cryptography

Review

Elliptic Curve Cryptography with Machine Learning

Jihane Jebrane [1], Akram Chhaybi [1], Saiida Lazaar [1] and Abderrahmane Nitaj [2,*]

[1] Mathematics, Computer Science and Applications TEAM, Abdelmalek Essaâdi University, ENSA, Tangier 90000, Morocco; jihanejbr1@gmail.com (J.J.); akramchhaybi1@gmail.com (A.C.); slazaar@uae.ac.ma (S.L.)

[2] Department of Mathematics, Normandie University, UNICAEN, CNRS, LMNO, 14000 Caen, France

* Correspondence: abderrahmane.nitaj@unicaen.fr

Abstract: Elliptic Curve Cryptography (ECC) is a technology based on the arithmetic of elliptic curves used to build strong and efficient cryptosystems and infrastructures. Several ECC systems, such as the Diffie–Hellman key exchange and the Elliptic Curve Digital Signature Algorithm, are deployed in real-life applications to enhance the security and efficiency of digital transactions. ECC has gained even more importance since the introduction of Bitcoin, the peer-to-peer electronic cash system, by Satoshi Nakamoto in 2008. In parallel, the integration of artificial intelligence, particularly machine learning, in various applications has increased the demand for robust cryptographic systems to ensure safety and security. In this paper, we present an overview of machine learning and Elliptic Curve Cryptography algorithms. We begin with a detailed review of the main ECC systems and evaluate their efficiency and security. Subsequently, we investigate potential applications of machine learning-based techniques to enhance the security and performance of ECC. This study includes the generation of optimal parameters for ECC systems using machine learning algorithms.

Keywords: elliptic curve cryptography; artificial intelligence; machine learning

check for updates

Academic Editor: Josef Pieprzyk

Received: 28 October 2024
Revised: 25 December 2024
Accepted: 26 December 2024
Published: 30 December 2024

Citation: Jebrane, J.; Chhaybi, A.; Lazaar, S.; Nitaj, A. Elliptic Curve Cryptography with Machine Learning. *Cryptography* **2025**, *9*, 3. https://doi.org/10.3390/cryptography9010003

1. Introduction

In cryptography, the security of a cryptosystem is often based on the hardness of a known and believed hard problem, such as factorization, discrete logarithm, and Learning With Errors (LWEs). Some of such hard problems could be solved with the help of algorithms implemented in large-scale quantum computers. A typical example is Shor's algorithm [1], which could break the most popular and most widely used public key cryptosystems, such as RSA [2] and Elliptic Curve Cryptography (ECC) [3,4].

Introduced independently by Koblitz [3] and Miller [4] in 1984, ECC is a subfield of asymmetric cryptography. It uses the algebraic properties of elliptic curves over finite fields, and its security is based on the hardness of the Elliptic Curve Discrete Logarithm Problem (ECDLP). ECC allows key exchange [5], encryption and decryption [6], digital signature [7], random number generation [8], and requires smaller key sizes compared with other asymmetric systems such as RSA. ECC is used in industrial applications such as the Bitcoin digital currency [9], the security of the transport layer [10], and various communication services.

The use of machine learning techniques in cryptography and security is still a rapidly evolving topic. Nevertheless, machine learning has already been deployed in certain applications, mainly for security issues. In recent years, machine learning algorithms have been used to implement and enhance the efficiency and security of various cryptographic systems. These algorithms are applied to analyze cryptosystems, detect intrusions, test the security of systems, and perform cryptanalysis.

The connection between machine learning (ML) and cryptography was first discussed by Rivest [11] in 1991. Since then, various intersections between the two fields have been extensively studied, covering both cryptography and cryptanalysis, the two subfields of cryptology. In cryptography, the schemes proposed in [12–14] are based on neural network models, while the schemes proposed in [15,16] are based on deep learning.

ML is employed to select optimal secret keys for use in encryption and decryption in a symmetric system, as well as optimal public keys for encryption in an asymmetric system [17–20]. ML is also utilized to observe the algebraic properties of encrypted data and to test the vulnerabilities of cryptographic systems [21]. Furthermore, it helps to understand the weaknesses and vulnerabilities of security and privacy and develop resilient defenses [22]. Various machine learning algorithms are also leveraged to build effective intrusion detection software packages, targeting both intrusions and attacks [23,24].

In cryptanalysis applications, Alani [25] introduced an attack on DES and Triple-DES based on a neural network. In 2015, Maghrebi et al. [26] proposed a method to apply deep learning in side-channel attacks.

In the ECC field, there are plenty of schemes for which implementation as well as security are challenging tasks. In [27], Tellez and Ortíz presented a study for possible applications of the Genetic Algorithm (GA) and the Particle Swarm Optimization (PSO), two artificial intelligence (AI) algorithms, to generate strong parameters for ECC. In [28], Villegas and Cordero presented an experimental evaluation of the resistance of ECC to simple power attacks using ML models. In [29], Weissbart et al. presented several attacks on the Edwards Digital Signature Algorithm (EdDSA) using machine learning techniques. In [30], Wøien et al. presented a neural network model for asymmetric encryption, focusing on algorithms in ECC. In [31], the performance of the execution time, the energy consumption, and the memory usage of the encryption/decryption algorithms of several lightweight cryptographic systems are studied using machine learning models.

In this paper, the main objective is to study how Elliptic Curve Cryptography can be performed with the support of machine learning. Section 2 provides an overview of the main concepts of artificial intelligence and machine learning. Section 3 introduces the arithmetical theory of elliptic curves. Section 4 examines elliptic curve cryptography. Section 5 discusses the main attacks on ECC. Section 6 explores the application of machine learning in the field of ECC. Section 7 summarizes and concludes this paper.

2. Artificial Intelligence and Machine Learning

AI is a combination of science and technology. It is based on several disciplines in engineering and mathematics, such as algebra, statistics, probability, and chaos theory. Other fields, including biology, computer science, information theory, and linguistics, also contribute to AI. Today, AI is applied across various fields such as vision systems, gaming, finance and banking, healthcare, language processing and recognition, self-driving vehicles, pharmaceutical discovery, chatbots, robotics, computer vision, data analysis, and cybersecurity.

2.1. Overview of Machine Learning

ML is a subfield of AI focused on creating, testing, and adapting computer procedures, algorithms, and programs that can automatically improve by learning from past experiences. It is used in various applications, such as financial fraud detection, healthcare report analysis, agricultural optimization, information dissemination, financial investment optimization, traffic prediction, and language translation.

There are three categories of machine learning algorithms: supervised, reinforcement, and unsupervised.

- Supervised learning. In supervised learning, the machine is under the supervision of an operator. The input and the output datasets are labeled and known to the operator and are proposed to the algorithm that is implemented in the machine. The task of the algorithm is to find a link between the input and the output datasets. To this end, the algorithm must identify patterns from the input dataset, learn from former statistical occurrences, and propose predictions. If the predictions are far from correct, then some parts of the algorithm are improved. This process continues until the predictions are acceptable, and the errors are sufficiently minimized. To improve the algorithm, several techniques are used such as classification, linear regression, and forecasting. The ultimate goal is that the algorithm can make correct predictions on any unseen data. A typical application of supervised learning is fraud detection. Fraudulent and suspicious transactions can be detected by the algorithm using stored data.
- Reinforcement learning. In this category of machine learning, the algorithm is trained to take certain accurate actions. This can be accomplished by rewarding the good actions and blaming the bad ones. To be accurate, the algorithm learns from experiences how to achieve a goal in an optimal way through interactions with the environment. The algorithm has to discover the actions that are desired or not. A typical example of reinforcement learning is autonomous driving. A solid autonomous driver must analyze and make several decisions and behaviors in various situations such as finding an optimal path, avoiding dense traffic, predicting travel time, and driving safely.
- Unsupervised learning. In unsupervised learning, the machine is independent of any human operator. The machine learning algorithm analyses and clusters the unlabeled datasets without the need for human help or intervention. The clustering technique permits the discovery of the hidden patterns and groups of unlabeled datasets based on their categories, similarities, and differences. The goal of unsupervised learning is to group the datasets into clusters that are more organized within an optimal number of classes. A typical application of unsupervised learning is customer segmentation by commercial companies. They can use an unsupervised learning algorithm to categorize their customer's common needs and cluster them into categories to propose their products to potential buyers.

2.2. Overview of Perceptron and Multilayer Perceptron

The perceptron is a basic supervised learning algorithm and the simplest type of artificial neural network, invented by Rosenblatt in 1958 [32]. There are two families of perceptrons: single-layer perceptrons, which can process only linear activation functions, and multilayer perceptrons, which can process nonlinear activation functions.

