The Impact of Quantum Computing on Cryptocurrencies

Po-Chun Wu, Zi-Li Lien, Yu-Chia Kuo

June 11, 2025

Abstract

Cryptocurrencies have gone viral in recent years, with Bitcoin being the most influential cryptocurrency. In this paper, we first introduce the protocol behind Bitcoin transactions, namely ECDSA, and classical cryptanalysis. Then, we analyze the construction of the quantum algorithm to solve ECDLP. Finally, we discuss some possible solutions with the aid of Post-Quantum Cryptography.

Presentation Video

Contents

1 Introduction

1.1 Cryptocurrency: Why They Exist

In the modern era, transactions are inevitable. Traditionally, they revolve around banks that constantly move gold bricks from chamber to chamber somewhere in the basement of a building in New York City. Every transaction requires a physical bank and physical gold bricks to be involved; however, as society thrives, the need for decentralized currency begins to emerge.

Among all competitors, Bitcoin is the most famous and is also the first modern cryptocurrency to successfully penetrate the market. Hence, in the following discussion, we will use the Bitcoin network system as the foundation and compare other cryptocurrencies along the way.

1.2 Bitcoin: How it works

The goal of the Bitcoin system is to record all transactions and create public consensus on which transactions are valid. In other terms, when users create fake transactions or modify current ones, the system corrects itself and "kicks out" those that "most of the other users" don't agree on. To provide security and trust, there are two layers of protection against fraud: the digital signature and the blockchain network.

Whenever a Bitcoin wallet is created, the system generates a public key Q and a private key d for the wallet (which represents the user). The public key is publicly accessible, and it is the "address" of the user's wallet to which others can send, while the private key is held by themselves privately. For each transaction, there is a message to be sent, including the details of the transaction, and the hash of the previous transaction that this one is based on. We denote this "message" as m. The main idea of checking if a transaction is "valid" is by checking if the user who created it has their own private key; if not, the transaction should be rejected. This is where the digital signature steps in.

A digital signature is a computed result of an algorithm that takes the message m and the private key d as inputs, and outputs a signature S. The signature should be significantly different for each transaction albeit using the same private key, and it should not be able to be used to reverse-compute

d. Nevertheless, by using another algorithm that takes S, m, and the public key Q as input, the signature S can be verified if it was created by using the private key d. Hence, everyone can verify if S is valid, but only the user with the private key d can create it. This mechanism prevents others from stealing money by creating fake transactions, adding the first layer of protection to the system.

After a transaction is made, it is broadcast to the pool of unconfirmed transactions, waiting for the community to verify it and add it to the shared list of previous transactions. Every user in the network can grab transactions from the pool to form a "block". In order for the block to be valid, the user should find a specific "nonce", a pseudo-random number that, if it is added to the block, the hash of the block would fulfill a criterion. After the nonce is found, the block is verified and is broadcast to the other users. Since finding the nonce is computationally-difficult, this procedure is called the "proof of work" since each block is verified only when the users spend computation time and energy to verify it. The computation is rewarded by bitcoins, and hence it is also called "mining".

When a block is broadcast, since the block also contains the previous block's hash, other users can keep the block and append it to the back of the previous block. However, if multiple blocks are received simultaneously, they would be all linked to the same block. Since the extension of blocks is done by every user in the community, there would eventually be a block that more blocks are extended from, making it the longest chain in the conflicting blocks, which would be preferred by the users. Since the proof of work requires blocks to be created at a slow pace, and every user actively participates in the creation of the blockchain, if a user wants to create a fake block, it would need to compete with the real block maintained by the whole community for it to be recognized by other users. Hence, for a user to fake a block, it should calculate the proof of work faster than all the other users in the network, and the chain out of the fake block should also be extended at a faster pace to outrun the real block. This is how the whole network of users make defrauding nearly impossible.

By using a digital signature, transaction records can be verified to ensure they were created by their intended creators. By using the blockchain network, users would need to compute at a near-impossible speed to beat the community, which ensures the validity of each block in the chain. By using these two protection mechanisms, the Bitcoin system can successfully record all transactions and create public consensus on which ones are valid.

1.3 How to Break the Bitcoin

Two components in the Bitcoin system are based on the concept of "impossible to reverse-engineer," which makes it secure: the digital signature generation procedure and the hash function. As mentioned previously, the digital signature ensures the transaction is genuine, and the hash function is used to create the chain such that blocks cannot be replaced without changing the hash. Moreover, the fact that the hash function is hard to reverse makes the proof of work useful in preventing fraud. Therefore, if attackers can reverse the digital signature generation procedure, they can get hold of the users' private keys and hence generate fake transactions. If the attackers can reverse the hash function and speed up the proof of work, they can generate longer blocks at a faster rate, out-compete the other users in the network, and thus successfully insert fake blocks into the blockchain.

The hash function, which is SHA256 in Bitcoin, is a function that maps an input of any length to a pseudo-random output of 256 bits. Trying to reverse this function means finding the input sequence such that the output is a specific value, which is the same as the "unstructured search problem". Since Grover's algorithm only has a quadratic speed-up to solving this problem, simply extending the bit size of the hash function by a multiple of 2 can counteract this algorithm. Hence, our main focus of discussion would be on the digital signature algorithm and the quantum advantage of solving it.

2 Elliptic Curve Digital Signature Algorithm (ECDSA): Mechanism and Security

The goal of the digital signature algorithm is to generate a signature S from the private key d and the message m such that it can be verified using only the public key Q. Moreover, it should be computationally difficult to obtain d from S, m, and Q, which the public gets access to. To achieve the goal, the algorithm leverages the difficulty of the "Elliptic Curve Discrete Logarithm Problem (ECDLP)", a variation of the discrete logarithm problem.

