SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

Registration number: FEI-xxxx-xxxx (TODO: have this resolved)

ELLIPTIC CURVE CRYPTOGRAPHY - SECURITY ANALYSIS AND ATTACK DEMONSTRATION

BACHELOR'S THESIS

2024 **Erik Ziman**

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

Registration number: FEI-xxxx-xxxx (TODO: have this resolved)

ELLIPTIC CURVE CRYPTOGRAPHY - SECURITY ANALYSIS AND ATTACK DEMONSTRATION

BACHELOR'S THESIS

Study Programme:	Applied Informatics
Study Field:	Computer Science
Training Workplace:	Institute of Computer Science and Mathematics
Supervisor:	Mgr. Karina Chudá, PhD.

Bratislava 2024 **Erik Ziman**

SÚHRN

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program:	Aplikovaná informatika
Autor:	Erik Ziman
Bakalárska práca:	Kryptografia na báze eliptických kriviek - analýza bezpečnosti a demonštrácia útokov
Vedúci záverečnej práce:	Mgr. Karina Chudá, PhD.
Miesto a rok predloženia práce:	Bratislava 2024

TODO (napisat po slovensky suhrn prace)

Kľúčové slová: kľúčové slovo1, kľúčové slovo2, kľúčové slovo3

ABSTRACT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

Study Programme:	Applied Informatics
Author:	Erik Ziman
Bachelor's thesis:	Elliptic curve cryptography - security analysis and attack demonstration
Supervisor:	Mgr. Karina Chudá, PhD.
Place and year of submission:	Bratislava 2024

TODO : write abstract

Keywords: keyword1, keyword2, keyword3

Acknowledgments

I would like to express a gratitude to my thesis supervisor.

List of Figures and Tables

List of Algorithm

List of listings

[bp,en]FEIstyle

mathtools amssymb float

includes/bibliography.bib

As Yale University professor Serge Lang once noted in the beginning of his book *Elliptic Curves: Diophantine Analysis*, “It is possible to write endlessly on elliptic curves. (This is not a threat)." Indeed, elliptic curves form a deep and intricate subject. In this thesis, we will dive into the topic of elliptic curves, with a particular focus on their impact in the field of modern cryptography. We aim to explore their significance and key properties, highlighting the crucial role they play in many cryptographic systems. The primary aim of this thesis is to deepen our understanding of this complex topic to such an extent that we are able to implement elliptic curve cryptography (ECC) in various applications and programs. In addition to exploring the theoretical significance of elliptic curves, we will also implement a few examples of elliptic curves referenced with corresponding source code and visual representation. Furthermore, this thesis will shed light on various attacks and common mistakes in the implementation of these curves, emphasizing the importance of secure and correct practices when building cryptographic applications.

Motivation for Using Elliptic Curves

The main advantage of ECC is the degree of security it provides when considering its comparatively smaller key sizes.

Table 1: Time to Break vs Key Sizes

Time to Break (MIPS-years)	**ECC Key Size (bits)**	**RSA Key Size (bits)**
10^4	106	512
10^8	132	768
10^{11}	160	1024
10^{20}	210	2048
10^{78}	600	21000

With the keys being smaller, we are able to have better computational efficiency of the algorithms; thus, our requirements on hardware resources can lower. Another advantage is that almost all currently known systems based on the discrete logarithm problem can be converted into elliptic curve-based systems. The vast majority of elliptic curve cryptography schemes rely on the Elliptic Curve Discrete Logarithm Problem (ECDLP)

for their security.

What is an Elliptic Curve?

In order to start defining elliptic curves, we first need to introduce some terms up front.

Algebraic curve:

An *algebraic curve* (over field K) is set of points (x, y) in the plane that satisfy a a non-constant polynomial equation in two variables. A *nonsingular algebraic curve* is algebraic curve without any singular points.

$$A := \{(x, y) \in K^2 \mid f(x, y) = 0\}$$

K-rational point:

A *K-rational point* is a solution (x, y) to the equation of an algebraic curve, where both x and y are in the field K.

$$P = (x_P, y_P) \quad \text{where } f(P) = 0 \text{ with } x_P, y_P \in K.$$

Point at infinity:

The point at infinity $\mathcal{O}$ is the identity element of elliptic curve arithmetic. Adding this point to any other point (including itself) results in that other point:

$$\mathcal{O} + P = P$$

$$\mathcal{O} + \mathcal{O} = \mathcal{O}$$

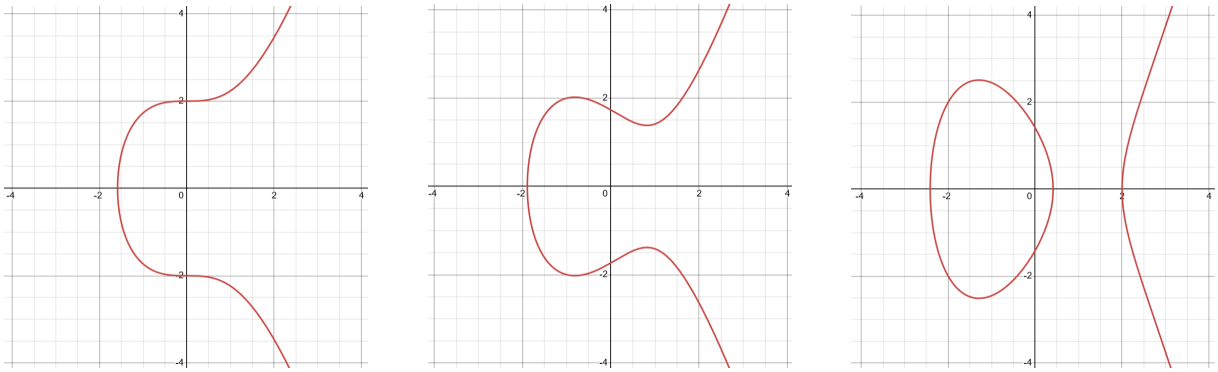
Elliptic curve:

An *elliptic curve* (over field K) is a nonsingular cubic curve, with at least 1 K-rational point. We will primarily be working with curves that are defined by The Weierstrass Form:

$$E := \{(x, y) \in K^2 \mid y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\} \quad \text{with } a, b \in K$$

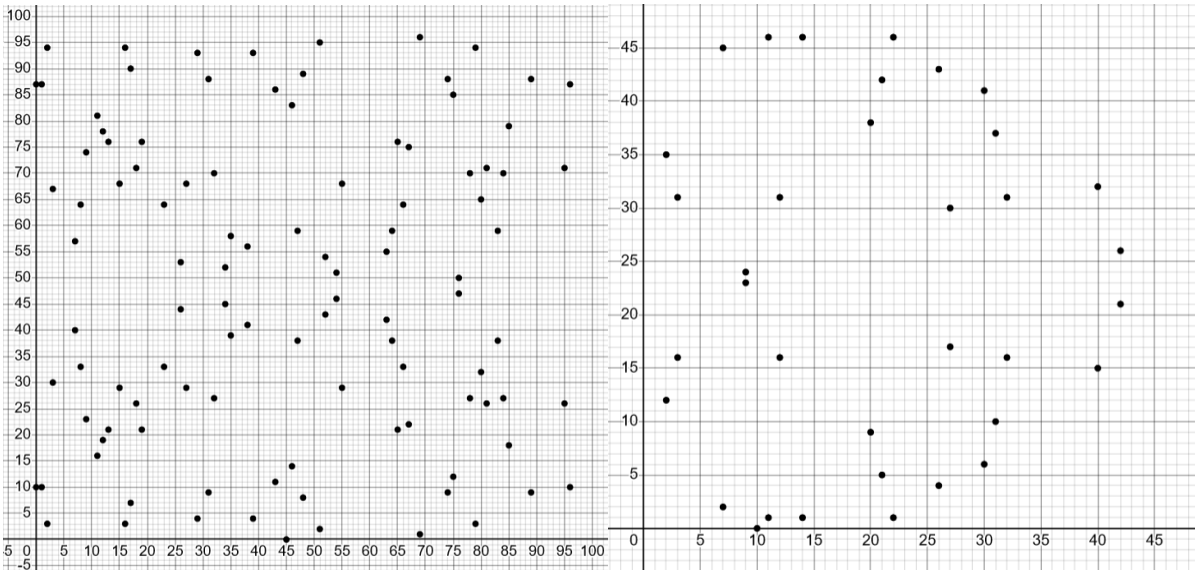
Elliptic curve is considered to form a group if it's cubic polynomial, has no repeated roots.