A single-layer perceptron is designed to categorize several binary inputs and give one binary output, generally 0 or 1. It is composed of several basic components, including an input layer, weights, a bias, an activation function, and a single output layer (see Figure 1). The perceptron starts by taking the bias, and a list of scalar input features. A weight is assigned to each input, and a linear combination of all couples (input, weight) is processed. The result of the linear combination is added to the bias, and introduced into the activation function, which decides to what category belong the input features. Typically, if the input features are $(x_0, x_2, \dots, x_n)$, the weights are $(w_1, w_2, \dots, w_n)$, the bias is b, and the function is f, then the output is

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right).$$

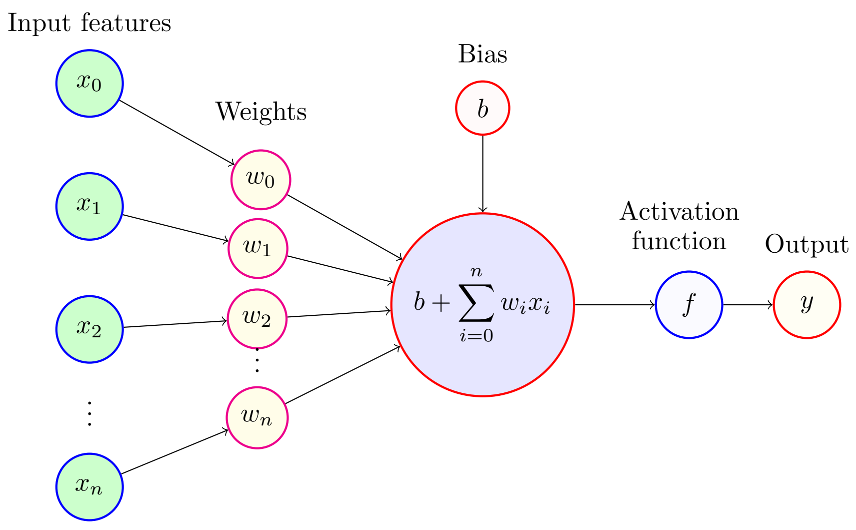

Figure 1. Single layer perceptron.

A multilayer perceptron is an artificial neural network that can process all kinds of data, including nonlinearly separable data. It is composed of an input layer, one or more hidden layers, and one output layer. The input layer is composed of one or more nodes where the initial input data is introduced. The hidden layers are also composed of one or more nodes. Each node in a hidden layer receives inputs from all the nodes of the previous layer. The information is processed and passed to the nodes of the next layer. At the end, the output layer receives the final inputs and produces the final output. The output layer is composed of a number of nodes, which represents the number of possible classes of featured information (see Figure 2).

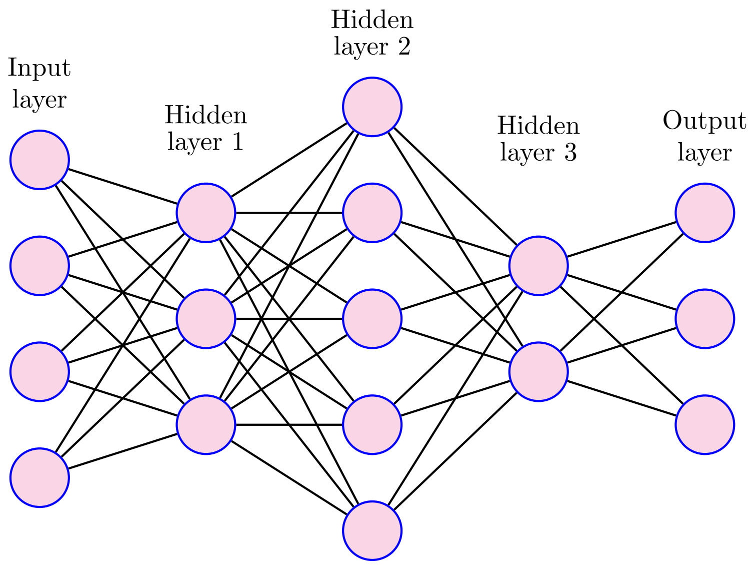

Figure 2. Multilayer perceptron.

Multilayer perceptrons are used in various applications such as speech and image recognition, banking, e-commerce, banking, and travel.

2.3. Overview of Artificial Neural Networks

Neural Networks are modern algorithms at the heart of machine learning, inspired by the human brain. They mimic the functioning of biological neurons to analyze tasks and propose solutions. A neural network is composed of a sequence of layers of nodes, namely, input layers, hidden layers, and output layers (see Figure 2). The data is introduced in the input layers and is processed in the hidden layers using activation functions. Finally, predictions are made by the output layers.

The nodes in two adjacent layers are connected, and the connections are guided by weights. Moreover, each node has an associated bias. The weights and biases are adjusted during the training phase of the neural network through feedforward and backpropagation. These adjusted weights and biases enable each node to optimize its computations.

There are various types of neural networks such as Generative Adversarial Networks (GANs), Convolutional Neural Networks (CNNs), Feedforward Neural Networks (FNNs), and Recurrent Neural Networks (RNNs).

3. Elliptic Curves over Finite Prime Fields

In this section, we give an overview of the elliptic curves over a finite prime field.

3.1. The Arithmetic of the Elliptic Curves

Let p be a prime number, and $\mathbb{F}_p$ be the finite prime field with p elements. Let $a_1, a_2, a_3, a_4, a_6 \in \mathbb{F}_p$. An elliptic curve E over $\mathbb{F}_p$ is the set of all elements $(x, y) \in \mathbb{F}_p^2$ such that the following Weierstrass equation is satisfied

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6.$$

For $p > 3$, the equation can be transformed into a short Weierstrass form

$$y^2 = x^3 + ax + b.$$

The requirement $4a^3 + 27b^2 \neq 0$ ensures that E is nonsingular. The solutions are often denoted as points $P = (x, y)$. The set of rational points of E, together with a specific point $\mathcal{O}$, called the point at infinity, is denoted $E(\mathbb{F}_p)$. The set $E(\mathbb{F}_p)$ has the structure of an Abelian group with the addition law, where $\mathcal{O}$ is the neutral element. The addition law uses the chord-tangent process. The following cases resume the addition law:

1. For all $P \in E(\mathbb{F}_p)$, $P + \mathcal{O} = \mathcal{O} + P = P$.
2. For all $P = (x, y) \in E(\mathbb{F}_p)$, $-P = (x, -y)$ is the opposite point of P such that $P + (-P) = \mathcal{O}$.
3. For all $P_1 = (x_1, y_1) \in E(\mathbb{F}_p)$ and $P_2 = (x_2, y_2) \in E(\mathbb{F}_p)$ with $P_2 \neq -P_1$, the sum of P_1 and P_2 is $P_3 = (x_3, y_3)$ with

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2, \\ y_3 &= \lambda(x_1 - x_3) - y_1, \end{aligned}$$

   where $\lambda = \frac{y_1 - y_2}{x_1 - x_2}$.
4. For all $P = (x, 0) \in E(\mathbb{F}_p)$, the double of P is $Q = 2P = \mathcal{O}$.
5. For all $P = (x, y) \in E(\mathbb{F}_p)$ with $y \neq 0$, the double of P is $Q = 2P = (x_3, y_3)$ with

$$\begin{aligned} x_3 &= \lambda^2 - 2x, \\ y_3 &= \lambda(x - x_3) - y, \end{aligned}$$

   where $\lambda = \frac{3x^2 + a}{2y}$.

With the addition law, $(E(\mathbb{F}_p), +)$ is structured with a scalar multiplication so that, for $P = (x, y) \in E(\mathbb{F}_p)$, and $n \in N$, the point nP is defined by

$$nP = \underbrace{P + \cdots + P}_{n \text{ times}}.$$

The order of $E(\mathbb{F}_p)$ can be estimated by the theorem of Hasse:

$$(\sqrt{p}-1)^2 \leq \#E(\mathbb{F}_p) \leq (\sqrt{p}-1)^2.$$

If $G \in E(\mathbb{F}_p)$ with $G \neq \mathcal{O}$, then G generates a cyclic subgroup of $E(\mathbb{F}_p)$, denoted $\langle G \rangle$, by

$$\langle G \rangle = \{G, 2G, \ldots, nG\},$$

where the integer n is the smallest divisor of $\#E(\mathbb{F}_p)$ satisfying $nG = \mathcal{O}$. Since n divides $\#E(\mathbb{F}_p)$, then $h = \frac{\#E(\mathbb{F}_p)}{n}$ is also an integer. It is called the cofactor of G.

3.2. Special Cryptographic Curves

Special curves are used to build some cryptographic systems to improve the efficiency of operations for limited-resource devices. We list below some of them.

- Edwards curves [33]. These curves were introduced by Edwards in 2007. Shortly after, Bernstein and Lange [34] transformed them with an equation of the form $x^2 + y^2 = 1 + dx^2y^2$ over a finite prime field $\mathbb{F}_p$ with $p > 2$ and $d \in \mathbb{F}_p \backslash \{0, 1\}$. Such curves have a single arithmetic addition and are suitable for use against side-channel attacks.
- Montgomery curves [35]. In 1987, Montgomery introduced a new form for elliptic curves. Montgomery's curves are defined over a finite field $\mathbb{F}_p$ by the equation $By^2 = x^3 + Ax^2 + x$ where $A, B \in \mathbb{F}_p$. Montgomery's curves are used to accelerate the scalar multiplication via Montgomery's ladder.
- Koblitz curves [36]. These curves are defined over a binary finite field $\mathbb{F}_{2^n}$ with the equation $y^2 + xy = x^3 + ax^2 + 1$ with $a \in \{0, 1\}$. They are used to accelerate the addition and the scalar multiplication.
- Binary elliptic curves. These are curves of the form $y^2 + xy = x^3 + ax^2 + b$ where $a, b \in \mathbb{F}_{2^n}$ and $b \neq 0$. Binary elliptic curves are not widely used, mainly because the ECDLP in such curves seems less hard than the ECDLP in elliptic curves over finite prime fields $\mathbb{F}_p$ with $p > 2$ (see [37] for more discussions).