2.1 Elliptic Curve Discrete Logarithm Problem (ECDLP)

Problem:
Given an elliptic curve $y^2 = x^3 + ax + b$, a point on the curve G, order N of G such that $N \cdot G = O$, and another point on the curve Q. Find the integer d between 1 and $N - 1$ such that $Q = d \cdot G$.

The "$\cdot$" operator is the scalar multiplication on an Abelian group, which means repeated addition on the group. The point O is the point at infinity. Addition $C = A + B$ on the group is defined as finding the third point C' crossed by the chord that extends from A to B on the elliptic curve, and reflecting that point C' by the x-Axis to gain the result C. If $A = (x_1, y_1)$ and $B = (x_2, y_2)$, then

$$A + B = \begin{cases} O, & \text{if } (x_2, y_2) = (x_1, -y_1) \\ (x_3, y_3), & \text{otherwise} \end{cases} \tag{1}$$

where $x_3 = \lambda^2 - (x_1 + x_2),\ y_3 = \lambda(x_1 - x_3) - y_1,$

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

The difficulty of this problem is quite intuitive. Given two points on the curve and a special operator related to it, it is almost impossible to find how many operations it should take for one point to reach the other one without brute-forcing through the search process. For a 256-bit ECDLP, it would take $2^{255} \approx 5.79 \times 10^{76}$ attempts on average to guess the targeted integer d. This is why the difficulty of the problem is leveraged to secure the digital signature.

2.2 Digital Signature Generation

In the Bitcoin protocol, an elliptic curve **secp256k1** $y^2 = x^3 + 7$ and a special point on the curve called the "generator point" G are predefined for all users. For each user, they have a random private key d, an integer between 1 and 2^{256}, and a public key Q that is calculated by $Q = d \cdot G$.

After each transaction, the digital signature is generated through the process as follows:

Algorithm 1 The Digital Signature Generation

```
1: Calculate $z = HASH(m)$, where $m$ is the message to be signed
2: Pick a random integer $k$ between 1 and $2^{256}$
3: Calculate $(x_1, y_1) = k \cdot G$
4: Calculate $r = x_1 \mod N$, if $r = 0$, go back to step 2
5: Calculate $s = k^{-1}(z + r \cdot d) \mod N$, if $s = 0$, go back to step 2
6: The signature $S$ is generated as $(r, s)$
```

2.3 Digital Signature Verification

After each transaction ends, the signature is attached behind the transaction details and is open for every user to append to their blocks. Before appending the transaction and starting the "proof of work" process, the system checks if the signature is valid by the following process:

Algorithm 2 Digital Signature Verification

```
1: Check if $r, s$ are integers between 1 and $2^{256} - 1$, if not, the signature is
   corrupted
2: Calculate $z = HASH(m)$, where $m$ is the message signed
3: Calculate $w = s^{-1} \mod N$
4: Calculate $u_1 = z \cdot w \mod N, u_2 = r \cot w \mod N$
5: Calculate $(x'_1, y'_1) = u_1 \cdot G + u_2 \cdot Q$
6: Calculate $r' = x'_1 \mod N$
7: Accept the signature if $r' = r$
```

Since the checking algorithm only requires the generator point G, the message m, and the public key Q to verify if $S = (r, s)$ is correct, it allows the user to keep their private key d without ever having the need of sharing it to the public. Moreover, if someone wants to recover the private key d from G, m, Q, and S, they have to solve the ECDLP stated above, which is pretty hard for classical computers. Therefore, people using Bitcoin can make transactions securely without their private key ever being exposed.

2.4 Why the Verification Process Works

In the fifth step of the verification process,

$$\begin{aligned}
(x_1', y_1') &= u_1 \cdot G + u_2 \cdot Q \\
&= u_1 \cdot G + u_2 \cdot (d \cdot G) \\
&= (u_1 + du_2) \cdot G \\
&= (zw + drw) \cdot G \\
&= w(z + dr) \cdot G \\
&= s^{-1}(z + dr) \cdot G \\
&= (k^{-1}(z + dr))^{-1}(z + dr) \cdot G \\
&= k(z + dr)^{-1}(z + dr) \cdot G \\
&= k \cdot G \\
&= (x_1, y_1)
\end{aligned}$$

Hence, for $r = x_1 \mod N, r' = x_1' \mod N$, $r = r'$.

3 Classical Cryptanalysis of ECDSA

As previously mentioned, ECDSA is difficult to solve, and by brute-forcing through all possibilities, it might take longer than the existence of the universe to solve for only one private key that is 256 bits long.

However, there are still classical algorithms that are known to be faster than brute-forcing through every choice of d. According to Shoup [1], the lower bound of generic algorithms (those that don't exploit specific weaknesses of the protocol) that solve the discrete logarithm problem in a group of order N is $\Omega(\sqrt{N})$. Hence, before discussing the quantum threat to ECDSA, we first study an asymptotically-optimal classical attack that solves ECDLP: the Pollard's Rho attack.

3.1 Floyd's Cycle Finding Algorithm

The Pollard's Rho attack is a variant of the Floyd's cycle finding algorithm whose goal is to find the starting point of a cycle in a linked list. This algorithm is called the "Tortoise and Hare algorithm" because it uses a fast pointer and a slow pointer to search for collisions (same values) in the linked list in $O(1)$ space complexity.

For this algorithm, we define x_i as the i-th element of the linked list. The steps of the algorithm are as follows.

