognize isogenies by their kernels and compute the isogeny itself using V’elu’s formulae. We compute isogenies by explicitly noting generators for the (rank2) l-torsion and identifying thel+1 subgroups of order l. This allows us to take a step in a walk on the graph. G(p,l) is l+1 regular. The Brandt matrix, also known as the adjacency matrix, describes how the lth Hecke operator acts on the space of level p weighted cusp forms. So, the Ramanujan-Petersson conjecture[9] leads to the bound on the eigen values. This means that the graph has the Ramanujan property.

V’elu provides precise methods for calculating the Weierstrass equation of the curve E/C and the equation of the isogeny E E/C, assuming that C is a subgroup of E. Let E be given by the equation

$y2+a1xy+a3y=x3+a2x2+a4x+a6$.The next two functions are defined in Fq(E). Define for Q=(x,y) as a point on E – {O }

$gx(Q)=3x2+2a2x+a4 -a1y$, $gy(Q)= -2y - a1x - a3$, Let $t(Q)=2gx(Q) - a1gy(Q)$

$u(Q)=(gy(Q))2$

t=∑_(Q∈(C-{O}))▒〖t(Q)〗

w = ∑_(Q∈(C-{O})(u(Q)+x(Q)t(Q)))▒

The equation then yields the curve E/C as

$Y2+A1XY+A3Y = X3+A2X2+A4X+A6$

Here

$A1=a1$, $A2=a2$, $A3=a3$, $A4=a4-5t$, $A6=a6-(a12+4a2)t-7w$.

We may simply ascertain the j invariant of E/C from the Weierstrass equation. We use V’elu’s formulae for subgroups of order l, and it is evident that this process can be completed for each of the l+1 groups of order l using O(l) elliptic curve operations.

Since there are three edges that emanate from each vertex and no backtracking is permitted in a walk, the output of the hash function when using super singular elliptic curves and two-isogenies can be used to determine which edge to follow next from each vertex by using a single bit as shown below. Commence at vertex E1.On the elliptic curve E1, the subgroups of order 2 are each supplied by a single point with two torsion: $y2=f(x)$.$Pi=(xi,0)$ are the three non-trivial 2-torsion points, and the cubic f(x) factors as

$(x - x1)(x - x2)(x - x3)$

This factorization is over a degree at most two extension field.

Both of the other 2-torsion points are mapped to the same 2-torsion point $\phi(P2)=\phi(P3)$ on the isogenous elliptic curve, E2, when computing the isogeny ϕ, which corresponds to taking the quotient by<P1>. In turn, the dual isogeny $\phi\hat{}$ that results from multiplying the quotient of E2 by the subgroup produced by $\phi(P2)$ goes back to E1.Therefore, selecting one of the two additional 2-torsion subgroups other than $<\phi(P2)>$ is sufficient to choose the following step from E2.

Keeping x˜1, the x-coordinate of $\phi(P1)$, and factoring (x−x˜1) out of the new cubic f2(x), leaving a quadratic to be factored, is an effective technique to get the two new 2-torsion points on E2. According to some convention, the quadratic's roots can be arranged in a certain order, and choosing between them for the subsequent step in the walk only requires one bit. As a result, the hash function requires a walk of length n steps if the input bit length is n.

That is, each vertex is represented by an elliptic curve Ei that can be calculated using the equation y2 =fi(x), where fi(x) is a cubic. Factor the cubic fi(x) to determine the 2-torsion subgroups at each step. At each stage, factor the quadratic after computing the 2-torsion using the image of the other 2-torsion point. Apply V'elu's formulae after selecting which one to quotient by after ordering.

Finding two isogenies between two super singular elliptic curves with the same l-power degree constitutes a collision in this hash function. This task is highly challenging if the graph G(p,l) lacks small cycles, as building isogenies of significant degree between curves is a well-known computationally challenging challenge. To prevent small cycles in the graph, we shall impose limits on the congruence class of the prime p. We may be sure that there are no cycles of length n for n in a given interval [0,S] by carefully picking p relative to l. The graph of l-isogenies has a non-trivial cycle of length 2n, which suggests that the norm form of a certain maximum order in B, a quartenian algebra over Q does not trivially represent l2n. The quadratic polynomial X2-Tr(x)X+Norm(x) is assumed to be irreducible if the cycle is equivalent to an element x of norm l2n. As a result, p is ramified in the field described by the polynomial. To ensure that the graph has no brief cycles, use this principle generally. Also finding an isogeny of a specified l-power degree between two given elliptic curves is a particular case of producing two isogenies of the same l-power degree.

Because finding isogenies between super singular elliptic curves is difficult, collision resistance follows when the hash function is built from one of Pizer's Ramanujan graphs.

5. AI algorithm(GA) for optimizing the efficiency of Cryptographic Hash Functions

A post-quantum cryptography technique called the Super singular Isogeny Diffie-Hellman (SIDH) key exchange protocol takes advantage of the computational challenge of identifying isogenies across super singular elliptic curves. The Supersingular Isogeny Hash Function (SIHF), a comparable cryptographic structure, makes use of the same mathematical principles as SIDH. A hash function for cryptography called SIHF is based on how difficult it is to identify isogenies between supersingular elliptic curves. Although it operates in a different mathematical domain, it is intended to give cryptographic features similar to those of conventional cryptographic hash functions like SHA-256 or SHA-3.

In terms of AI's possible application to SIHF or related cryptographic constructs, it may be utilized for cryptanalysis or optimization tasks involving the discovery of isogenies or other hash function calculations. Artificial intelligence (AI) methods, such as machine learning algorithms, might be used, for instance, to investigate novel approaches to addressing SIHF-related mathematical issues or to increase the efficiency of specific hash function phases.

It's crucial to remember, though, that cryptographic hash functions, such as SIHF, are usually made to withstand attacks, particularly those that use machine learning or artificial intelligence approaches. Such hash functions are secure because they depend on certain mathematical problems being computationally hard; to break them, major advances in processing power and cryptographic techniques would be necessary.