4. Elliptic Curves Cryptography

In this section, we describe the main schemes in the ECC. Their security is based on the hardness of the ECDLP. Let E be an elliptic curve over $\mathbb{F}_p$, and $G \in E(\mathbb{F}_p)$ be a base point of order n. Given a point $P \in \langle G \rangle$, find the integer k such that $0 \leq k \leq n-1$, and $P = kG$.

4.1. The Diffie–Hellman Elliptic Curve Key Agreement Algorithm (ECDH)

The Diffie–Hellman elliptic curve key exchange is designed to secretly and securely communicate a key that can be used for various applications such as symmetric cryptosystems. Assume that two entities, A and B, want to agree on a common key. The Elliptic Curve Diffie–Hellman key agreement algorithm (ECDH) can be used in the following steps.

1. The entities A and B agree on a finite field $\mathbb{F}_p$, an elliptic curve E over $\mathbb{F}_p$, and a base point $G \in E(\mathbb{F}_p)$ of large order.
2. The entity A selects a private random integer a, computes $P_a = aG$, and sends P_a to the entity B.
3. The entity B selects a private random integer b, computes $P_b = bG$, and sends P_b to the entity A.
4. The entity A computes $Q = aP_b$.
5. The entity B computes $Q = bP_a$.

The shared key is $Q = aP_b = bP_a = abG$.

4.2. The ElGamal Elliptic Curve Cryptosystem (ECEG)

One of the most popular public key schemes is the ElGamal cryptosystem [6]. It is based on the Diffie–Hellman key exchange. A version for elliptic curves can be described as follows, where entity A wants to safely send a message to entity B.

1. The entities A and B agree on a finite field $\mathbb{F}_p$, an elliptic curve E over $\mathbb{F}_p$, and a base point $G \in E(\mathbb{F}_p)$ of large order.
2. The entity B selects a private random integer b, computes $P_b = bG$, and sends P_b to the entity A.
3. The entity A transforms the message to a point $M \in E(\mathbb{F}_p)$.
4. The entity A selects a private random integer a, computes $P_a = aG$, $C = M + aP_b$, and sends P_a and C to the entity B.
5. The entity B computes $M = C - bP_a$.

The decryption is correct since

$$C - bP_a = M + aP_b - baG = M + abG - baG = M.$$

4.3. The Elliptic Curve Digital Signature Algorithm (ECDSA)

The ECDSA is a digital signature scheme based on elliptic curves, proposed in 2001 by Johnson, Menezes, and Vanstone [7]. It was standardized in the ANSI X9.62 [38], IEEE 1363-2000 [39], and ISO/IEC 15946-2 standards. It enables to sign a message so that the recipient can check that the message is transmitted by the correct entity. To work with ECDSA, two entities A and B first agree on a finite field $\mathbb{F}_p$ where p is a prime number, on an elliptic curve $E : y^2 = x^3 + ax + b$ over $\mathbb{F}_p$, on a point P with a large order n. Then, A selects a private key $d_A \in [1, n-1]$, and B selects a private key $d_B \in [1, n-1]$. Moreover, A computes its public key $Q_A = d_A P$.

Assume that the entity A wants to send a message m to the entity B using ECDSA. The signature generation algorithm is performed by A (see [40]) as presented in Algorithm 1.

Algorithm 1 Signature generation algorithm

```
Require: A hash function $H$, an elliptic curve $E$, a message $m$, a base point $P \in E$, the order
    $n$ of $P$, and the private key $d_A$ of A.
Ensure: The signature $(r, s)$.
 1: Compute $z = H(m)$.
 2: Choose a random integer $k$ with $1 \leq k \leq n-1$ and $\gcd(k, n) = 1$.
 3: Compute $(x_1, y_1) = kP$ in $E$.
 4: Compute $r \equiv x_1 \pmod{n}$.
 5: if $r = 0$ then
 6:     Restart from Step 2.
 7: end if
 8: Compute $k_2 \equiv k^{-1} \pmod{n}$.
 9: Compute $s \equiv k_2(z + rd_A) \pmod{n}$.
10: if $s = 0$ then
11:     Restart from Step 2.
12: end if
13: Return the signature $(r, s)$.
```

Next, the entity B can verify the signature of entity A using the verification algorithm as presented in Algorithm 2.

Algorithm 2 Signature verification algorithm

```
Require: The hash function $H$, the elliptic curve $E$, the base point $P \in E$, the order $n$ of $P$,
    the public key $Q_A$ of A, and the signature $(r, s)$.
Ensure: Acceptance or rejection of the signature.
 1: if $r \notin [1, n-1]$, or $s \notin [1, n-1]$ then
 2:     Return Rejection
 3: end if
 4: Compute $z = H(m)$.
 5: Compute $w \equiv s^{-1} \pmod{n}$.
 6: Compute $u_1 \equiv zw \pmod{n}$.
 7: Compute $u_2 \equiv rw \pmod{n}$.
 8: Compute $(x_1, y_1) = u_1 P + u_2 Q_A$ in $E$.
 9: if $r \equiv x_1 \pmod{n}$ then
10:     Return Acceptance.
11: else
12:     Return Rejection.
13: end if
```

5. Security of ECC

In this section, we present the most powerful attacks on ECC systems. Most of the attacks are designed to solve the elliptic curve discrete polynomial.

5.1. *Pollard's Rho Algorithm*

Let n be the order of the subgroup $\langle P \rangle$, and $Q \in \langle P \rangle$ with $Q = kP$. Pollard's rho method tries to find a collision, that is, two couples of integers (a, b), (a', b') such that $(a, b) \neq (a', b')$ and $aP + bQ = a'P + b'q$. Equivalently, this is $(a - a')P = (b' - b)Q$, from which one deduces $a - a' \equiv k(b' - b) \pmod{n}$. If $\gcd(b' - b, n) = 1$, then

$$k \equiv (a - a')(b' - b)^{-1} \pmod{n}.$$

If the couples (a, b) and (a', b') are selected randomly in $[1, n-1]$, the expected running time is $\mathcal{O}\left(\sqrt{\pi n/2}\right)$, and the storage of the triples $(a, b, aP + bq)$ requires $\mathcal{O}\left(\sqrt{\pi n/2}\right)$ cells, which is infeasible if n is large. Nevertheless, some variants of Pollard's rho method solve the ECDLP with the same running time, but with much less storage. The following variant is one of them. It proceeds as in Algorithm 3, where the following functions are used

$$f(R_i) = \begin{cases} P + R_i & \text{if } R_i \in S_1, \\ 2R_i & \text{if } R_i \in S_2, \\ Q + R_i & \text{if } R_i \in S_3, \end{cases}$$

$$g(a_i) = \begin{cases} 1 + a_i \pmod{n} & \text{if } R_i \in S_1, \\ 2a_i \pmod{n} & \text{if } R_i \in S_2, \\ a_i \pmod{n} & \text{if } R_i \in S_3, \end{cases}$$

$$h(b_i) = \begin{cases} b_i \pmod{n} & \text{if } R_i \in S_1, \\ 2b_i \pmod{n} & \text{if } R_i \in S_2, \\ 1 + b_i \pmod{n} & \text{if } R_i \in S_3. \end{cases}$$

Algorithm 3 Pollard's rho algorithm for the ECDLP

```
Require: An elliptic curve $E$, a base point $P \in E$, the order $n$ of $P$, a point $Q \in \langle P \rangle$.
Ensure: The integer $k$ such that $Q = kP$.
 1: Partition $\langle P \rangle$ in three sets of almost equal size, namely $\langle P \rangle = S_1 \cup S_2 \cup S_3$.
 2: Choose two random integers $a_0, b_0 \in [1, n-1]$.
 3: Compute $R_0 = a_0 P + b_0 Q$.
 4: Compute $R_1 = f(R_0)$, $a_1 = g(a_0)$, $b_1 = h(b_0)$.
 5: Compute $R_2 = f(R_1)$, $a_2 = g(a_1)$, $b_2 = h(b_1)$.
 6: Set $i = 0$
 7: while $R_i \neq R_{2i}$ do
 8:     Compute $R_{i+1} = f(R_i)$, $a_{i+1} = g(a_i)$, $b_{i+1} = h(b_i)$.
 9:     Compute $R_{2(i+1)} = f(f(R_{2i}))$, $a_{2(i+1)} = g(g(a_{2i}))$, $b_{2(i+1)} = h(h(b_{2i}))$.
10:     $i = i + 1$.
11: end while
12: if $\gcd(b_i - b_{2i}, n) = 1$ then
13:     Compute $k \equiv (a_{2i} - a_i)(b_i - b_{2i})^{-1} \pmod{n}$.
14: else
15:     Go to step 2.
16: end if
17: Return $k$.
```

Several variants have been proposed to improve Pollard's rho method [41–43]. Moreover, there exists a parallelized variant of Pollard's rho method (see [40], Section 4.1.2), which can be applied to M processors, with running time $\mathcal{O}\left(\frac{1}{M}\sqrt{\frac{\pi n}{2}}\right)$.