Algorithm 3 Floyd's Cycle Finding Algorithm

```
1: Let $p_f$: fast pointer = $x_0$, $p_s$: slow pointer = $x_0$
2: For each iteration, $p_f$ goes two steps and $p_s$ goes one steps
3: After $i$ iterations, $p_f$ and $p_s$ meet, then $p_s$ goes back to $x_0$ and $p_f$ switches
   to moving one step at a time
4: After another $j$ iterations, $p_f$ and $p_s$ meet again, and this point is the
   start of the cycle in the linked list
```

In this algorithm, we denote the length of the cycle (period) λ, and the length of the path from x_0 to the start of the cycle as μ. When p_f and p_s first meet after i iterations, $p_f = 2i$, $p_s = i$. We let $i = \mu + l_s\lambda + y$, $2i = \mu + l_f\lambda + y$, where y is the length between the start of the cycle and the meeting point mod *lambda*, while l_s and l_f are the times p_s and p_f went through the cycle, respectively. Hence,

$$\begin{aligned} 2i = 2\mu + 2l_s\lambda + 2y &= \mu + l_f\lambda + y \\ \mu + (2l_s - l_f)\lambda + y &= 0 \\ \mu + (2l_s - l_f + 1)\lambda &= \lambda - y \\ \mu &\equiv \lambda - y \pmod{\lambda} \end{aligned}$$

That is, after p_s and p_f meet, the next time they meet is at the starting point of the cycle, which is $\mu + l\lambda$ for some integer l. The time complexity of this algorithm is $O(\mu + \lambda)$, while the space complexity is $O(1)$.

3.2 Pollard's Rho Attack

The Pollard's Rho is an application of the Floyd's cycle finding algorithm, with the linked list mentioned above replaced with a special function such that $x_{i+1} = f(x_i)$, which functions similarly to a linked list.

In order to use the Floyd's cycle finding algorithm to solve ECDLP, we need to have a "cycle" for the algorithm to find. As seen in the fifth step in the verification process of the ECDSA, (x_1', y_1') are generated by $u_1 \cdot G + u_2 \cdot Q = (u_1 + du_2) \cdot G$, where d is the private key to be solved for. Thus, if we can find another pair of (u_1', u_2') such that

$$(u_1' + du_2') \cdot G \equiv (u_1 + du_2) \cdot G \pmod{N}$$

, then we can obtain $d \equiv \frac{u_1 - u_1'}{u_2' - u_2} \pmod{N}$, and the private key is found.

To find the pair (u_1, u_2) and (u_1', u_2'), the algorithm defines a function

$$R_{i+1} = f(R_i) = \begin{cases} R_i + G, & R_i \in S_0 \\ 2R_i, & R_i \in S_1 \\ R_i + Q, & R_i \in S_2 \end{cases}$$

where R_i is the i-th element in the "linked-list", and S_0, S_1, S_2 are subgroups on the curve. In terms of u,

$$(u_1^{i+1}, u_2^{i+1}) = f(u_1^i, u_2^i) = \begin{cases} (u_1^i + 1, u_2^i), & (u_1^i + du_2^i) \cdot G \in S_0 \\ (2u_1^i, 2u_2^i), & (u_1^i + du_2^i) \cdot G \in S_1 \\ (u_1^i, u_2^i + 1), & (u_1^i + du_2^i) \cdot G \in S_2 \end{cases}$$

This function generates a pseudo-random sequence (u_1^0, u_2^0), (u_1^1, u_2^1), (u_1^2, u_2^2), ... (u_1^i, u_2^i) such that the sequence will eventually enter a loop of length λ after μ iterations.

Now, this problem is transformed from finding a unique d such that $Q = d \cdot P$ into finding any pair (u_1, u_2) and (u_1', u_2') such that $(u_1 + du_2) \cdot G \equiv (u_1' + du_2') \cdot G \pmod{N}$. Since the order is N, there would be $N - 1$ unique values of d to guess, and it would on average take $\frac{d}{2}$ attempts by randomly guessing d. However, in the Pollard's Rho case, since there are only $N - 1$ unique values of R_i, and the algorithm tries its best to find two (u_1, u_2) pairs to match R_i, the probability is the same as guessing if there exist any two people in a room of r people whose birthday is the same, for N dates to pick from. This probability of r values, N possible outcomes, and checking if two values collide, is $1 - \prod_{i=0}^{r-1} \left(1 - \frac{i}{N}\right)$. As seen in [2], when r comes near to $\frac{\sqrt{\pi N}}{2}$, the probability of finding a collision in r steps is more than a 50%. Comparing the random-guessing case of time complexity $O(N)$ with Pollard's Rho attack of time complexity $O(\sqrt{N})$, it is a quadratic speedup, and it matches the lower bound in [1], which makes the algorithm asymptotically-optimal.

In conclusion, the fastest classical algorithm "Pollard's Rho" can solve the ECDLP in $O(\sqrt{N})$, which is approximately 3×10^{38} attempts until finding the private key of a single user using Bitcoin. If a computer tries 10^{12} times per second, it needs 30 quintillion (3×10^{19}) years to finish. Hence, Bitcoin is classically-impossible to break.

4 The Quantum Threat to ECDSA and Cryptocurrencies

As stated in Section 2, ECDLP is a variation of the discrete logarithm problem. Thus, it is not hard for one to imagine that the problem could be solved by quantum algorithms in a similar fashion to the way Peter Shor solved the discrete logarithm problem [3].

Since the quantum algorithm of ECDLP bears resemblance to that of the discrete logarithm, we can simply modify the latter to obtain an outline of the former algorithm.