Examples over R:



(a) $y^2 = x^3 + 4$ (b) $y^2 = x^3 - 2x + 3$ (c) $y^2 = x^3 - 5x + 2$

Figure 1: Elliptic curves (over R)



(a) $y^2 = x^3 - x + 3 \mod 97$ (b) $y^2 = x^3 - x - 3 \mod 47$

Figure 2: Elliptic curves (over finite fields)

Operations on Elliptic Curves

Now that we know what an elliptic curve is, let's define the operational rules for performing point calculations on these curves.

$E := \{(x, y) \in K^2 \mid y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$ with $a, b \in K$

$P_i = (x_{P_i}, y_{P_i})$ with $P_i \in E$

Negating a point:

$$P = (x_P, y_P)$$

$$-P = -(x_P, y_P) = (x_P, -y_P)$$

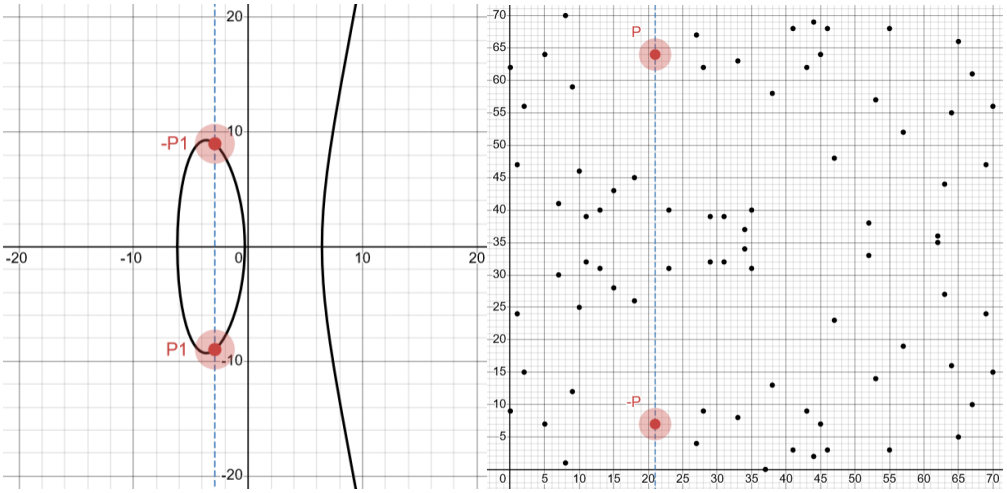


Figure 3: Visual representation of point negation

Finding inverse of a point:

$$P = (x_P, y_P)$$

$$P^{-1} = -P$$

Addition:

Addition is commutative meaning $P_i + P_j = P_j + P_i$.

In these examples below we asume that $P_3 = P_1 + P_2$

1.) if $P_1 \neq P_2$ and $P_1, P_2 \neq \mathcal{O}$:

$$\lambda = \frac{y_{P_2} - y_{P_1}}{x_{P_2} - x_{P_1}}$$

$$P_3 = (\lambda^2 - x_{P_1} - x_{P_2} \ , \ \lambda(x_{P_1} - x_{P_3}) - y_{P_1})$$

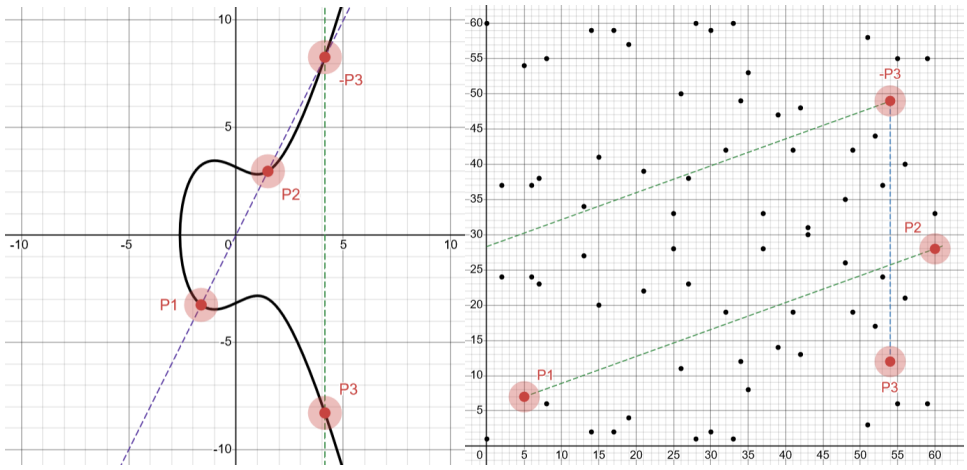


Figure 4: Visual representation of point addition (case 1)