5.2. The Pohlig–Hellman Algorithm

The Pohlig–Hellman attack on the discrete logarithm problem was first presented in [44]. It applies optimally when $\#E(\mathbb{F}_p)$ is divisible only by small prime factors. It reduces the problem of computing the ECDLP over subgroups of prime order.

Let n be the order of the group $\langle P \rangle$. Suppose that $n = p_1^{n_1} p_2^{n_2} \cdots p_r^{n_r}$. Let $Q \in \langle P \rangle$ with $Q = kP$. The goal of the Pohlig–Hellman method is to find $k \in [0, n-1]$ using the Chinese Remainder Theorem by solving the system

$$\begin{aligned} k &\equiv k_1 \pmod{p_1^{n_1}}, \\ k &\equiv k_2 \pmod{p_2^{n_2}}, \\ &\vdots \\ k &\equiv k_r \pmod{p_r^{n_r}}, \end{aligned}$$

for which the unique solution in $[0, n-1]$ is

$$k \equiv \sum_{i=1}^{r} k_i N_i x_i \pmod{n}, \quad \text{with } N_i = \frac{n}{p_i^{n_i}}, \quad x_i = \frac{1}{N_i} \pmod{p_i^{n_i}}.$$

The values k_i, $i = 1, \ldots, r$, are computed recursively. Set

$$k_i = z_0^{(i)} + z_1^{(i)} p_i + z_2^{(i)} p_i^2 + \cdots + z_{n_i-1}^{(i)} p_i^{n_i-1},$$

with $z_j^{(i)} \in [0, p_i - 1]$. Also, set

$$P_0^{(i)} = \frac{n}{p_i} P, \quad Q_0^{(i)} = \frac{n}{p_i} Q.$$

Then, since $p_i P_0^{(i)} = \mathcal{O}$, and $k = k_i + m_i p_i^{n_i}$ for some integer m_i, $P_0^{(i)}$ satisfies

$$kP_0^{(i)} = k_i P_0^{(i)} + m_i p_i^{n_i} P_0^{(i)} = k_i P_0^{(i)} = z_0^{(i)} P_0^{(i)}.$$

Then

$$Q_0^{(i)} = \frac{n}{p_i} Q = k\frac{n}{p_i} P = kP_0^{(i)} = z_0^{(i)} P_0^{(i)}.$$

Hence, $z_0^{(i)}$ can be computed by solving the discrete logarithm $Q_0^{(i)} = z_0^{(i)} P_0^{(i)}$ in $\left\langle P_0^{(i)} \right\rangle$.

Using $z_0^{(i)}$, we set

$$Q_1^{(i)} = \frac{n}{p_i^2}\left(Q - z_0^{(i)} P\right),$$

which satisfies

$$Q_1^{(i)} = z_1^{(i)} P_0^{(i)}.$$

Again, $z_1^{(i)}$ can be computed by solving the discrete logarithm $Q_1^{(i)} = z_1^{(i)} P_0^{(i)}$ in $\left\langle P_0^{(i)} \right\rangle$.

This procedure is repeated recursively and leads to the computation of $z_s^{(i)}$ by solving the discrete logarithm $Q_s^{(i)} = z_s^{(i)} P_0^{(i)}$ in $\left\langle P_0^{(i)} \right\rangle$ where

$$Q_s^{(i)} = \frac{n}{p_i^{s+1}}\left(Q - \left(z_0^{(i)} + z_1^{(i)} p + \cdots + z_{s-1}^{(i)} p^{s-1}\right) P\right).$$

The Pohlig–Hellman method can be summarized in Algorithm 4.

Algorithm 4 Pohlig–Hellman algorithm for the ECDLP

```
Require: An elliptic curve $E$, a base point $P \in E$, the order $n$ of $P$, a point $Q \in \langle P \rangle$.
Ensure: The integer $k$ such that $Q = kP$.
 1: Factor $n$ as $n = p_1^{n_1} p_2^{n_2} \cdots p_r^{n_r}$.
 2: Set $k = 0$.
 3: for $i$ from 1 to $r$ do
 4:     Set $k_i = 0$.
 5:     Compute $P_0 = \frac{n}{p_i} P$.
 6:     Compute $R = \frac{n}{p_i} Q$.
 7:     for $j$ from 0 to $n_i - 1$ do
 8:         Compute $z$ such that $R = zP_0$.
 9:         Compute $k_i = k_i + zp^j$.
10:         Compute $R = \frac{n}{p_i^{j+1}}(Q - k_i P)$.
11:     end for
12:     Compute $N_i = \frac{n}{p_i^{n_i}}$.
13:     Compute $x_i \equiv N_i^{-1} \pmod{p_i^{n_i}}$.
14:     Compute $k \equiv k + k_i N_i x_i \pmod{n}$.
15: end for
16: Return $k$.
```

The complexity of the Pohlig–Hellman method is expressed in the form $\mathcal{O}\left(\sum_{i=1}^{r} n_i \left(\log(n) + \sqrt{p_i}\right)\right)$, but for most values of n, the complexity is of $\mathcal{O}\left(\sqrt{q}\right)$, where q is the largest prime factor of n. As a consequence, to maximize the resistance of solving the ECDLP by the Pohlig–Hellman method, the order $\#E(\mathbb{F}_p)$ should be a multiple of at most one large prime number.

5.3. The Side-Channel Attacks

To test the security of a cryptosystem, several kinds of security are applied such as provable security and side-channel security. While provable security seems more theoretical, side-channel security is devoted to practical implementations of cryptographic systems. Attacks that scrutinize the implementation procedures are called side-channel attacks. A naive and direct implementation of some public key systems such as RSA, DH, and ECC can leak information about their private keys, which permits to recovery of the entire key. A typical example is the modular exponentiation in RSA and DH, as well as the double and add procedure for scalar multiplication of points on elliptic curves.

In 1996, Kocher [45] presented the power analysis, the first possible side-channel attack. Since then, various types of side-channel attacks have been proposed for practical use. Some are based on implementation issues such as single power analysis [45], differential power analysis [46], fault attacks [47], and timing attacks [45].

If the addition of two points P and Q is naively implemented, then it is possible to guess if it is computed for $P \neq Q$ or $P = Q$. Similarly, if the scalar multiplication kP is simply implemented using the double and add method, then one can guess all the bits of the binary decomposition of k. This is feasible by measuring the time taken to perform the computation for any bit. When the bit is 1, one has to compute an addition on the elliptic curve as in Steps 5–7 of Algorithm 5, while no addition is needed when the bit is 0. As a consequence, performing a computation for a bit 1 is longer than performing a computation for a bit 0.

Algorithm 5 Left to right double and add method

```
Require: An elliptic curve $E$, a point $P \in E(\mathbb{F}_p)$, an integer $k$.
Ensure: The point $Q = kP \in E(\mathbb{F}_p)$.
1: Decompose $k = a_{s-1}2^{s-1} + \cdots + a_1 2 + a_0$, $a_i \in \{0,1\}$, $a_{s-1} = 1$.
2: Set $Q = \mathcal{O}$.
3: for $i$ from $s-1$ down to 0 do
4:     Compute $Q = 2Q$.
5:     if $a_i = 1$ then
6:         Compute $Q = Q + P$.
7:     end if
8: end for
9: Return $Q$.
```

Several algorithms for scalar multiplication have been proposed against timing attacks [48]. They make the scalar multiplication regular and constant-time. A typical example is the double and add always method, as presented in Algorithm 6.

A yet more regular and more resistant way to perform the scalar multiplication on elliptic curves is the Montgomery ladder [35]. This algorithm was originally specified for Montgomery's elliptic curves and was later generalized to any elliptic curve with Weierstrass form, independently by Brier and Joye in [49], and Izu and Takagi in [50].

Another known side channel attack is fault attack [47,51]. It consists in injecting a fault during the arithmetic operations and exploiting the output to guess a part of or even the whole private key. The basic idea is to inject a fault in the regular computation on the original curve E to force it to be performed in a parallel computation on a weaker curve E' where the ECDLP is easy to solve. To avoid fault attacks, several countermeasures have been proposed. The basic countermeasure is to check whether the output is still a point of E. Another countermeasure is to use a less sensitive scalar multiplication method, such as Montgomery's ladder method, as presented in Algorithm 7.