Algorithm 4 Quantum algorithm for ECDLP

```
1: Prepare the state $|\phi\rangle = \frac{1}{p-1}\sum_{a,b\in\mathbb{Z}_{p-1}} |a\rangle|b\rangle|a\cdot G + b\cdot P\rangle = \frac{1}{p-1}\sum_{a,b\in\mathbb{Z}_{p-1}} |a\rangle|b\rangle|(a+db)\cdot G\rangle$
2: Perform measurement based on the third register, which collapses the state, and obtain $|\psi\rangle = \frac{1}{|S|}\sum_{a,b\in S} |a\rangle|b\rangle|r\cdot G\rangle$, where the set $S = \{a,b|a,b\in\mathbb{Z}_{p-1}, a+db \mod (p-1) = r\}$
3: Apply QFT $\mod p-1$ to each of the first two registers respectively, and measure to obtain $c_1, c_2 \in \mathbb{Z}_{p-1}$ $s.t.$ $dc_1 = c_2$.
4: Find $c_1^{-1}$ by the Extended Euclidean Algorithm classically and obtain $d = c_1^{-1}c_2 \mod p-1$.
```

Although the above outline is essentially the entire quantum algorithm for ECLDP, we still have to ensure that the quantum circuit is capable of computing group operations, namely $a\cdot G + b\cdot P$ in the first step.

Hence, we will define the group operations and modular operations in the quantum circuit necessary for the algorithm, as well as go through every step in detail in the following.

4.1 Modular Arithmetic for Quantum Circuits

4.1.1 Modular Addition

We will build this circuit based on the integer addition circuit:

$$|x\rangle|y\rangle|0\rangle \mapsto |x\rangle|(x+y)_{n-1\ldots0}\rangle|(x+y)_n\rangle$$

which is described by Takahashi et al. [4] The circuit performs addition of 2 n-bit binary numbers, and has $O(n)$ depth, $O(n)$ size with no ancilla qubits. Note that the subtraction circuit can be realized by the inverse of the circuit.

For the modular addition circuit, we prepare the 2 input registers, a constant p, and 2 carry qubits. We first perform an integer addition on the input registers, and subtract p from the addition results. If the first carry qubit is $|1\rangle$, it implies the original result is less than p and is now negative. Hence, we add back p with it being controlled by the first carry qubit. Finally, we compare the first two register to determine the value of the first carry qubit and uncompute it, to maintain the reversibility of the circuit. The whole process can be denoted as follows:

$$|x\rangle|y\rangle|0\rangle|0\rangle \xrightarrow{+} |x\rangle|(x+y)_{n-1\ldots0}\rangle|(x+y)_n\rangle|0\rangle$$

$$\xrightarrow{-p} |x\rangle|(x+y-p)_{n-1\ldots0}\rangle|(x+y-p)_n\rangle|(x+y-p)_{n+1}\rangle$$

$$\xrightarrow{C(+p)} |x\rangle|x+y \mod p\rangle|(x+y-p)_n\rangle|0\rangle$$

$$\xrightarrow{CX} |x\rangle|x+y \mod p\rangle|0\rangle|0\rangle$$

4.1.2 Modular Doubling

The circuit of modular doubling is similar to that of modular addition. One prepares an input register, a constant p, and 2 carry qubits. Then, replace the first integer addition operation with a left shift. Finally, since p must be odd, $2x \mod p$ must be odd if and only if p is subtracted from $2x$. Thus, the LSB of $|2x \mod p\rangle$ can be used as the control bit of the last CNOT gate.

$$|x\rangle|0\rangle|0\rangle \xrightarrow{SL} |(2x)_{n-1\ldots0}\rangle|(2x)_n\rangle|0\rangle$$

$$\xrightarrow{-p} |(2x-p)_{n-1\ldots0}\rangle|(2x-p)_n\rangle|(2x-p)_{n+1}\rangle$$

$$\xrightarrow{C(+p)} |2x \mod p\rangle|(2x-p)_n\rangle|0\rangle$$

$$\xrightarrow{CX} |2x \mod p\rangle|0\rangle|0\rangle$$

4.1.3 Modular Multiplication

For modular multiplication, we take advantage of the fact that multiplication of binary numbers can be represented as:

$$xy = (\sum_{i=0}^{n-1} 2^i x_i)y = x_0y + 2x_1y + 4x_2y + \cdots = x_0y + 2(x_1y + 2(x_2y + \cdots)),$$

which consists of a series of modular addition and modular doubling, since x_i is either 0 or 1, x_iy is essentially a controlled modular addition, with x_i

being the control bit. Therefore, the modular multiplication circuit can be depicted as:

$$|x\rangle|y\rangle|0\rangle \xmapsto{A} |x\rangle|y\rangle|x_{n-1}y\rangle \xmapsto{Dbl} |x\rangle|y\rangle|2x_{n-1}y\rangle$$

$$\xmapsto{A} |x\rangle|y\rangle|2x_{n-1}y + x_{n-2}y\rangle \xmapsto{Dbl} |x\rangle|y\rangle|2(2x_{n-1}y + x_{n-2}y)\rangle$$

$$\xmapsto{A} \cdots \xmapsto{Dbl} |x\rangle|y\rangle|2(2(2(\cdots + x_3y) + x_2y) + x_1y)\rangle$$

$$\xmapsto{A} |x\rangle|y\rangle|2(2(2(\cdots + x_3y) + x_2y) + x_1y) + x_0y\rangle = |x\rangle|y\rangle|xy \mod p\rangle,$$

where A and Dbl stands for Modular Addition and Modular Doubling, respectively.