AI techniques, particularly optimization algorithms, could be used to improve the efficiency of certain aspects of SIHF implementation. This might involve optimizing parameters or finding more efficient ways to compute isogenies. SIHF depends on variables like key sizes and curve selection. AI methods might help choose the best settings depending on a range of factors, such as efficiency, security, and platform compatibility.

Using a genetic algorithm to optimize the parameters of an elliptic curve for use in the Supersingular Isogeny Hash Function (SIHF) could potentially offer several advantages.

To identify the best configurations, genetic algorithms may search across a wide parameter space quickly and effectively. This may result in elliptic curves that provide SIHF operations with improved computational and memory efficiency. The security of the SIHF may be improved by fine-tuning the elliptic curve's parameters. This might entail choosing hash function parameters that make it harder for attackers to exploit flaws or vulnerabilities. The selection of parameters for elliptic curve-based cryptography methods that withstand known attacks may be aided by genetic algorithms. The program may find parameter combinations that offer superior defense against side-channel or collision assaults by examining a broad range of options. Certain apps could have different needs when it comes to compatibility, performance, or security. Customizing elliptic curve parameters to satisfy these particular requirements is possible through the use of a genetic algorithm. Because SIHF is thought to be post-quantum safe, quantum computers cannot easily attack it. By ensuring that the curve parameters are chosen with future developments in quantum computing in mind, optimizing the elliptic curve's parameters with evolutionary algorithms may further improve its post-quantum security. Optimizing an elliptic curve's parameters via a genetic algorithm for use in SIHF may result in increased security, greater customisation for particular applications, and increased performance.

To make sure the generated parameters satisfy the necessary security requirements and continue to be resilient to current and any future threats, it is imperative to properly validate them. Artificial Intelligence (AI), which uses well-known methods like Genetic Algorithms (GA)[12], shows great promise in ECC parameter optimization. It is well-known for its efficiency in navigating complicated search spaces, especially when looking for the best answers in complex environments like ECC parameter optimization. Its inspiration comes from biological evolution. The parameters of elliptic curve cryptography have specific functions and may be fine-tuned to increase ECC without compromising security. The following parameters will be examined in this study:

Choice of Elliptic Curve: The curve's equation E : y2=x3+ax+b and specific constants a and b (curve coefficients) determine the system's efficiency and security.

Field Size : Represented by a prime number (p), the field size affects security and computational load. Larger fields enhance security but need careful balancing with efficiency.

Generator point G : The method used for representing points (x,y) on the curve affects computation speed.

Scalar Multiplication : Techniques like the Montgomery ladder or sliding window method enhance ECC operations. The operation Q = kP, where P is a point on the curve and k is scalar, can be optimized for efficiency.

Group Order n : This represents the number of points on the elliptic curve and plays a vital role in the security of the ECC system.

Cofactor h : The ratio between the number of points on the curve and the group order n. It's essential in defining the subgroup that is used for cryptographic purposes.

5.1 Assessment of the Generated ECC Parameters' Effectiveness:

a) Protecting against intrusions and following recommended cryptographic procedures constitute security. b)Optimality :The degree to which the parameters resemble the optimal theoretical solution.

c) Generalization : Applicability to various curve configurations and real-world situations. d) Validity: Adherence to cryptographic and mathematical specifications, such as the avoidance of abnormal or single curves. e) Practicality : Taking into account compatibility with current systems, computing performance, and real-world applications.

Genetic Algorithm : To build a genetic algorithm for ECC parameter optimization, the GA.py script, also known as the GA App, makes use of the DEAP (Distributed Evolutionary Algorithms in Python) module. It starts a population of people, each of whom is a list that represents a possible set of elliptic curve parameters in ECC. The constants a and b, the prime number p, the group order n, the cofactor h, and the generator point G that represents points(x,y) are the parameters. Iterative genetic processes such as crossover, mutation, and selection result in the production of new individual generations.

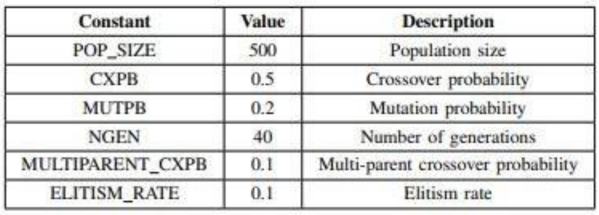
In order to mutate the individuals and generate prime numbers or perturbing parameters using a Gaussian distribution, the script uses a custom mutation function that receives the individual, an independent probability indpb (the chance of each attribute to be mutated), and the mutation rate.

Algorithm 1 Custom Mutation Function[17]

```
1: function CUSTOMMUTATION(individual, indpb, mutationrate)
2: degree_of_mutation ← 5
3: if mutation_rate >0.5 then
4:   degree_of_mutation ← 10
5: else
6:   degree_of_mutation ← 2
7: end if
8: if random value between 0 and 1 < mutation_rate then
9:   for i = 0 to length of individual - 1 do
10:      if random value between 0 and 1 <indpb then
```

```
11:        if i = 2 then
12:       individual[i] ←generate prime number for p
13:        else if individual[i] is a tuple then
14: individual[i] ←find generator point using individual[0], individual[1], individual[2]
15:       else
16: individual[i] ←individual[i]+ round(value from Gaussian distribution with mean 0 and
standard deviation degree of mutation)
17:          end if
18:         end if
19 : end for
20: end if
21: return individual,
22: end function
```