Algorithm 6 Double and add always method

```
Require: An elliptic curve $E$, a point $P \in E$, an integer $k$.
Ensure: The point $Q = kP \in E$.
 1: Decompose $k = a_{s-1}2^{s-1} + \cdots + a_1 2 + a_0$, $a_i \in \{0,1\}$, $a_{s-1} = 1$.
 2: Set $Q = \mathcal{O}$.
 3: for $i$ from $s-1$ down to 0 do
 4:     Compute $Q = 2Q$.
 5:     Compute $R = Q + P$.
 6:     if $a_i = 1$ then
 7:         Set $Q = R$.
 8:     else
 9:         Set $Q = Q$.
10:     end if
11: end for
12: Return $Q$.
```

Algorithm 7 Montgomery's ladder

```
Require: An elliptic curve $E$, a point $P \in E$, an integer $k$.
Ensure: The point $Q = kP \in E$.
 1: Decompose $k = a_{s-1}2^{s-1} + \cdots + a_1 2 + a_0$, $a_i \in \{0,1\}$, $a_{s-1} = 1$.
 2: Set $Q_0 = P$.
 3: Set $Q_1 = 2P$.
 4: for $i$ from $s-2$ down to 0 do
 5:     Compute $Q_{1-a_i} = Q_0 + Q_1$.
 6:     Compute $Q_{a_i} = 2Q_{a_i}$.
 7: end for
 8: Return $Q_0$.
```

5.4. Shor's Algorithm

In 1994, Shor [1,52] presented a quantum algorithm to factor large composite numbers, and to solve the discrete logarithm problem in a finite field of prime order. Shor's algorithm was extended to solve the elliptic curve discrete logarithm problem by Proos and Zalka [53] in 2003. It may be exploited by a large-scale quantum computer and would undermine the security of the most popular public key systems such as RSA, DH, ElGamal, and ECC. If E is an elliptic curve over $\mathbb{F}_p$, then Shor's algorithm can be efficiently used to solve the elliptic curve discrete logarithm in a polynomial running time of $\Omega(\log(\#E(\mathbb{F}_p)))$ (see [54], Theorem 1.2). A detailed description of Shor's algorithm for the ECDLP is proposed in [55].

5.5. Other Attacks

Several attacks have been presented to compute the ECDLP, some are less efficient than Pollard's rho method, and some are more efficient for specific types of elliptic curves.

- The baby-step–giant-step algorithm was invented by Shanks [56] in 1971. While its running time is approximately the same as Pollard's rho method, it requires approximately $\sqrt{n}$ space for values storage. The idea behind this method is to choose an integer $m > \sqrt{n}$, to compute $P' = mP$, to compute, and to store all values of aP (the baby steps) and $Q - aP'$ (the giant steps) for $a = 1, \ldots, m$ and to compare the stored lists. If one match is found, then $aP = Q - bmP$ for some integers a and b. This gives $Q = (a + mb)P$, and $k \equiv a + mb \pmod{n}$.
- The MOV attack, due to Menezes, Okamoto, and Vanstone [57], is efficient when the elliptic curve is supersingular, that is $\#E(\mathbb{F}_p) = p + 1$. It is based on Weil pairing that maps two points in $E(\mathbb{F}_p)$ to an element in $\mathbb{F}_{p^k}$. The integer k is the embedding degree associated with any elliptic curve $E(\mathbb{F}_p)$. It is the smallest integer $k \geq 2$ such that $\#E(\mathbb{F}_p)$ divides $p^k - 1$. If $P_1, P_2, Q = rP_1$ are three given points in $E(\mathbb{F}_p)$ with

an unknown r, and e is the Weil pairing, then one can compute $a = (P_1, P_2) \in \mathbb{F}_{p^k}$, and $b = e(Q, P_2) \in \mathbb{F}_{p^k}$. Hence,

$$b = e(Q, P_2) = e(rP_1, P_2) = e(P_1, P_2)^r = a^r,$$

that is $b = a^r$. This reduces to the discrete logarithm problem in $\mathbb{F}_{p^k}$. For supersingular curves, $k \leq 6$ is sufficiently small, and the discrete logarithm problem can be easily solved over $\mathbb{F}_{p^k}$. If the elliptic curve is not supersingular, it is required that $k \geq 100$.

- The elliptic curves such that $\#E(\mathbb{F}_p)$ are called anomalous and are weak for the attacks presented in [58–60]. In such curves, the ECDLP can be reduced to the discrete logarithm problem in the additive field $(\mathbb{F}_p, +)$ which is easy to solve.

5.6. Robust Elliptic Curves for Cryptography

To avoid the attacks described before, it is crucial to choose robust elliptic curves for use in cryptography. We list here a few criteria for this purpose.

- The size of $\#E(\mathbb{F}_p)$, as well as the size of $\#\langle P \rangle$ should be large enough to resist the attacks that have a running time or storage that depend on $n = \#\langle P \rangle$ such as Pollard's rho method, Pohlig–Hellman's method, and baby-step–giant-step method.
- Both $\#E(\mathbb{F}_p)$ and $\#\langle P \rangle$ should have a dominant large prime factor. This property ensures that Pollard's rho attack and Pohlig–Hellman's attack will be ineffective.
- The curve E should not be anomalous, that is, the order $\#E(\mathbb{F}_p)$ should not be equal to p. When the curve is anomalous, the ECDLP in E can be reduced to the additive discrete logarithm problem in $\mathbb{F}_p$, which is trivial to solve [58–60].
- The curve E should not be supersingular, that is the order $\#E(\mathbb{F}_p)$ should not be equal to $p + 1$. This requirement follows the work of Menezes, Okamoto, and Vanstone [57], and the work of Frey and Rück [61]. Both works show that, for an elliptic curve E over $\mathbb{F}_p$, the ECDLP can be transferred from $E(\mathbb{F}_p)$ to the Discrete Logarithm Problem (DLP) in the multiplicative group $\mathbb{F}_{p^k}^{\times}$ for some positive integer k. If k is small, typically $k < \log^2(p)$, then the DLP in $\mathbb{F}_p^k$ can be attacked by a standard method, such as the baby-step–giant-step [56], Pollard's method [62], Pohlig–Hellman's method [44], or the index calculus method [63]. To avoid a MOV attack, it is required to check that $\#E(\mathbb{F}_p)$ does not divide the integers $p^r - 1$ for $1 \leq r \leq 100$.

We notice that several tools are devoted to selecting safe elliptic curves. A typical example is [64] where the security of almost all popular cryptographic elliptic curves is discussed.

6. ECC and Machine Learning

In this section, we discuss the use of machine learning to enhance the security and efficiency of ECC.

6.1. Speeding Up the Generation Phase

AI has significant potential for optimizing parameters in ECC, particularly through techniques like GAN [65,66], GA, PSO, and compression techniques [67]. These AI-driven methods enhance ECC's efficiency by reducing computational overhead in the generation phase, which is crucial for applications requiring both high security and real-time performance.

6.1.1. GANs and AI-Driven ECC Optimization

GANs are a machine learning framework with two neural networks, a generator and a discriminator, trained simultaneously. The generator produces synthetic data resembling a given dataset, while the discriminator assesses these samples against real data.

In cryptography, GANs offer the advantage of generating secure, random encryption keys, which enhances system resilience against attacks. Unlike traditional encryption, GANs use floating-point numbers, enabling more complex encryption patterns beyond binary sequences [68].

The authors in [27] analyze and compare the effectiveness of GA and PSO in optimizing ECC parameters within a simulated e-commerce environment, emphasizing their potential to improve cybersecurity. Meanwhile, the authors in [69] introduce an image encryption method that combines ECC with GA to bolster data security and confidentiality.

GA utilizes principles of biological evolution to generate and refine a population of candidate solutions, known as chromosomes, through processes like selection, crossover, and mutation. By evaluating each candidate using a fitness function, GA effectively navigates complex search spaces to converge on optimal ECC parameters, enhancing security and efficiency. Similarly, PSO mimics social behaviors observed in nature, offering simplicity in implementation and a tendency to avoid local optima. Together, these AI-driven methods present innovative solutions to the challenges faced in ECC optimization. The integration of GA into the process of generating keys for ECC enhances both the security and efficiency of key pairs [70]. This approach begins with the initialization of a population of candidate keys, represented as chromosomes, where each chromosome corresponds to a point on the elliptic curve defined by specific parameters. The x and y coordinates of these points are generated randomly within the curve's constraints, as presented in Algorithm 8, allowing for the creation of multiple potential keys [19,71].