4.1.4 Extended Euclidean Algorithm and Modular Inverse

The Euclidean Algorithm is well known for finding the greatest common divisor of two integers A and B. In short, the algorithm subtracts a multiple of the smaller integer from the larger one, until one of which becomes 0. The algorithm works thanks to the property:

$$\gcd(A, B) = \gcd(A - qB, B)$$

Moreover, the algorithm can be extended to obtain other information:

Problem:
Given two integers A and B, compute the greatest common divisor $\gcd(A, B)$, as well as integer coefficients a and b such that

$$aA + bB = \gcd(A, B)$$

Note that since $\gcd(A, B)$ are smaller than A or B, one of a and b must be negative. Here we introduce the Extended Euclidean Algorithm to solve the above problem:

Algorithm 5 The Extended Euclidean Algorithm

```
1: Prepare two Euclidean pairs $(a, A) = (0, A)$ and $(b, B) = (1, B)$
2: while $A \neq 0$ and $B \neq 0$ do
3:     if $A \geq B$ then
4:         $q \leftarrow \lfloor \frac{A}{B} \rfloor$
5:         $(a, A) \leftarrow (a - qb, A - qB)$
6:     else
7:         $q \leftarrow \lfloor \frac{B}{A} \rfloor$
8:         $(b, B) \leftarrow (b - qa, B - qA)$
9:     end if
10: end while
```

Notice that if we set $A = p, B = x$, since p is prime, $\gcd(p, x) = 1$. If we then run the Extended Euclidean Algorithm, we get coefficients such that:

$$ap + bx = 1,$$

which can be rewritten as:

$$\begin{cases} bx \mod p = 1, & \text{if } b > 0 \\ (b + p)x \mod p = 1, & \text{if } b < 0 \end{cases}$$

Implying either b or $b+p$ is the modular inverse of x by definition. Hence, we can obtain the modular inverse of an input by running the Extended Euclidean Algorithm.

4.1.5 Modular Division

With the aforementioned operation, we can construct the modular division circuit by leveraging a combination of the modular inverse and the modular multiplication, as follows:

$$|x\rangle|y\rangle|0\rangle \xmapsto{I} |x^{-1}\rangle|y\rangle|0\rangle \xmapsto{M} |x^{-1}\rangle|y\rangle|x^{-1}y\rangle$$
$$\xmapsto{I} |x\rangle|y\rangle|x^{-1}y\rangle \xmapsto{M^*} |x\rangle|0\rangle|x^{-1}y\rangle \xmapsto{SWAP} |x\rangle|x^{-1}y\rangle|0\rangle,$$

where M, I, and M^* stands for Modular Multiplication, Modular Inverse, and the reverse of Modular Multiplication, respectively.

4.2 Group Operations for Quantum Circuits

Now that we have defined the modular arithmetic, we can construct the group operations for quantum circuits based on the steps mentioned in Section 2.1.

4.2.1 Addition

Based on Eq.(1) and Eq.(2), the operation of adding point $B = (x_2, y_2)$ to point $A = (x_1, y_1)$ can be constructed as follows:

$$|x_1\rangle|y_1\rangle \overset{A}{\longmapsto} |x_1 - x_2\rangle|y_1 - y_2\rangle \overset{Div}{\longmapsto} |x_1 - x_2\rangle|\lambda\rangle$$
$$\overset{M}{\longmapsto} |x_3 - x_2\rangle|\lambda\rangle \overset{M}{\longmapsto} |x_3 - x_2\rangle|y_3 + y_2\rangle \overset{A}{\longmapsto} |x_3\rangle|y_3\rangle,$$

where $(x_3, y_3) = A+B$ is the result of group addition. A, Div, and M stands for Modular Addition, Modular Division, and Modular Multiplication, respectively.

4.2.2 Integer Multiples

To compute an integer multiple of a point P, we take advantage of the following property:

$$x \cdot P = \sum_{i=0}^{n-1} 2^i x_i \cdot P = \sum_{i=0}^{n-1} x_i \cdot P_i,$$

where $P_i = 2^i P$. P_i can be computed classically by the recursive equation $P_i = P_{i-1} + P_{i-1}$. Hence, the integer multiple operation is essentially a series of controlled group addition.

4.3 Quantum Algorithm for ECDLP

Finally, we describe the algorithm step by step, examine how the algorithm works, and analyze the required running time.

First, construct the equal superposition state $|\phi\rangle = \frac{1}{p-1}\sum_{a,b\in\mathbb{Z}_{p-1}} |a\rangle|b\rangle|0\rangle$ and apply the oracle U_f, where the quantum gate is given by:

$$U_f|x\rangle|y\rangle|z\rangle = |x\rangle|y\rangle|z \oplus (x \cdot G + y \cdot P)\rangle,$$

where the gate is constructed by group addition and integer multiple operations. We then get:

$$|\phi\rangle \xmapsto{U_f} \frac{1}{p-1} \sum_{a,b\in\mathbb{Z}_{p-1}} |a\rangle|b\rangle|a\cdot G + b\cdot P\rangle = \frac{1}{p-1} \sum_{a,b\in\mathbb{Z}_{p-1}} |a\rangle|b\rangle|(a+db)\cdot G\rangle$$

Then, a measurement based on the third register collapses the state into $|r\cdot G\rangle$ with an arbitrary r, thus the state becomes:

$$|\psi\rangle = \frac{1}{|S|} \sum_{a,b\in S} |a\rangle|b\rangle|r\cdot G\rangle,\ \ S = \{a,b|a,b\in\mathbb{Z}_{p-1}, a+db \mod (p-1) = r\}$$

Next, by applying QFT $\mod p-1$ to each of the first two registers, we obtain:

$$\begin{aligned}|\psi\rangle &= \frac{1}{|S|} \sum_{a,b\in S} |a\rangle|b\rangle|r\cdot G\rangle \\ &\xmapsto{Q_{p-1}\otimes Q_{p-1}\otimes I} \frac{1}{p-1}\frac{1}{|S|} \sum_{a,b\in S} \Big(\sum_{c_1\in\mathbb{Z}_{p-1}} \omega^{ac_1}|c_1\rangle\Big)\Big(\sum_{c_2\in\mathbb{Z}_{p-1}} \omega^{bc_2}|c_2\rangle\Big)|r\cdot G\rangle \\ &= \frac{1}{p-1}\frac{1}{|S|} \sum_{c_1,c_2\in\mathbb{Z}_{p-1}} \omega^{rc_1} \sum_{a,b\in S} \omega^{b(dc_1-c_2)}|c_1\rangle|c_2\rangle|r\cdot G\rangle,\end{aligned}$$

note that the term $\omega^{b(dc_1-c_2)}$ vanishes to zero after summation if $dc_1 - c_2 \neq 0$. Thus, we can measure the first two registers and obtain a pair (c_1, c_2) satisfying $dc_1 - c_2 = 0$.