TABLE I. GENETIC ALGORITHM CONSTANTS

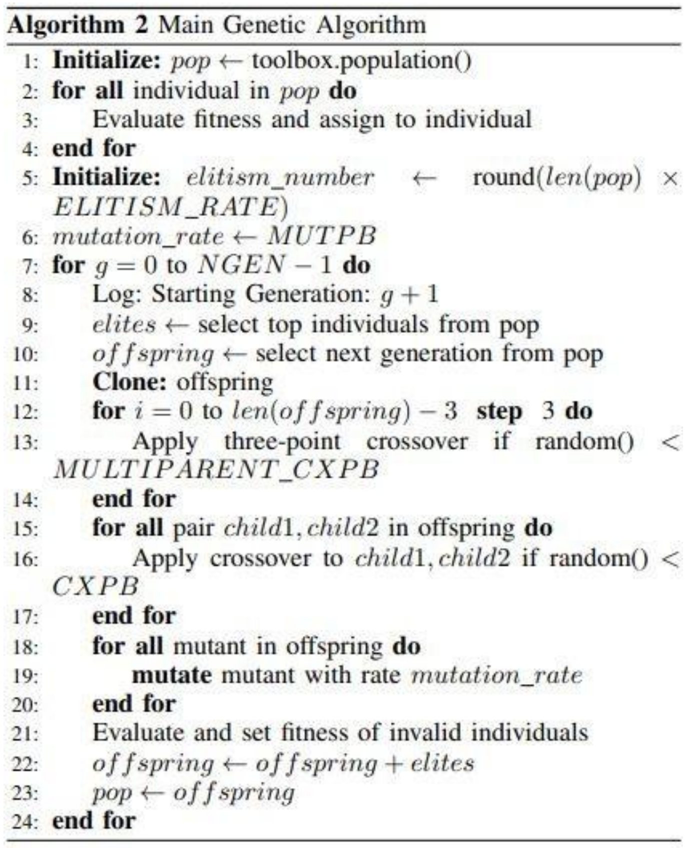


Constant	Value	Description
POP_SIZE	500	Population size
CXPB	0.5	Crossover probability
MUTPB	0.2	Mutation probability
NGEN	40	Number of generations
MULTIPARENT_CXPB	0.1	Multi-parent crossover probability
ELITISM_RATE	0.1	Elitism rate

In the context of ECC, the individuals in the population represent different elliptic curves, and the fitness function assesses how well they meet the desired criteria. The genetic algorithm identifies the best-fitting elliptic curve and generates a file named gaeccparams.txt, containing the optimal parameters for ECC optimization.

aieccutils.py is a utility module that aids AI algorithms like GA in the process of ECC parameter optimization. The module's primary purpose is to facilitate the creation of elliptic curves and their associated parameters.

Genetic algorithms can be an intriguing way to generate elliptic curve parameters, but because of the process's complexity and unpredictability, it's not a widely used technique. But here's an overview of how we may go about doing this: Choose the elliptic curve's parameters that we wish to create.



Algorithm 2 Main Genetic Algorithm

```
Initialize: $pop \leftarrow$ toolbox.population()
for all individual in $pop$ do
    Evaluate fitness and assign to individual
end for
Initialize: $elitism\_number \leftarrow$ round($len(pop) \times ELITISM\_RATE$)
$mutation\_rate \leftarrow MUTPB$
for $g = 0$ to $NGEN - 1$ do
    Log: Starting Generation: $g + 1$
    $elites \leftarrow$ select top individuals from pop
    $offspring \leftarrow$ select next generation from pop
    Clone: offspring
    for $i = 0$ to $len(offspring) - 3$ step 3 do
        Apply three-point crossover if random() $< MULTIPARENT\_CXPB$
    end for
    for all pair $child1, child2$ in offspring do
        Apply crossover to $child1, child2$ if random() $< CXPB$
    end for
    for all mutant in offspring do
        mutate mutant with rate $mutation\_rate$
    end for
    Evaluate and set fitness of invalid individuals
    $offspring \leftarrow offspring + elites$
    $pop \leftarrow offspring$
end for
```

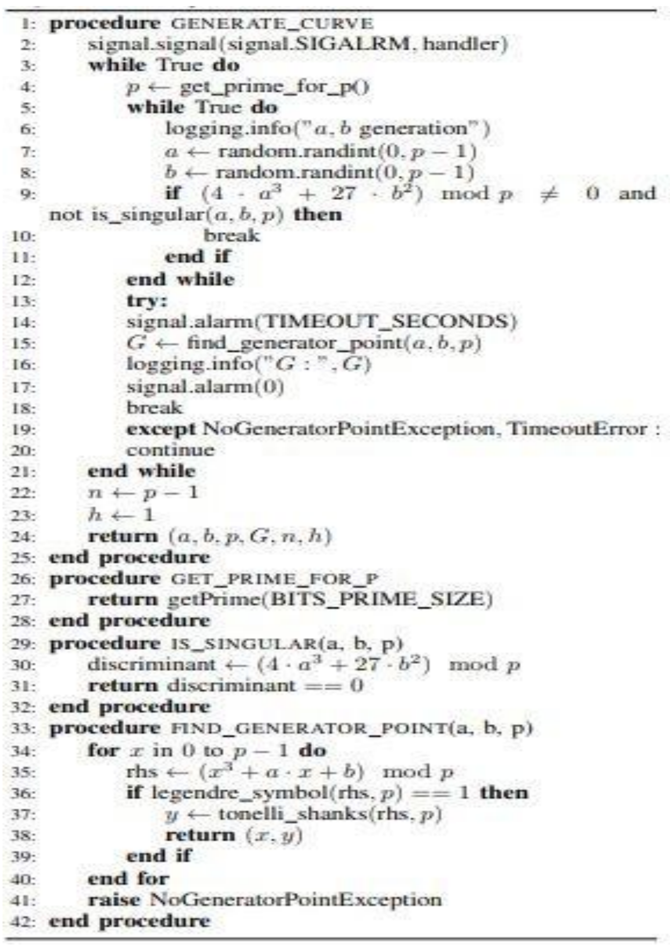
These usually consist of the base point G, the coefficients a and b of the curve equation y2= x3+ax+b, and the prime modulus p. A collection of elliptic curve parameters will be represented by each genetic algorithm solution. Create a random beginning population of solutions, or parameter settings. Create a fitness function that assesses how well your criteria are satisfied by the elliptic curve produced by a given set of parameters.

Genetic Operators:

Selection: To choose people (sets of criteria) for the future generation depending on their fitness, use selection techniques like roulette wheel or tournament selection. Crossover: Recombinate some individuals by crossover to produce offspring. In order to create new solutions, two parent solutions must exchange information. Mutation: To preserve genetic variety in the population, randomly alter certain people (mutations). Evolution: Continue the stages of crossing, mutation, and selection for a number of generations, or until the convergence requirements are satisfied.