Algorithm 8 GAN-Based ECC Key Generation Algorithm

```
Require: Elliptic curve parameters $E(a,b,p)$, a base point $G \in E$, the order $n$ of $G$, GAN components: generator $\mathcal{G}$ and discriminator $\mathcal{D}$.
Ensure: A valid ECC key pair $(d,Q)$ where $Q = d \cdot G$.
 1: Initialize GAN parameters:
        - Define the architectures for $\mathcal{G}$ and $\mathcal{D}$.
        - Set random initial weights for $\mathcal{G}$ and $\mathcal{D}$.
        - Define the loss functions for adversarial training.
 2: Prepare a dataset of valid ECC keys:
        - Generate random private keys $d \in [1, n-1]$.
        - Compute corresponding public keys $Q = d \cdot G$.
 3: Train the GAN:
 4: while GAN training not converged do
 5:     Train the discriminator $\mathcal{D}$:
        - Input: Real key pairs $(d,Q)$ and generated key pairs $(\hat{d},\hat{Q})$.
        - Update $\mathcal{D}$ to classify real vs. fake key pairs.
 6:     Train the generator $\mathcal{G}$:
        - Generate synthetic private keys $\hat{d}$ from random noise $z$.
        - Update $\mathcal{G}$ to minimize $\mathcal{D}$'s ability to distinguish real from fake keys.
 7: end while
 8: Generate ECC keys:
 9: Generate a private key $\hat{d} = \mathcal{G}(z)$ from random noise $z$.
10: Compute the corresponding public key $\hat{Q} = \hat{d} \cdot G$.
11: Validate the key pair:
        - Ensure $\hat{Q} \in E(a,b,p)$.
        - If validation fails, restart from Step 1.
12: Output: Return all valid key pairs $(\hat{d},\hat{Q})$.
```

The algorithm describes a method for generating ECC key pairs using GANs. GANs consist of a generator, which creates synthetic private keys from random noise, and a discriminator, which distinguishes real key pairs from generated ones. The GANs are trained on a dataset of valid ECC key pairs, where each private key is a randomly chosen integer within the valid range, and the corresponding public key is computed using elliptic curve point multiplication. During training, the generator aims to produce private keys that closely resemble real ones, while the discriminator learns to classify key pairs as real or synthetic. Once the GAN training converges, the generator is used to produce private keys, and the associated public keys are computed using the ECC base point and curve parameters. A validation step ensures that the generated public keys lie on the elliptic curve, confirming the correctness of the key pairs. The algorithm outputs all valid key pairs, ready for use in cryptographic applications.

6.1.2. Applying PSO to ECC Key Generation

PSO is a heuristic optimization algorithm developed by Kennedy and Eberhart in 1995, inspired by the natural behaviors observed in bird flocks searching for food. This approach can be effectively applied to enhance the process of generating secure key pairs in ECC.

Within the PSO framework, individual "particles" symbolize potential candidates for elliptic curve parameters, such as curve coefficients or key pair values. Each particle represents a point in the solution space and is initialized with random values for the parameters. These particles are also assigned velocities that guide their movements within the solution space.
The PSO algorithm follows these key steps to optimize ECC key generation:

1. Initialization: A swarm of particles is initialized with random ECC parameter configurations, each associated with a random velocity.
2. Fitness Evaluation: The fitness of each particle is computed based on specific criteria. In the context of ECC, the fitness function evaluates the cryptographic strength, randomness, and operational efficiency of the candidate parameters.
3. Updating Positions and Velocities: Particles update their velocities and positions iteratively. The acceleration of each particle is influenced by two factors: its own personal best position (where it achieved the highest fitness so far) and the global best position (the best fitness among all particles in the swarm). These updates enable particles to balance exploration and exploitation within the search space.
4. Refinement and Convergence: Over successive iterations, particles move closer to the optimal solution, refining their positions based on both individual and collective experience. The algorithm terminates when convergence is achieved or a predefined number of iterations is completed.

By applying PSO to ECC, the algorithm identifies the global best position in the swarm, representing the optimized ECC parameters. These parameters can then be used to generate secure and robust key pairs.

Using PSO for key generation in ECC offers significant advantages over conventional methods. The cooperative dynamics of particles enable the algorithm to efficiently navigate the solution space, enhancing the randomness and robustness of the generated keys. Unlike GA, PSO emphasizes collaboration rather than competition, leading to a more adaptive and precise optimization process.

This approach ensures that the resulting ECC key pairs are not only highly secure but also optimized for performance, making PSO a valuable tool in modern cryptographic systems.

6.1.3. Applying Compression to ECC Key Generation

AI-driven compression techniques offer a promising approach to enhancing the efficiency and security of ECC key generation. This method leverages artificial intelligence to analyze the input stream, identify repetitive patterns, and replace them with more efficient, unused character sets [69]. Given ECC's inherent advantage of requiring smaller key sizes for equivalent security compared with traditional cryptographic methods, AI-based compression further refines this process in several key ways:

1. Key Size Reduction: ECC already benefits from compact key sizes, and AI-based compression can further reduce the amount of data involved by eliminating redundancies in the input stream. This results in more efficient key representation, allowing for faster cryptographic operations while maintaining robust security.
2. Enhanced Computational Efficiency: By optimizing the input data and removing unnecessary repetition, AI-driven compression reduces the computational workload required during key generation. This is particularly beneficial for resource-constrained environments, where reducing the number of operations can significantly enhance system performance.
3. Improved Security Through Increased Randomness: The process of transforming repetitive input patterns into less predictable forms introduces additional randomness into the key generation process. This increases the cryptographic strength of the generated keys, making them more resilient to attacks, such as brute force and other forms of cryptanalysis.
4. Optimized Resource Utilization: In systems with limited computational and memory resources, such as mobile devices and IoT environments, the ability to minimize data processing during key generation is crucial. AI-based compression ensures that the key generation process uses fewer resources, enabling faster, secure key production even under constraints.

6.2. Enhancing the Security

ECC is integral to modern cryptographic systems, and with advancements in AI, novel approaches have been applied for both enhancing and analyzing the security of ECC. AI-based methods offer new possibilities for cryptanalysis, helping to identify vulnerabilities and improve cryptographic processes. This section examines studies that apply AI techniques in the cryptanalysis of ECC, highlighting key insights and gaps (see Table 1).

Table 1. Summary of research studies on ECC and their limitations.

Ref.	Limitations
[72]	Focuses on ECC cryptanalysis but does not extend its research to parameter optimization or explore AI techniques beyond basic cryptanalysis.
[73]	Addresses the optimization of power consumption for mobile devices using PSO and Simplified Swarm Optimization but fails to provide a comprehensive comparison with GA for ECC optimization.
[67]	Explores PSO for ECC key generation but does not offer a thorough comparison with other AI techniques like GA. The research is centered on key generation, without considering the broader optimization of ECC parameters in other contexts, such as large-scale cryptographic systems.
[74]	Investigates the use of DNA-based cryptography and Hyperelliptic Curve Cryptography (HECC) for securing multicloud environments but does not explore other AI techniques such as GA or PSO for ECC. The study also lacks practical implementation details for use in real-world applications.

GANs can pose significant threats to ECC through various attack vectors. One method involves key generation attacks, where GANs can be trained on known key pairs to learn their distribution, enabling them to produce new keys that closely resemble valid ones, potentially allowing an attacker to intercept or decrypt messages. Additionally, GANs can generate adversarial examples that mimic legitimate keys during key exchange protocols, thereby misleading the system and facilitating unauthorized access. Through adversarial training, GANs can simulate adversary behavior crafting plaintexts or ciphertexts that exploit vulnerabilities in ECC implementations, such as chosen-plaintext and chosen-ciphertext attacks. Moreover, they can perform model inversion attacks by analyzing system outputs and reconstructing private keys or sensitive information from public data shared during cryptographic operations. Lastly, GANs may exploit implementation flaws by training on side-channel information, leading to targeted attacks that compromise ECC security. These emerging threats necessitate a thorough understanding of the interactions between GANs and ECC to enhance cryptographic resilience against such sophisticated adversarial techniques [75,76].

6.3. Use of Machine Learning for ECC

To boost the effectiveness of the algorithms of the cryptographic systems based on the elliptic curve cryptography, especially for the Internet of Things (IoT), and devices with limited resources, machine learning is a practical tool to improve their efficiency and security. Below, we summarize some of the tasks that machine learning can perform to enhance the cryptographic systems in the field of ECC:

- Generate strong private keys and seeds for use in ECC systems.
- Select the most efficient and secure elliptic curves in various forms with large keys.
- Implement the most efficient elliptic curve algorithms [40] and operations to perform the computation in an optimal time.
- Implement the most prominent, secure, and efficient key exchange protocols such as ECDH.
- Implement the most prominent, secure, and efficient digital signature algorithms such as ECDSA [7], or EdDSA [77], especially Ed25519. This will guarantee the integrity and the authenticity of the shared keys. Moreover, it ensures the parties sign their public keys, and allows a third party to verify the authenticity of the keys.
- Implement the most prominent, secure, and efficient public key cryptosystems based on elliptic curves such as the Elliptic Curve Integrated Encryption Scheme (ECIES) [78]. This enables to encrypt of small data messages such as PINs, and phone or credit card numbers. This also enables to transport of larger session keys for use in symmetric cryptography.
- Implement and test the most powerful attacks on ECC systems in order to test their security.

7. Conclusions

We presented the theory of ECC, including its arithmetic, applications, security, and the main attacks that can be launched to compromise systems based on ECC. We also introduced the basic concepts of machine learning and explored how it can be used to enhance the security and efficiency of the algorithms employed in ECC. The study demonstrated that ECC can significantly benefit from machine learning technology, particularly in generating optimal parameters that are resistant to common attacks against ECC.