2.) if $P_1 = P_2$ and $y_{P_1}, y_{P_2} \neq 0$ and $P_1, P_2 \neq \mathcal{O}$:

$$m = \frac{3x_{P_1}^2 + a}{2y_{P_1}}$$

$$P_3 = (m^2 - 2x_{P1} \ , \ m(x_{P_1} - x_{P_3}) - y_{P_1})$$

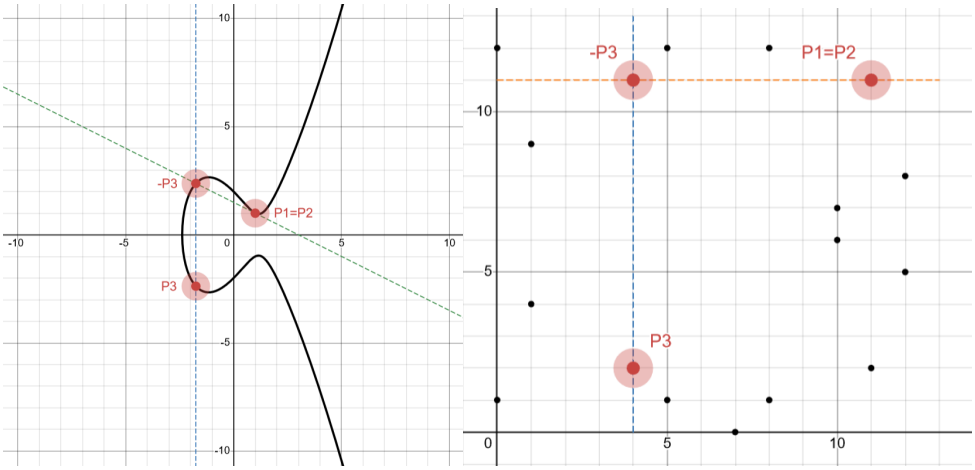


Figure 5: Visual representation of point addition (case 2)

3.) if $P_1 = P_2$ and $y_{P_1}, y_{P_2} = 0$

$$P_3 = \mathcal{O}$$

4.) if $P_1 \neq \mathcal{O}$, $P_2 = \mathcal{O}$

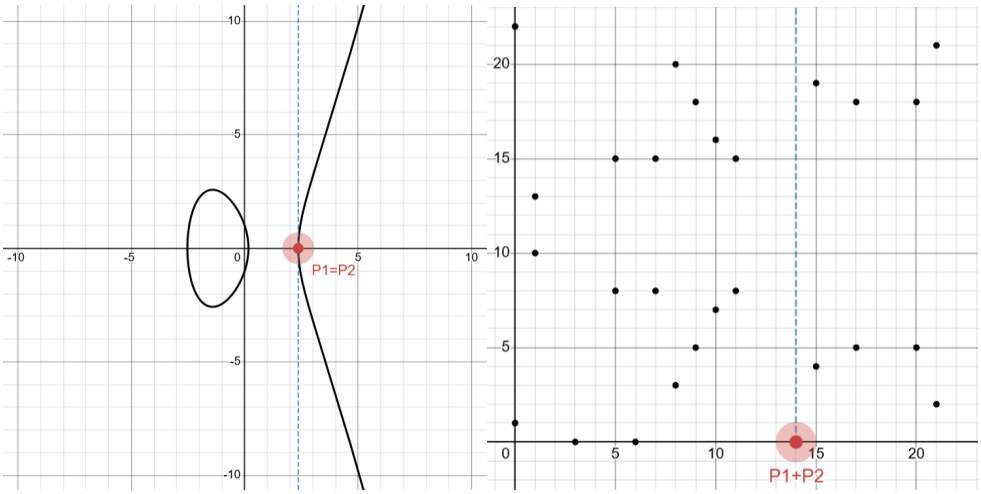


Figure 6: Visual representation of point addition (case 3)

$$P_3 = P_1$$

5.) if $P_1 = -P_2$

$$P_3 = \mathcal{O}$$

Subtraction:

$$P_1 - P_2 = P_1 + (-P_2)$$

$$(x_{P_1}, y_{P_1}) - (x_{P_2}, y_{P_2}) = (x_{P_1}, y_{P_1}) + (x_{P_2}, -y_{P_2})$$

Multiplication:

Only scalar multiplication is possible. By multiplication, we understand repeated addition of point to itself.

$$k \times P = P + P + ... \ k \text{ times}$$

Larger multiples of points:

This works well in theory but what if k was a really large number? It is obvious that in order to implement secure elliptic curve based algorithms, we will need to work with big multiples of points. The faster we can get to the result, the better. There is a number of techniques which can help us achieve faster computation of these big point multiplications.