Finally, we compute the modular inverse c_1^{-1} and obtain the private key $d = c_1^{-1}c_2$.

An analysis of the running time shows that the algorithm runs in $O(poly(n))$ time, where n is the number of binary bits of p.

As stated in Section 4.1.1, modular addition takes $O(n)$, and so does modular doubling, since they have similar structures. Modular multiplication consists of n modular addition and modular doubling, thus taking $O(n^2)$. Modular inverse, or the Extended Euclidean Algorithm, also takes $O(n^2)$, as analyzed in Proos' and Zalka's paper [5]. Therefore, modular division, which consists of modular multiplication and modular inverse, takes $O(n^2)$ as well. In general, the quantum gate U_f takes $O(poly(n))$ time as group addition and integer multiple in U_f are also combinations of the above operations.

Since the other steps are identical to those of the discrete logarithm problem, we can conclude that the whole algorithm runs in $O(poly(n))$ time, which is advantageous to classical's $O(\exp(n))$ time.

5 Post-Quantum Cryptography (PQC) as a Solution

As discussed in the previous chapter, the rise of quantum computing brings great risks to cryptography. Cryptocurrencies that use algorithms like ECDSA are especially at risk. The Shor algorithm can solve the ECDLP quickly, which is closely related to the security of private keys. Due to this, private keys may no longer be safe. This weakness can damage transaction security, as quantum computers may be able to fake ECDSA signatures [6, 7].

To fight against this new danger, Post-Quantum Cryptography (PQC) has grown. PQC works on making and setting standards for crypto systems that can resist attacks from both regular and quantum computers. This work is not just theory. It is shown by global efforts like the 2016 contest started by the US National Institute of Standards and Technology (NIST). The goal of this contest is to find and set standards for crypto-methods that can survive quantum attacks [8, 7].

5.1 Major Approaches to Post-Quantum Cryptography

PQC includes a variety of cryptographic approaches instead of offering a single solution. These approaches are based on mathematical problems that are currently considered computationally unfeasible, even for classical and quantum computers. Various PQC approaches have shown potential, each with its own strengths, limitations, and design trade-offs. During the NIST standardization effort, lattice-based, code-based, and multivariate methods proved to be among the strongest candidates [7].

5.1.1 Lattice-based Cryptography

Lattice-based cryptography has become one of the major types of post-quantum cryptographic candidates. Its security is grounded in the assumed computational difficulty of specific problems related to mathematical lattices, such as the Shortest Vector Problem (SVP) and Closest Vector Problem (CVP), as well as their algebraic extensions, including the Short Integer Solution (SIS) problem and the Learning With Errors (LWE) problem [7]. Despite significant research efforts, these problems remain resistant to efficient quantum solutions, even with algorithms like Shor's.

In classical computing, the primary method for solving lattice problems involves basis reduction, transforming an arbitrary lattice basis into a new set of shorter, nearly orthogonal vectors. Although algorithms such as Lenstra-Lenstra-Lovász (LLL) can approximate a reduced basis in polynomial time, their performance deteriorates significantly in high-dimensional spaces, which are standard in cryptographic settings [7].

Lattice-based schemes can be constructed for both public-key encryption (PKE), often realized as Key Encapsulation Mechanisms (KEMs), and digital signature algorithms. Prominent examples that have advanced in the NIST PQC standardization process include CRYSTALS-Kyber (a KEM) and CRYSTALS-Dilithium (a digital signature scheme) [8]. Another notable lattice-based system is NTRU, a finalist in the NIST PQC project, which is defined over a polynomial ring $\mathbb{Z}[X]/(X^N - 1)$. In NTRU, the private key consists of small polynomials f and g, and the public key is $h = f_q^{-1} \otimes g \pmod q$. Its security is related to the SVP in a specific lattice derived from these polynomials, where (f, g) is likely the shortest vector [7].

In general, key generation in many lattice-based signature schemes, such as Dilithium, involves generating a private key consisting of one or more vectors (or matrices) with small integer coefficients. The corresponding public key typically involves a publicly known matrix A and a vector t computed as $t = As_1 + s_2 \pmod q$, where s_1 and s_2 are components of the secret key and q is a modulus. The security rests on the difficulty of recovering the "small" secret vectors s_1 and s_2 given the public matrix A and the public vector t. This is related to the SIS and LWE problems. For example, Dilithium's security specifically relies on the hardness of the Module Learning With Errors (MLWE) and Module Short Integer Solution (MSIS) problems.

The Dilithium signature protocol typically relies on a Fiat–Shamir transformation augmented with rejection sampling. In this process, the signer first selects a random commitment and then derives a challenge by hashing both the message and that commitment. Using the secret key, the signer computes a response to this challenge. The subsequent rejection sampling step (often referred to as an "abort") forces the signature components, particularly the response vector, to remain within a small norm bound, thereby preventing any leakage of private-key information.

Experimental evaluations have shown that Dilithium can outperform ECDSA in key generation, signature creation, and signature verification within contemporary blockchain environments, while simultaneously offering stronger resistance to quantum adversaries [6]. These findings imply that transitioning to postquantum cryptography (PQC)-based wallets and transactions, by adopting PQC digital signature schemes, constitutes a practical strategy to protect against quantum-enabled counterfeiting [6].