Author Contributions: Conceptualization, S.L. and A.N.; methodology, S.L. and A.N.; software, J.J., A.C., S.L., and A.N.; validation, S.L. and A.N.; formal analysis, J.J., A.C., S.L., and A.N.; investigation, J.J., A.C., S.L., and A.N.; writing—original draft preparation, J.J., A.C., S.L., and A.N.; writing—review and editing, S.L. and A.N.; visualization, J.J., A.C., S.L., and A.N.; supervision, S.L. and A.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
CNN	Convolutional Neural Network
DLP	Discrete Logarithm Problem
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie–Hellman
ECDLP	Elliptic Curve Discrete Logarithm Problem
ECDSA	Elliptic Curve Digital Signature Algorithm
ECEG	ElGamal Elliptic Curve Cryptosystem
ECIES	Elliptic Curve Integrated Encryption Scheme
EdDSA	Edwards Curve Digital Signature Algorithm
FNN	Feedforward Neural Networks
GAN	Generative Adversarial Network
GA	Genetic Algorithm
HECC	Hyperelliptic Curve Cryptography
LWE	Learning With Error
ML	Machine Learning
PSO	Particle Swarm Optimization
RNN	Recurrent Neural Networks
RSA	Rivest, Shamir, Adelman

References

1. Shor, P.W. Algorithms for quantum computation: Discrete logarithms and factoring. In Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, USA, 20–22 November 1994; pp. 124–134.
2. Rivest, R.; Shamir, A.; Adleman, L. A Method for Obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **1978**, *21*, 120–126. [CrossRef]
3. Koblitz, N. Elliptic curve cryptosystems. *Math. Comput.* **1987**, *48*, 203–209. [CrossRef]
4. Miller, V.S. Use of elliptic curves in cryptography. In *Advances in Cryptology—CRYPTO '85 Proceedings. CRYPTO 1985*; Lecture Notes in Computer Sciences; Springer: Berlin/Heidelberg, Germany, 1986; Volume 218, pp. 417–426.
5. Diffie, W.; Hellman, M.E. New directions in cryptography. *IEEE Trans. Inf. Theory* **1976**, *IT-22*, 644–654. [CrossRef]
6. El Gamal, T. A public key cryptosystem and signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theory* **1985**, *IT-31*, 496–473. [CrossRef]
7. Johnson, D.; Menezes, A.; Vanstone, S. The elliptic curve digital signature algorithm (ECDSA). *Int. J. Inf. Secur.* **2001**, *1*, 36–63. [CrossRef]
8. Kaliski, B.S. A pseudo-random bit generator based on elliptic logarithms. In *Advances in Cryptology—CRYPTO'86*; Odlyzko, A.M., Ed.; Springer: Berlin/Heidelberg, Germany, 1986; Volume 263, pp. 84–103.
9. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2009. Available online: http://bitcoin.org/bitcoin.pdf (accessed on 25 December 2024).
10. Rescorla, E. The Transport Layer Security (TLS) Protocol Version 1.3. IETF TLS Workgroup. RFC 8446. Proposed Standard. August 2018. Available online: https://www.rfc-editor.org/info/rfc8446 (accessed on 25 December 2024)
11. Rivest, R.L. Cryptography and machine learning. In *Advances in Cryptology—ASIACRYPT '91*; Imai, H., Rivest, R.L., Matsumoto, T., Eds.; Springer: Berlin/Heidelberg, Germany, 1993; pp. 427–439.
12. Volna, E.; Kotyrba, M.; Kocian, V.; Janosek, M. Cryptography based on neural network. In Proceedings of the ECMS, Koblenz, Germany, 29 May–1 June 2012; pp. 386–391.
13. Noura, H.; Samhat, A.E.; Harkouss, Y.; Yahiya, T.A. Design and realization of a new neural block cipher. In Proceedings of the 2015 International Conference on Applied Research in Computer Science and Engineering (ICAR), Beirut, Lebanon, 8–9 October 2015; pp. 1–6.

14. Sagar, V.; Kumar, K. A symmetric key cryptographic algorithm using counter propagation network (cpn). In Proceedings of the 2014 International Conference on Information and Communication Technology for Competitive Strategies, Udaipur Rajasthan, India, 14–16 November 2014; pp. 1–5.
15. Kalsi, S.; Kaur, H.; Chang, V. DNA Cryptography and Deep Learning using Genetic Algorithm with NW algorithm for Key Generation. *J. Med. Syst.* **2018**, *42*, 17. [CrossRef] [PubMed]
16. Abadi, M.; Andersen, D.G. Learning to protect communications with adversarial neural cryptography. *arXiv* **2016**, arXiv:1610.06918.
17. Saini, A.; Sehrawat, R. Enhancing Data Security through Machine Learning-based Key Generation and Encryption. *Eng. Technol. Appl. Sci. Res.* **2024**, *14*, 14148–14154. [CrossRef]
18. Singh, P.; Pranav, P.; Anwar, S.; Dutta, S. Leveraging generative adversarial networks for enhanced cryptographic key generation. *Concurr. Comput. Pract. Exp.* **2024**, *36*, e8226. [CrossRef]
19. Kumar, S.; Sharma, D. Key Generation in Cryptography Using Elliptic-Curve Cryptography and Genetic Algorithm. *Eng. Proc.* **2023**, *59*, 59. [CrossRef]
20. Nitaj, A.; Rachidi, T. Applications of Neural Network-Based AI in Cryptography. *Cryptography* **2023**, *7*, 39. [CrossRef]
21. Benamira, A.; Gerault, D.; Peyrin, T.; Tan, Q.Q. A Deeper Look at Machine Learning-Based Cryptanalysis. In *Advances in Cryptology EUROCRYPT 2021. EUROCRYPT 2021*; Lecture Notes in Computer Science; Canteaut, A., Standaert, F.X., Eds.; Springer: Cham, Switzerland, 2021; Volume 12696.
22. Baracaldo, A.N. Oprea: Machine Learning Security and Privacy. *IEEE Secur. Priv.* **2022**, *20*, 11–13. [CrossRef]
23. Talukder, M.A.; Islam, M.M.; Uddin, M.A.; Hasan, K.F.; Sharmin, S.; Alyami, S.A.; Moni, M.A. Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction. *J. Big Data* **2024**, *11*, 33. [CrossRef]
24. Dini, P.; Elhanashi, A.; Begni, A.; Saponara, S.; Zheng, Q.; Gasmi, K. Overview on Intrusion Detection Systems Design Exploiting Machine Learning for Networking Cybersecurity. *Appl. Sci.* **2023**, *13*, 7507. [CrossRef]
25. Alani, M.M. Neuro-cryptanalysis of DES and triple-DES. In Proceedings of the International Conference on Neural Information Processing, Doha, Qatar, 12–15 November 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 637–646.
26. Maghrebi, H.; Portigliatti, T.; Prouff, E. Breaking cryptographic implementations using deep learning techniques. In *Security, Privacy, and Applied Cryptography Engineering—Proceedings of the International Conference on Security, Privacy, and Applied Cryptography Engineering Hyderabad, India, 14–18 December 2016*; Springer: Cham, Switzerland, 2016; pp. 3–26.
27. Tellez, F.; Ortíz, J. Comparing AI Algorithms for Optimizing Elliptic Curve Cryptography Parameters in E-Commerce Integrations: A Pre-Quantum Analysis. *Int. J. Adv. Comput. Sci. Appl.* **2024**, *15*, 1539–1553. https://arxiv.org/abs/2310.06752 [CrossRef]
28. Villegas, F.I.L.; Cordero, C.V. Machine Learning Analysis for Side-Channel Attacks over Elliptic Curve Cryptography. In Proceedings of the 2021 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON), ValparaÃso, Chile, 6–9 December 2021; pp. 1–7.
29. Weissbart, L.; Picek, S.; Batina, L. One Trace Is All It Takes: Machine Learning-Based Side-Channel Attack on EdDSA. In *Security, Privacy, and Applied Cryptography Engineering, SPACE 2019*; Bhasin, S., Mendelson, A., Nandi, M., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2019; Volume 11947.
30. Wøien, M.C.; Catak, F.O.; Kuzlu, M.; Cali, U. Neural Networks Meet Elliptic Curve Cryptography: A Novel Approach to Secure Communication. *arXiv* **2024**, arXiv:2407.08831.
31. Chinbat, T.; Madanian, S.; Airehrour, D.; Hassandoust, F. Machine learning cryptography methods for IoT in healthcare. *BMC Med. Inform. Decis. Mak.* **2024**, *24*, 153. [CrossRef]
32. Rosenblatt, F. The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychol. Rev.* **1958**, *65*, 386–408. Available online: https://www.ling.upenn.edu/courses/cogs501/Rosenblatt1958.pdf (accessed on 25 December 2024) [CrossRef]
33. Edwards, H.M. A normal form for elliptic curves. *Bull. Amer. Math. Soc.* **2007**, *44*, 393–422. [CrossRef]
34. Bernstein, D.J.; Lange, T. Faster addition and doubling on elliptic curves. In *Advances in Cryptology—ASIACRYPT 2007*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 29–50.
35. Montgomery, P.L. Speeding the Pollard and elliptic curve methods of factorization. *Math. Comput.* **1987**, *48*, 243–264. [CrossRef]
36. Koblitz, N. CM-curves with good cryptographic properties. In *Advances in Cryptology-Crypto'91*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1992; Volume 576, pp. 279–287.
37. Pornin, T. Efficient and Complete Formulas for Binary Curves Cryptology ePrint Archive, Paper 2022/1325. Available online: https://eprint.iacr.org/2022/1325 (accessed on 25 December 2024).
38. ANSI X9.62. Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). 2005. Available online: https://standards.globalspec.com/std/1955141/ansi-x9-62 (accessed on 25 December 2024).
39. *IEEE Std 1363-2000*; IEEE Standard Specifications for Public-Key Cryptography. 2000. Available online: https://ieeexplore.ieee.org/document/891000 (accessed on 25 December 2024).