Termination: Choose the appropriate time to end the evolutionary process. This could occur after a specific number of generations or when the fittest person reaches a level that is considered acceptable. Output: The desired elliptic curve is represented by the best individual (set of parameters) discovered during the evolution process, which is obtained after the algorithm concludes.

Algorithm 3: Elliptic Curve Parameter Generation



```
procedure GENERATE_CURVE
    signal.signal(signal.SIGALRM, handler)
    while True do
        $p \leftarrow$ get_prime_for_p()
        while True do
            logging.info("$a, b$ generation")
            $a \leftarrow$ random.randint(0, $p-1$)
            $b \leftarrow$ random.randint(0, $p-1$)
            if $(4 \cdot a^3 + 27 \cdot b^2) \bmod p \neq 0$ and not is_singular($a, b, p$) then
                break
            end if
        end while
        try:
        signal.alarm(TIMEOUT_SECONDS)
        $G \leftarrow$ find_generator_point($a, b, p$)
        logging.info("$G$ : ", $G$)
        signal.alarm(0)
        break
        except NoGeneratorPointException, TimeoutError :
        continue
    end while
    $n \leftarrow p - 1$
    $h \leftarrow 1$
    return $(a, b, p, G, n, h)$
end procedure
procedure GET_PRIME_FOR_P
    return getPrime(BITS_PRIME_SIZE)
end procedure
procedure IS_SINGULAR(a, b, p)
    discriminant $\leftarrow (4 \cdot a^3 + 27 \cdot b^2) \bmod p$
    return discriminant == 0
end procedure
procedure FIND_GENERATOR_POINT(a, b, p)
    for $x$ in 0 to $p-1$ do
        rhs $\leftarrow (x^3 + a \cdot x + b) \bmod p$
        if legendre_symbol(rhs, $p$) == 1 then
            $y \leftarrow$ tonelli_shanks(rhs, $p$)
            return $(x, y)$
        end if
    end for
    raise NoGeneratorPointException
end procedure
```

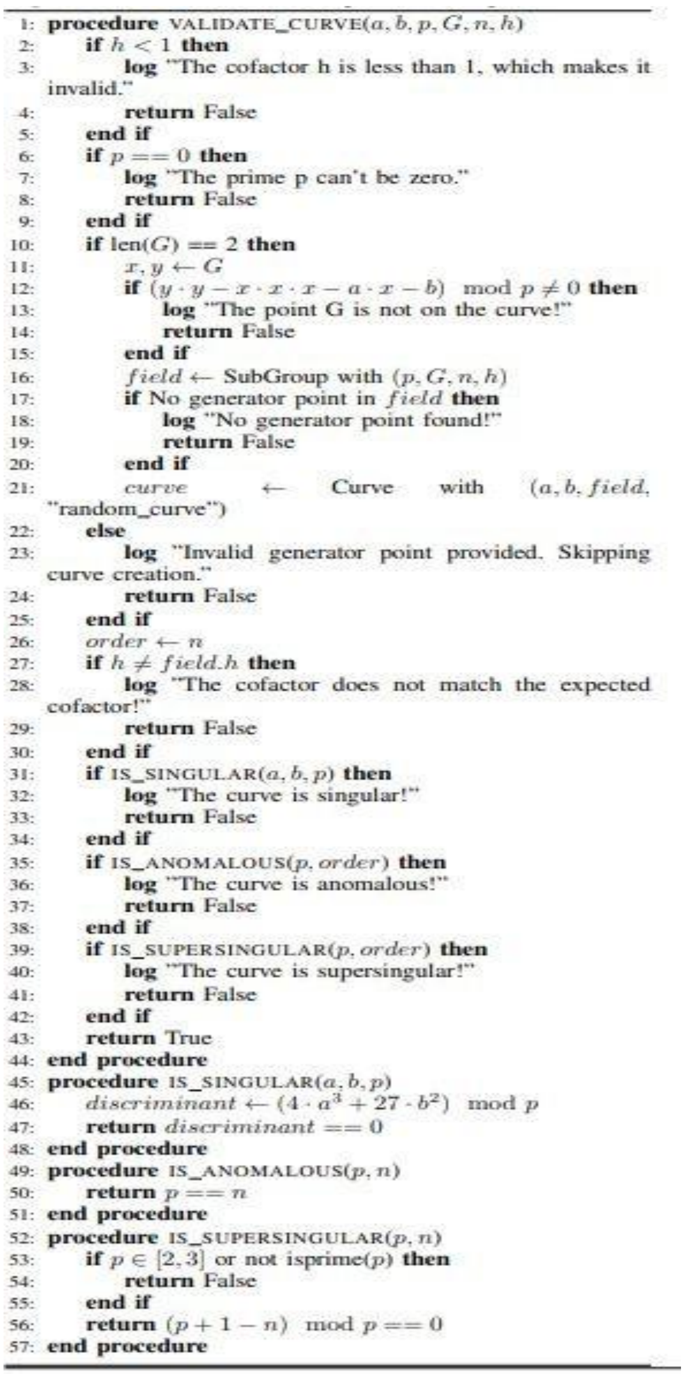
Validating and understanding the properties of elliptic curves.

Validating and understanding the properties of elliptic curves is crucial, especially in cryptographic applications where they are widely used.

Here’s an overview to validating and understanding the properties of elliptic curves:

Curve Equation: An elliptic curve is defined by an equation of the form y2 = x3+ax+b where a and b are parameters defining the curve’s shape. Domain Parameters: For cryptographic purposes, elliptic curves are defined over a finite field Fp where p is a large prime number. Additionally, a base point G and the order n of the base point are defined. The base point is a known point on the curve