5.1.2 Quantum Resistance of Code-based Cryptography

Code-based cryptosystems, such as Classic McEliece, owe their post-quantum security to the intrinsic difficulty of key problems in coding theory, most notably the syndrome decoding problem for arbitrary linear codes.

- **NP-Hardness of the Core Problem:** Determining a low-weight error vector for a general linear code (the Syndrome Decoding Problem) is NP-hard, which implies that no efficient classical algorithm is known. Although NP-hardness alone does not guarantee resistance against quantum algorithms, it nevertheless underscores the significant computational challenge.

- **No Suitable Structure for Shor's Algorithm:** Unlike integer factoring or discrete logarithms, decoding random linear codes does not exhibit the kind of algebraic periodicity that Shor's algorithm exploits. As a result, there is no clear way to apply Shor's method to achieve exponential speedups.

- **Grover's Algorithm Yields Only Quadratic Gains:** In principle, Grover's search could be used to locate a low-weight error vector more quickly. However, by choosing code parameters (length, dimension, and target error weight) sufficiently large, even a quadratic speedup leaves the attack's complexity beyond feasible limits for near-term quantum machines.

- **Decades of Unbroken Security:** Since its proposal, the McEliece scheme, particularly when instantiated with binary Goppa codes has resisted both classical and quantum cryptanalysis for over forty years. This enduring record provides strong empirical support for its robustness.

- **Obfuscation of the Hidden Structure:** Many code-based constructions hide an efficiently decodable code (the private key) behind a transformation that makes it appear as a random linear code (the public key). To date, no quantum algorithm has demonstrated an effective means of reversing this disguise, provided that the code parameters are chosen appropriately.

Hence, code-based cryptography is widely regarded as quantum-secure, given the absence of known quantum attacks against its foundational problems and the long-standing resistance of its flagship schemes to cryptanalytic efforts.

5.2 Other Notable PQC Families and Connections

In addition to lattice- and code-based approaches, several other classes of post-quantum cryptography have been extensively studied:

- **Hash-based Signatures:** Schemes such as SPHINCS+ (one of NIST's PQC selections) derive security solely from hash-function properties. Since quantum adversaries can only achieve a quadratic improvement (via Grover's algorithm) in searching for hash preimages, increasing the hash output size sufficiently counteracts that advantage [8].
- **Multivariate Cryptography:** The hardness here stems from solving large systems of multivariate polynomial equations over finite fields, a problem believed to be intractable for both classical and quantum computers.
- **Isogeny based Cryptography:** Schemes built on finding isogenies between supersingular elliptic curves offer a different security basis, although recent cryptanalytic progress has challenged some constructions.

Despite their distinct mathematical origins, code-based and lattice-based cryptosystems share several conceptual parallels. Both types of schemes involve a "hidden basis" for codes, a basis of codewords that allows efficient decoding; for lattices, a short basis enabling fast reduction. In the McEliece system, security reduces to the intractability of distinguishing a structured Goppa code from a uniformly random linear code. In a lattice-based scheme like NTRU, hiding a good (short) basis behind a "bad" public basis relies on the difficulty of lattice basis reduction. Moreover, the Minimum Distance Problem (MDP) in coding theory—finding the nonzero codeword with least Hamming weight—is closely analogous to the Shortest Vector Problem (SVP) in lattices, where one seeks the nonzero vector of minimal Euclidean norm. Similarly, the Nearest Codeword Problem (NCP) corresponds to the Closest Vector Problem (CVP) in lattices [7]. Some modern list-decoding algorithms for Goppa codes even incorporate lattice-basis-reduction techniques during decoding, further illustrating these interconnections [7].

5.3 Security Considerations: Quantum Resistance vs. Provable Security

When one says that a cryptographic primitive is "quantum-resistant" or "post-quantum," it means that, to the best of current knowledge, no quantum algorithm can solve its underlying mathematical problem in polynomial time. This designation depends on ongoing research in quantum computing and algorithm development. For NIST's selected PQC candidates, confidence in their security rests on decades of cryptanalytic scrutiny and the lack of effective quantum attacks against problems such as LWE, SIS, or decoding random linear codes.

It is important to emphasize that "quantum resistance" does not equate to unconditional security against all future attacks. Like classical cryptography, where RSA assumes factoring is hard, and ECDSA assumes the elliptic curve discrete logarithm problem is hard, post-quantum schemes rely on the conjectured intractability of certain problems even for quantum machines. "Provable security" refers to a formal reduction: breaking the cryptosystem implies solving the hard problem (for instance, Dilithium's security reduces to the presumed quantum-hardness of MLWE and MSIS). Thus, Dilithium, Kyber, and other NIST finalists possess security proofs that connect an adversary's success directly to solving an assumed difficult problem.

Still, provable security does not guarantee immunity to unknown algorithmic breakthroughs. The NIST PQC standardization process incorporates extensive public review and cryptanalysis to confidence in long-term security. At the same time, research continues on quantum algorithms and specialized attacks targeting PQC primitives.

By contrast, information-theoretic schemes, like the one-time pad, offer unconditional security but are impractical for applications such as digital signatures in blockchain systems, given requirements like enormous pre-shared keys. Other advanced paradigms, such as "Mistrustful Quantum Cryptography," which encompasses tasks like quantum money or secure multi-party quantum computation, operate under different assumptions and protocols and are not direct replacements for ECDSA in modern digital infrastructures. The ongoing effort in PQC focuses on designing practical, computationally secure primitives that can be deployed today to protect against the looming quantum threat.