40. Hankerson, D.; Vanstone, S.; Menezes, A. *Guide to Elliptic Curve Cryptography*; Springer: New York, NY, USA, 2004.
41. Brent, R.P. An improved monte carlo factorization algorithm. *Bit Numer. Math.* **1980**, *20*, 176–184. [CrossRef]
42. Teske, E. On random walks for Pollard's rho method. *Math. Comput.* **2000**, *70*, 809–825. [CrossRef]
43. Oorschot, P.C.V.; Wiener, M.J. Parallel collision search with cryptanalytic applications. *J. Cryptol.* **1999**, *12*, 1–28. [CrossRef]
44. Pohlig, S.; Hellman, M. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Trans. Inf. Theory* **1977**, *24*, 106–110. [CrossRef]
45. Kocher, P. Timing attacks on implementations of Diffie-Hellmann, RSA, DSS, and other systems. In *CRYPTO'96*; LNCS 1109; Springer: Berlin/Heidelberg, Germany, 1996; pp. 104–113.
46. Kocher, P.; Jaffe, J.; Jun, B. Differential power analysis. In Proceedings of the 19th International Advances in Cryptology Conference, CRYPTO'99, Barbara, CA, USA, 15–19 August 1999; pp. 388–397.
47. Boneh, D.; DeMillo, R.; Lipton, R. On the Importance of Checking Cryptographic Protocols for Faults. In *Advances in Cryptology, Proc. EUROCRYPT'97*; Fumy, W., Ed.; Springer: Berlin/Heidelberg, Germany, 1997; pp. 37–51.
48. Joye, M. Elliptic curves and side-channel analysis. *ST J. Syst. Res.* **2003**, *4*, 283–306.
49. Brier, E.; Joye, M. Weierstrass elliptic curves and side-channel attacks. In *PKC 2002*; LNCS; Springer: Berlin/Heidelberg, Germany, 2002; pp. 335–345.
50. Izu, T.; Takagi, T. A fast parallel elliptic curve multiplication resistant against side channel attacks. In *PKC 2002*; LNCS 2274; Springer: Berlin/Heidelberg, Germany, 2002; pp. 371–374.
51. Biehl, I.; Meyer, B.; Müller, V. Differential fault attacks on elliptic curve cryptosystems. In *CRYPTO 2000: Advances in Cryptology*; LNCS 1880; Springer: Berlin/Heidelberg, Germany, 2000; pp.131–146.
52. Shor, P.W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.* **1997**, *26*, 1484–1509. [CrossRef]
53. Proos, J.; Zalka, C. Shor's discrete logarithm quantum algorithm for elliptic curves. *arXiv* **2003**, arXiv:quant-ph/0301141. [CrossRef]
54. Hhan, M.; Yamakawa, T.; Yun, A. Quantum Complexity for Discrete Logarithms and Related Problems, Cryptology ePrint Archive, Paper 2023/1054. 2023. Available online: https://eprint.iacr.org/2023/1054 (accessed on 25 December 2024).
55. Liu, X.; Yang, H.; Yang, L. Minimizing CNOT-count in quantum circuit of the extended Shor's algorithm for ECDLP. *Cybersecurity* **2023**, *6*, 48. [CrossRef]
56. Shanks, D. Class Number, A Theory of Factorization, and Genera. *Am. Math. Soc. Proc. Symp. Pure Math.* **1971**, *20*, 415–440.
57. Menezes, A.J.; Okamoto, T.; Vanstone, S.A. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Trans. Inf. Theory* **1993**, *39*, 1639–1646. [CrossRef]
58. Semaev, I. Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p. *Math. Comput.* **1998**, *67*, 353–356. [CrossRef]
59. Smart, N.P. The discrete logarithm problem on elliptic curves of trace one. *J. Cryptol.* **1999**, *12*, 110–125. [CrossRef]
60. Satoh, T.; Araki, K. Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves. *Comment. Math. Univ. Sancti Pauli* **1998**, *47*, 81–92.
61. Frey, G.; Rück, H.-G. A remark concerning m-divisibility and the discrete logarithm problem in the divisor class group of curves. *Math. Comp.* **1994**, *62*, 865–874.
62. Pollard, J.M. A Monte Carlo method for factorization. *BIT Numer. Math.* **1975**, *15*, 331–334. [CrossRef]
63. Joux, A.; Odlyzko, A.; Pierrot, C. The Past, Evolving Present, and Future of the Discrete Logarithm. In *Open Problems in Mathematics and Computational Science*; Koç, Ç., Ed.; Springer: Cham, Switzerland, 2014.
64. Bernstein, D.J.; Lange, T. SafeCurves: Choosing Safe Curves for Elliptic-Curve Cryptography. 2013. Available online: https://safecurves.cr.yp.to (accessed on 25 December 2024).
65. Singh, P.; Dutta, S.; Pranav, P. Optimizing GANs for Cryptography: The Role and Impact of Activation Functions in Neural Layers Assessing the Cryptographic Strength. *Appl. Sci.* **2024**, *14*, 2379. [CrossRef]
66. Chhaybi, A.; Lazaar, S. System call frequency analysis based generative adversarial network model for zero day detection on mobile devices. *Int. J. Electr. Comput. Eng. (IJECE)* **2024**, *14*, 1969–1978. [CrossRef]
67. Kota, S.; Padmanabhuni, V.N.; Budda, K.; Sruthi, K. Authentication and encryption using modified elliptic curve cryptography with particle swarm optimization and cuckoo search algorithm. *J. Inst. Eng. Ser. B* **2018**, *99*, 343–351. [CrossRef]
68. Das, P.P.; Tawadros, D.; Wiese, L. Privacy-Preserving Medical Data Generation Using Adversarial Learning. In *Information Security. ISC 2023*; Lecture Notes in Computer Science; Athanasopoulos, E., Mennink, B., Eds.; Springer: Cham, Switzerland, 2023; Volume 14411.
69. Kumar, S.; Sharma, D. A chaotic-based image encryption scheme using elliptic curve cryptography and genetic algorithm. *Artif. Intell. Rev.* **2024**, *57*, 87. [CrossRef]
70. Jebrane, J.; Lazaar, S. An enhanced and verifiable lightweight authentication protocol for securing the Internet of Medical Things (IoMT) based on CP-ABE encryption. *Int. J. Inf. Secur.* **2024**, *23*, 3691–3710. [CrossRef]

71. Maimuţ, D.; Matei, A.C. Speeding-Up Elliptic Curve Cryptography Algorithms. *Mathematics* **2022**, *10*, 3676. [CrossRef]
72. Ribaric, T.; Houghten, S. Genetic programming for improved cryptanalysis of elliptic curve cryptosystems. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastián, Spain, 5–8 June 2017; pp. 419–426.
73. Mullai, A.; Mani, K. Enhancing the security in RSA and elliptic curve cryptography based on addition chain using simplified Swarm Optimization and Particle Swarm Optimization for mobile devices. *Int. J. Inf. Technol.* **2020**, *13*, 551–564. [CrossRef]
74. Selvi, S.; Gobi, M.; Kanchana, M.; Mary, S. Hyper elliptic curve cryptography in multi cloud-security using DNA (genetic) techniques. In Proceedings of the 2017 International Conference on Communication and Multimedia Computing (ICCMC), Erode, India, 18–19 July 2017; pp. 934–939.
75. Kashyap, U.; Padhi, S.K.; Ali, S.S. Attack GAN (AGAN): A new Security Evaluation Tool for Perceptual Encryption. *arXiv* **2024**, arXiv:2407.06570.
76. Zhou, L.; Chen, J.; Zhang, Y.; Su, C. Marino Anthony James: Security analysis and new models on the intelligent symmetric key encryption. *Comput. Secur.* **2019**, *80*, 14–24. [CrossRef]
77. Bernstein, D.; Duif, N.; Lange, T.; Schwabe, P.; Yang, B. High-Speed High-Security Signatures, Cryptology ePrint Archive, Paper 2011/368. 2011. Available online: https://eprint.iacr.org/2011/368 (accessed on 25 December 2024).
78. Bellare, M.; Rogaway, P. Minimizing the use of random oracles in authenticated encryption schemes. In *Information and Communications Security*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1997; Volume 1334 pp. 1–16.