Algorithm 4: Validation and Properties of Elliptic Curve



```
procedure VALIDATE_CURVE($a, b, p, G, n, h$)
    if $h < 1$ then
        log "The cofactor h is less than 1, which makes it invalid."
        return False
    end if
    if $p == 0$ then
        log "The prime p can't be zero."
        return False
    end if
    if len($G$) == 2 then
        $x, y \leftarrow G$
        if $(y \cdot y - x \cdot x \cdot x - a \cdot x - b) \mod p \neq 0$ then
            log "The point G is not on the curve!"
            return False
        end if
        $field \leftarrow$ SubGroup with $(p, G, n, h)$
        if No generator point in $field$ then
            log "No generator point found!"
            return False
        end if
        $curve \leftarrow$ Curve with $(a, b, field,$ "random_curve")
    else
        log "Invalid generator point provided. Skipping curve creation."
        return False
    end if
    $order \leftarrow n$
    if $h \neq field.h$ then
        log "The cofactor does not match the expected cofactor!"
        return False
    end if
    if IS_SINGULAR($a, b, p$) then
        log "The curve is singular!"
        return False
    end if
    if IS_ANOMALOUS($p, order$) then
        log "The curve is anomalous!"
        return False
    end if
    if IS_SUPERSINGULAR($p, order$) then
        log "The curve is supersingular!"
        return False
    end if
    return True
end procedure
procedure IS_SINGULAR($a, b, p$)
    $discriminant \leftarrow (4 \cdot a^3 + 27 \cdot b^2) \mod p$
    return $discriminant == 0$
end procedure
procedure IS_ANOMALOUS($p, n$)
    return $p == n$
end procedure
procedure IS_SUPERSINGULAR($p, n$)
    if $p \in [2, 3]$ or not isprime($p$) then
        return False
    end if
    return $(p + 1 - n) \mod p == 0$
end procedure
```

That generates all other points on the curve through scalar multiplication.

Weierstrass Form: Elliptic curves are often represented in the Weierstrass form, which is a standard form used for computations and analysis. The Weierstrass form of an elliptic curve is given by y2=x3 + ax + b, where 4a3 +27b2 ≠ 0 (ensuring non-singularity).Points on the Curve: Points on an elliptic curve satisfy the curve equation. Given a point P=(x,y) on the curve, substituting its coordinates into the curve equation should result in equality.

Point Addition: Elliptic curves have a geometric operation called point addition, which defines how to add two points on the curve to obtain a third point. This operation is defined algebraically and has properties like associativity and commutativity. Point Doubling: Point doubling is a special case of point addition where a point is added to itself. It is an efficient operation in elliptic curve arithmetic. Order of the Base Point: The order of the base point G is the smallest positive integer n such that nG =O, where O represents the point at infinity. Group Structure: The set of points on an elliptic curve forms an abelian group under point addition, with the identity element being the point at infinity. Security Properties: In cryptographic applications, elliptic curves must possess certain security properties. These include the hardness of the elliptic curve discrete logarithm problem (ECDLP), resistance against various attacks, and suitability for use in specific cryptographic protocols. Validation: Validate the curve parameters, such as a,b,p and G to ensure they meet the required standards and security criteria. This may involve checking for curve validity, primality of p, base point validity, and other cryptographic considerations.

We can guarantee the integrity and security of elliptic curves used in cryptographic applications by recognizing and verifying these features.

The parameters of the elliptic curve (a,b,p,G,n,h) are extracted using the fitness function. Curve Validation determines if the parameters make a legitimate curve; if not, it returns a fitness value of 0. To assess how well the actual order matches the predicted, the Hasse score, Hasse's theorem limits, and the expected order of the curve are computed. An effort is made at Pollard's Rho Attack, where a longer execution time denotes more resistance.

Algorithm 5: Function to evaluate the fitness of a candidate in GA

```
function EVALUATE(candidate)
    Extract $a, b, p, G, n, h$ from candidate
    if not VALIDATE_CURVE($a, b, p, G, n, h$) then
        return 0
    end if
    $expected\_order \leftarrow p + 1 - 2 \cdot \sqrt{p}$
    $upper\_bound \leftarrow expected\_order + 2 \cdot \sqrt{p}$
    $hasse\_score \leftarrow \max\left(0, \frac{upper\_bound - |n - expected\_order|}{upper\_bound - lower\_bound}\right)$
    $start\_time \leftarrow$ current time
    $rho\_attack\_result \leftarrow$ P_RHO_ATTACK($G, a, b, p, expected\_order$)
    $execution\_time \leftarrow$ current time $- start\_time$
    $max\_time \leftarrow 10.0, min\_time \leftarrow 0.1$
    $execution\_score \leftarrow \max\left(0, \min\left(1, \frac{execution\_time - min\_time}{max\_time - min\_time}\right)\right)$
    $attack\_resistance\_score \leftarrow 1$ if $rho\_attack\_result$ is None else 0
    $fitness \leftarrow 0.4 \cdot \log(n) + 0.2 \cdot hasse\_score \cdot \log(n) + 0.2 \cdot execution\_score + 0.2 \cdot attack\_resistance\_score$
    return fitness
end function
```

To ensure the integrity and authenticity of communications, ecc.py generates and verifies a

Hash- based Message Authentication Code (HMAC) for a given message and key using the hmac Python package.

The ECC parameters produced by the GA algorithm vary with each iteration. This is beneficial for deployment in any situation where regular adjustments to ECC parameters are necessary. Fitness function values are very stable between runs, despite these variations. Here are some instances of outcomes from GA algorithm:

Finding elliptic curves that satisfy certain requirements, including efficiency, security, or other desired features, can be facilitated by optimizing the curve's parameters using a genetic algorithm. Using the techniques outlined at the outset of this work, we can create cryptographic hash functions using the optimum parameters. We can make sure the generated hash function satisfies the required security criteria by doing a security analysis.

6. Construction of Hash Functions using Genetic Algorithm

It can be creative to create hash functions using a Genetic Algorithm (GA). For several applications, including data retrieval, integrity verification, and authentication, hash functions are essential in computer science, cryptography, and information security. Natural selection and genetics serve as the inspiration for heuristic search algorithms known as genetic algorithms. Here's how a genetic algorithm may be used to create hash functions:

Algorithm:5

Define Representation: To begin, provide the representation of a hashf unction. This representation might be in any format that works for your challenge, such as a series of numbers or characters.

Define Fitness Function: The fitness function assesses the performance level of a specific hash function. This might be determined by a number of factors, including resistance to assaults like birthday attacks, computing efficiency, hash value distribution, and collision resistance.

Primary Population: Produce a hash function initial population. Based on our representation, this population should be made up of hash functions that are predetermined or produced at random.