6 Conclusion and Future Works

6.1 Conclusion

This paper started by dissecting the foundational security principles of modern cryptocurrencies, with a specific focus on Bitcoin. We established that the integrity of transactions and the ownership of assets are fundamentally guaranteed by the Elliptic Curve Digital Signature Algorithm (ECDSA). The security of ECDSA, in turn, relies on the classical intractability of the Elliptic Curve Discrete Logarithm Problem (ECDLP). While classical cryptanalysis, even with optimized algorithms like Pollard's Rho attack, poses no practical threat to the 256-bit security of Bitcoin, the arrival of fault-tolerant quantum computing presents an existential risk.

We showed that a quantum computer using a changed version of Shor's algorithm can break ECDLP in polynomial time. We explained the quantum circuits needed for modular math and elliptic curve operations. This proves the threat isn't just theory—it's a real engineering problem. Breaking ECDLP fast would destroy Bitcoin's security model and similar systems. Attackers could get private keys from public keys, letting them fake signatures and steal money.

Because of this threat, we studied Post-Quantum Cryptography (PQC) as a solution. We looked at main PQC types like lattice-based, code-based, and hash-based schemes. These systems rely on hard math problems, like the Shortest Vector Problem (SVP) in lattices and the Syndrome Decoding Problem in coding theory. These problems seem to resist attacks from both classical and quantum computers now. Options like CRYSTALS-Dilithium from the NIST process are strong replacements for ECDSA. They resist quantum attacks and sometimes work as well as or better than current systems.

6.2 Future Works

Building upon the findings of this paper, several critical aspects for future research and development emerge:

1. **Performance Benchmarking in Live Environments:** Theoretical analysis and standalone experiments show PQC schemes like Dilithium are efficient, but future work must focus on large-scale testing in existing blockchain systems. This should include how they affect signature size, verification time, transaction speed, and network delay.

2. **Refined Quantum Resource Estimation:** We can improve how we analyze the quantum threat. Future studies must focus on better estimates of resources needed to break Bitcoin's secp256k1 curve. These resources include logical qubits, T-gates, and total time. This work would help show how urgent the threat is as quantum hardware gets better.

3. **Transition Strategies and Hybrid Schemes:** Moving a multi-billion dollar system to a new cryptographic standard is a huge chal-

lenge. We need research to create safe and solid transition plans. This research should look at hybrid signature schemes. These combine a classical signature (like ECDSA) with a PQC signature. This helps guard against unknown weaknesses in new PQC algorithms and also protects against future quantum attackers.

4. **Economic and Game-Theoretic Analysis:** A future quantum attack or moving to PQC will have big effects on economics and game theory. Research should look at scenarios like the "first-mover advantage" for someone with a quantum computer, the reasons for or against miners using a PQC-hardened fork, and how the market might react to a "quantum-safe" cryptocurrency.

5. **Comprehensive Security Assessment:** This paper focused on digital signatures, but other cryptocurrency parts also need a full quantum-risk check. These include hash functions (which Grover's algorithm can attack faster), key-derivation functions, and wallet software. We need a complete security check to make sure there are no weak points in a post-quantum world.

7 Division of Work

Table 1: division of work table

Student Number	**Name**	**Works**
B11901054	Po-Chun, Wu	section 1, 2, 3, 6
B11901042	Zi-Li, Lien	section 4, 6
B11901047	Yu-Chia, Kuo	section 5, 6

8 Presentation Video

https://drive.google.com/file/d/1Sour8GXsfoWeXly_aD3FhsM0Oue-cWfl/view

References

[1] V. Shoup, "Lower bounds for discrete logarithms and related problems," in *Advances in Cryptology — EUROCRYPT '97* (W. Fumy, ed.), (Berlin, Heidelberg), pp. 256–266, Springer Berlin Heidelberg, 1997.

[2] A. Menezes, "Evaluation of security level of cryptography: The elliptic curve discrete logarithm problem (ecdlp)," 2001.

[3] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, p. 1484–1509, Oct. 1997.

[4] Y. Takahashi, S. Tani, and N. Kunihiro, "Quantum addition circuits and unbounded fan-out," 2009.

[5] J. Proos and C. Zalka, "Shor's discrete logarithm quantum algorithm for elliptic curves," 2004.

[6] A. C. H. Chen, "The security performance analysis of blockchain system based on post-quantum cryptography – a case study of cryptocurrency exchanges," 2024.

[7] A. Meyer, "Post-quantum cryptography: An analysis of code-based and lattice-based cryptosystems," 2025.

[8] S. Holmes and L. Chen, "Assessment of quantum threat to bitcoin and derived cryptocurrencies." Cryptology ePrint Archive, Paper 2021/967, 2021.

[9] S. Lokhande, I. Gupta, and D. Kulkarni, "A review: Solving ecdlp problem using pollard's rho algorithm," *IJARCCE*, vol. 6, pp. 377–380, 05 2017.

[10] Kriptomat, "A Short History of Cryptocurrencies." `https://kriptomat.io/cryptocurrencies/history-of-cryptocurrency/`, 2025.

[11] "Bitcoin: A Peer-to-Peer Electronic Cash System." `https://bitcoin.org/bitcoin.pdf`.

[12] S. Driscoll, "How Bitcoin Works Underhood." `https://www.imponderablethings.com/2013/07/how-bitcoin-works-underhood.html`.

[13] "Handout Elliptic Curve Crypto." `https://www.site.uottawa.ca/chouinar/HandoutEllipticCurveCrypto.pdf`.

[14] "ECDSA: Elliptic Curve Signatures." `https://cryptobook.nakov.com/digital-signatures/ecdsa-sign-verify-messages`.

[15] M. Roetteler, M. Naehrig, K. M. Svore, and K. Lauter, “Quantum resource estimates for computing elliptic curve discrete logarithms,” 2017.