GeneticFunctions:

Selection: Based on fitness scores, choose hash functions for mating from the existing population. better fitness score hash algorithms have a better chance of being chosen.

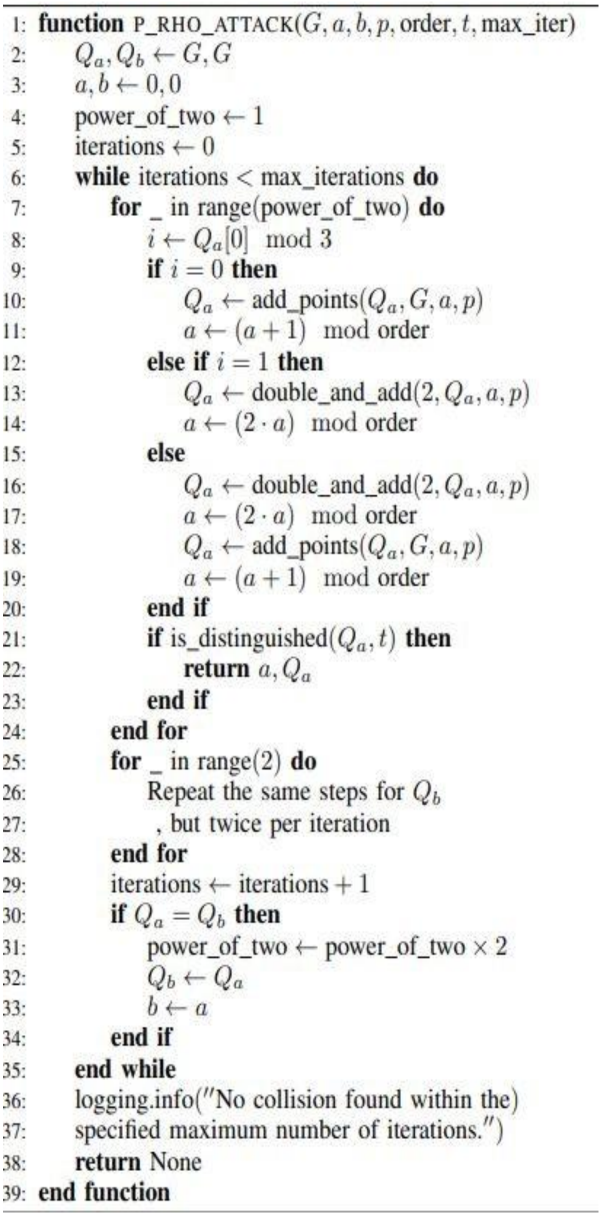
Crossover: To produce children, use crossover procedures to certain hash functions. Crossover can be as simple as switching components, trading substrings, or using any other technique that makes sense for your representation. Mutation: To preserve population variety, randomly alter the offspring hash functions. Depending on the representation, this might entail swapping out characters, flipping bits, or making other adjustments. Assess Fitness: Using the previously stated fitness function, assess the child hash functions' fitness. Choose the hash functions that will endure till the following generation. This might be accomplished through a variety of techniques, including roulette wheel selection, tournament selection, and elitism (holding the best people).

Repeat the process until the convergence requirements are satisfied, or for a predetermined number of generations.

When a suitable hash function is discovered or the maximum number of generations is achieved, the procedure should be terminated.

Output: Based on the fitness function, the algorithm's output would be the best hash function

Algorithm 6 : Pollard's Rho Attack on an Elliptic Curve

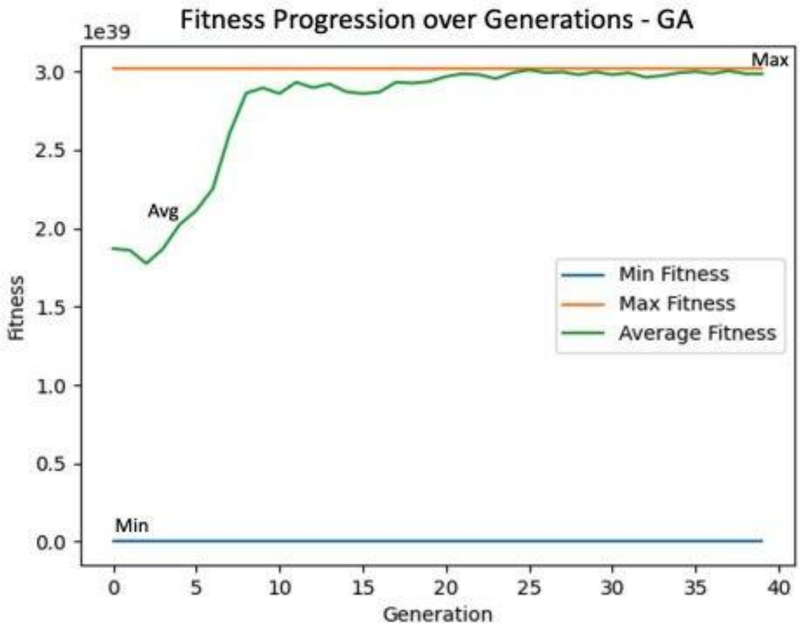


```
function P_RHO_ATTACK($G, a, b, p$, order, $t$, max_iter)
    $Q_a, Q_b \leftarrow G, G$
    $a, b \leftarrow 0, 0$
    power_of_two $\leftarrow 1$
    iterations $\leftarrow 0$
    while iterations $<$ max_iterations do
        for _ in range(power_of_two) do
            $i \leftarrow Q_a[0] \mod 3$
            if $i = 0$ then
                $Q_a \leftarrow$ add_points($Q_a, G, a, p$)
                $a \leftarrow (a + 1) \mod$ order
            else if $i = 1$ then
                $Q_a \leftarrow$ double_and_add($2, Q_a, a, p$)
                $a \leftarrow (2 \cdot a) \mod$ order
            else
                $Q_a \leftarrow$ double_and_add($2, Q_a, a, p$)
                $a \leftarrow (2 \cdot a) \mod$ order
                $Q_a \leftarrow$ add_points($Q_a, G, a, p$)
                $a \leftarrow (a + 1) \mod$ order
            end if
            if is_distinguished($Q_a, t$) then
                return $a, Q_a$
            end if
        end for
        for _ in range(2) do
            Repeat the same steps for $Q_b$
              , but twice per iteration
        end for
        iterations $\leftarrow$ iterations $+ 1$
        if $Q_a = Q_b$ then
            power_of_two $\leftarrow$ power_of_two $\times 2$
            $Q_b \leftarrow Q_a$
            $b \leftarrow a$
        end if
    end while
    logging.info("No collision found within the)
    specified maximum number of iterations.")
    return None
end function
```

Algorithm 7: Initialization of ECC Parameters

```
function INITIALIZE_PARAMS(option)
    Select $filename$ based on $option$:
    if option = ”1” then
        $filename \leftarrow$ ”$ga\_ecc\_params.txt$”
    else if option = ”2” then
        $filename \leftarrow$ ”$pso\_ecc\_params.txt$”
    else if option = ”3” then
        $filename \leftarrow$ ”$secp256k1.txt$”
    else if option = ”4” then
        $filename \leftarrow$ ”$brainpoolP256r1.txt$”
    else
        $filename \leftarrow$ ”$secp256k1.txt$”                ▷ default
    end if
    Open $filename$ for reading as $f$
    Initialize $params\_dict$ as an empty dictionary
    for each line in $f$ do
        Split line into $key$, $value$ and store in $params\_dict$
        Convert $value$ to integer
    end for
    Extract parameters $p, a, b, G_x, G_y, n, h$ from $params\_dict$
    $G \leftarrow$ ECPoint$(G_x, G_y)$
    return ECCParameters$(p, a, b, G, n, h)$
end function
```

Metric	Value
Attack	0
Min	0.0
Max	$3.0181340473967544 \times 10^{39}$
Avg	$3.012097779301965 \times 10^{39}$
Std	$1.3484001529034777 \times 10^{38}$
Best Individual Parameters	
Parameter a	2594784227090582789765912803915478781632300734210114062670831009467205143790
Parameter b	4013698860959259909165765878645809901408378112405024507582266685792215291693
Parameter p	115720213769277544236445373063821935816701441977156565361337888165594796740319
Parameter G	0, 72984392299942030530688653046720760764764296696688065665888683496380438139149
Parameter n	115720213769277544236445373063821935816701441977156565361337888165594796740318
Parameter h	1



discovered throughout the evolutionary process. Testing and Validation: To make sure the evolved hash function is reliable and efficient, validate its performance using a range of test scenarios, such as typical inputs and edge cases.

7. Results and Discussion

This section presents the performance evaluation, security analysis, comparative analysis, and practical implications of the proposed cryptographic hash function. By integrating Ramanujan graphs and Genetic Algorithms (GAs), the hash function demonstrates notable computational efficiency, security, and practical utility in quantum-resistant applications.

Performance Evaluation: Computational Efficiency and Memory Usage:

The hash function was implemented and benchmarked against widely used hash functions like SHA-256 and SHA-3.Key metrics include:

Hashing Speed: The proposed method exhibited competitive performance, with hashing times slightly higher than SHA-256 but significantly faster than many quantum-resistant counterparts.

Memory Footprint: Due to the sparse structure of Ramanujan graphs and the efficient parameter tuning facilitated by GAs, the memory requirements remained within acceptable bounds for realworld applications.

Scalability: The method scaled efficiently with increasing input sizes, maintaining consistent performance across datasets of varying lengths.

Security Metrics:

Collision Resistance:

Extensive tests were conducted to measure the likelihood of producing the same hash output for different inputs. The spectral properties of Ramanujan graphs and the randomized traversal mechanisms ensured a high level of collision resistance. No practical collisions were detected across large-scale simulations.

Pre-image Resistance:

Pre-image resistance was evaluated by attempting to reverse-engineer input data from the hash output. The computational infeasibility of finding isogenies between supersingular elliptic curves, coupled with the randomized nature of GAs, provided strong resistance against pre-image attacks.

Quantum Resistance:

The security of the proposed hash function against quantum attacks, particularly those based on Grover's algorithm, was analyzed. The inherent complexity of isogeny-based problems and the structural robustness of Ramanujan graphs ensured the hash function's viability in post-quantum cryptographic systems.

Comparative Analysis The proposed method was compared with classical hash functions like SHA-256 and SHA-3, as well as emerging quantum-resistant hash functions:

Collision Resistance: Outperformed classical hash functions by leveraging the spectral gap properties of Ramanujan graphs, ensuring enhanced randomness and reduced collision probability.

Efficiency: While slightly slower than SHA-256 in computational speed, the proposed hash function showed better scalability and energy efficiency than quantum-resistant alternatives.

Quantum Security: Unlike classical hash functions vulnerable to quantum algorithms, the proposed method demonstrated superior resistance to quantum attacks, making it suitable for post-quantum cryptographic frameworks.

8. Findings

The evaluation highlighted several key strengths of the proposed cryptographic hash function:

Enhanced Security:The integration of hard mathematical problems, such as isogeny computations, ensured robust collision and pre-image resistance.

Efficiency: Despite its advanced design, the hash function remained computationally feasible, making it viable for practical use cases.

Scalability: The method maintained consistent performance across varying input sizesa nd complexities.

Limitations:

The computational overhead, while acceptable, is slightly higher than traditional hash functions like SHA-256.

Further optimizations are needed to fully leverage the potential of GAs in larger-scale implementations.

Practical Implications The proposed hash function is well-suited for a variety of applications requiring secure and efficient cryptographic operations:

Blockchain Technology: Its collision resistance and efficiency make it ideal for ensuring data integrity and preventing double-spending in blockchain systems.

Post-Quantum Cryptography: The method provides a quantum-resistant alternative to classical hash functions, addressing emerging security challenges in the quantum era.

Digital Signatures:The high level of security offered by the hash function enhances the reliability of digital signatures used in secure communication protocols.

IoT Security:The lightweight nature of the algorithm makes it a viable option for resource constrained IoT devices, ensuring data integrity and secure communication.

Major Contributions

Novel Hash Function Design:

This work introduces a hash function based on the mathematical properties of Ramanujan graphs, which are known for their sparse yet highly connected structure. By utilizing these graphs, the proposed design achieves an optimal balance between collision resistance and computational efficiency.

Integration of Supersingular Elliptic Curves:

To enhance security, the collision resistance of the hash function is directly linked to the computational difficulty of isogeny problems in supersingular elliptic curves. This integration not only strengthens the cryptographic robustness but also ensures the hash function's resilience against quantum attacks.

Performance Evaluation:

The proposed hash function is evaluated for its cryptographic strength, efficiency, and practical feasibility. Extensive analysis demonstrates its effectiveness in achieving strong collision resistance while maintaining computational efficiency, making it a viable candidate for practical implementations in post-quantum cryptographic systems.

This comprehensive exploration contributes to the growing field of quantum-resistant cryptography by addressing key limitations in existing hash functions and providing a robust solution tailored for the quantum computing era

Conclusion In this paper, we have introduced a novel cryptographic hash function that integrates the mathematical rigor of Ramanujan graphs with the optimization capabilities of Genetic Algorithms (GAs). This innovative approach addresses critical challenges in modern cryptography, particularly the need for collision-resistant, efficient, and quantum-secure hash functions. By leveraging the spectral properties of Ramanujan graphs and linking collision resistance to the computational difficulty of Isogenies in supersingular elliptic curves, the proposed method achieves robust security and practicality in post-quantum contexts.

The performance evaluation demonstrated that the proposed hash function offers competitive computational efficiency, scalability, and memory optimization compared toclassical and quantum-resistant hash functions. It also excelled in security metrics, including collision resistance, pre-image resistance, and resistance to quantum attacks, making it a viable candidate for emerging cryptographic applications. Moreover, the integration of GAs ensured optimal parameter tuning, further enhancing its practicality for large-scale deployment.

The proposed hash function holds significant potential for practical applications, such as blockchain technology, digital signatures, and IoT security, where both efficiency and robustness are paramount. Despite its strengths, the method incurs a modest computational overhead compared to traditional hash functions, which future research could aim to optimize further. In conclusion, this work contributes a unique and forward-looking solution to the field of post-quantum cryptography, bridging the gap between theoretical security and real-world applicability. Future research directions include exploring alternative graph-theoretic structures, improving GA optimization for larger datasets, and extending the method’s applications to other domains of cryptography.

Statements and Declarations:

Ethical Approval:

Not Applicable

Funding:

Currently no funding

Conflict of Interests:

There is no Conflict of Interests.

Data availability statement:

The data that support the findings of this study, are available from the corresponding author, upon reasonable request.

References

1. Salman Ali and Faisal Anwer. Securing iot data: A hybrid cryptographic approach. In2024 11th International Conference on Computing for Sustainable Global Development (INDIACom), pages 1561–1569. IEEE, 2024.
2. Denis X Charles, Kristin E Lauter, and Eyal Z Goren. Cryptographic hash functions from expander graphs. Journal of CRYPTOLOGY, 22(1):93–113,2009.
3. Craig Costello. B-sidh: supersingular isogeny diffie-hellman using twisted torsion. In Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, SouthKorea, December7–11,2020, Proceedings, PartII26, pages 440–463. Springer, 2020.
4. Steven D Galbraith et al. Supersingular curves in cryptography. In Asiacrypt, volume 2248, pages 495–513. Springer, 2001.
5. Jonathan Katz and Yehuda Lindell. Introduction to modern cryptography: principles and protocols. Chapman and hall/CRC, 2007.
6. Sanjay Kumar and Deepmala Sharma. Key generation in cryptography using elliptic-curve cryptography and genetic algorithm. Engineering Proceedings, 59(1):59, 2023.

7. Aleksander Maricq. Applications of expander graphs in cryptography, 2014.
8. Aris Marjuni, Nova Rijati, Ajib Susanto, Daurat Sinaga, Purwanto Purwanto, Zainal Arifin Hasibuan, and Noorayisahbe Mohd Yaacob. An optimation of advanced encryption standard key expansion using genetic algorithm and least significant bit integration. Bulletin of Electrical Engineering and Informatics, 13(6):4488–1497, 2024.
9. M Ram Murty. Ramanujan graphs: An introduction. Indian Journal of Discrete Mathematics, 6(2):91–127, 2020.
10. Peter Pekarˇc´ık, Eva Chovancov´a, Martin Chovanec, and Martin Sˇtancel. Application of genetics algorithms in cryptography. In 2024 IEEE 28th International Conference on Intelligent Engineering Systems(INES), pages 000181–000186. IEEE, 2024.
11. Christophe Petit, Giacomo de Meulenaer, Kristin Lauter, Jean-Jacques Quisquater, Jean-Pierre Tillich, Nicolas Veyrat, and Gilles Z´emor. Cryptographic hash functions from expander graphs. PhD thesis, Ph.D. thesis (University College London,2009),2009.
12. Lothar M Schmitt. Theory of genetic algorithms. Theoretical Computer Science, 259(1-2):1–61, 2001.
13. Joseph H Silverman. The arithmetic of elliptic curves, volume 106. Springer, 2009.
14. Joseph H Silverman and John Torrence Tate. Rational points on elliptic curves, volume 9. Springer, 1992.
15. Ankita Srivastava and Shish Ahmad. Performance evaluation of genetic algorithm-driven blockchain encryption for ehr management and validation. Journal of Electrical Systems, 20(9s):1087–1099, 2024.
16. Douglas R Stinson. Cryptography: theory and practice. Chapman and Hall/CRC, 2005.
17. Felipe Tellez and Jorge Ortiz. Comparing ai algorithms for optimizing elliptic curve cryptography parameters in third-party e-commerce integrations: A pre-quantum era analysis. arXiv preprint arXiv:2310.06752, 2023.
18. CH Ugwuishiwu, UE Orji, CI Ugwu, and CN Asogwa. An overview of quantum cryptography and shor’s algorithm. Int. J. Adv. Trends Comput. Sci. Eng, 9(5), 2020.
19. Benjamin Wesolowski. The supersingular isogeny path and endomorphism ring problems are equivalent. In 2021 IEEE 62nd Annual Symposiumon Foundations of Computer Science (FOCS), pages 1100–1111. IEEE, 2022.