Double and add method:

We already know that elliptic curves form a group over finite field F_P considering P is a non-even prime. This means that whenever we add any of the two members of this group together the result will also have to be a member of this group. For example:

$$3 \times P + 9 \times P = 12 \times P$$

As we saw in this example (TODO add reference to tangent point addition) adding a point P_1 to P_2 is being calculated the same way as adding P_1 to itself (assuming $P_1 = P_2$). Since $P_1 + P_1 = 2 \times P_1$, adding a point to itself is the same operation as doubling the point. Now we have effective way for doubling a point using simple addition.

We can leverage this by taking k_2 and start progressively doubling P as many times as there are binary digits from least significant bit (LSB) up to most significant "1" bit. For each "1" bit in k's binary form, we add the corresponding multiple of 2 × P to the accumulated result. Here's an example:

$$41 \times P$$

$$41_{10} = 110011_2$$

Bits of 41	**Current Doubling**	**Result (After Addition if bit = 1)**
1	P	P
1	$2 \times P$	$P + 2 \times P = 3 \times P$
0	$4 \times P$	$3 \times P$
0	$8 \times P$	$3 \times P$
1	$16 \times P$	$3 \times P + 16 \times P = 19 \times P$
1	$32 \times P$	$19 \times P + 32 \times P = 41 \times P$

Table 2: Binary Representation and Double-and-Add Method for $41 \times P$

This computation uses $log_2(n)$ multiplications and on average $\frac{1}{2}log_2(n)$ additions.

Algorithm 1 Double and add algorithm for point multiplication

1: $P_3 = \mathcal{O}$
2: $P_2 = P_1$
3: **while** $k > 0$ **do**
4: **if** $k \bmod 2 = 1$ **then**
5: $P_3 = P_3 + P_2$
6: **end if**
7: $P_2 = 2 \times P_2$
8: $k >> 1$
9: **end while**
10: **return** P_3

If we don't limit ourselves to binary representation of a number k we are able to optimize the computing time and resources even further by converting number k in to balanced ternary numeral system which uses coefficients {-1,0,1}. Subtraction has same complexity as addition since it is the same operation with a twist of negating y coordinate.

Elliptic Curves over finite field

Order of EC

Order of EC over finite field is equal to number of points on this curve + point at infinity $\mathcal{O}$. We mark order of EC to be n.

Generator point of EC subgroups

Generator point G_S of a subgroup S is a point on EC, which can generate any other point in S. We can generate all elements of S by multiplying G_S by all elements of $\langle 0, r-1 \rangle$ where r is the order of S. In cases where S contains all points of EC, G generates all points of EC.

Subgroups formed by EC

Points on EC over finite field can form one or more finite cyclic subgroups. There are cases when EC will generate only one subgroup which is of same order as the EC itself (n). If EC forms multiple subgroups, it is given that these subgroups are all non-overlapping and cyclic. Sum of the orders of these subgroups + point at infinity $\mathcal{O}$ must be equal to order of the EC.

Cofactor of EC

Cofactor is equal to number of subgroups formed by EC. We mark cofactor of EC to be h

Point compression and decompression

When working with EC over finite field, we can observe that to every x coordinate there are at most 2 corresponding y coordinates. One of these is guaranteed to be odd and the other one to be even. We can use this information to further narrow down our requirements on storing these points in computer memory by just saving the x coordinate and the parity of y coordinate (we only need 1 bit for this). Then to decompress such point we need to calculate these two formulas and then chose the coordinate with correct parity:

$y_1 = \sqrt{x^3 + ax + b} \mod p$

$y_2 = -y_1 \mod p$

EC in key exchange mechanisms:

Discrete logarithm problem DLP:

DLP is a "one-way" problem area in mathematics which considers the following qualities of modular arithmetic combined with exponential functions:
- It relatively easy to compute $a^b \mod p$ when given a, b and p
- However, finding b when given a and p is much harder task

This asymmetry is fundemental DLP concept and it is one of the key elements which many cryptographic protocols use when relying on complexity of this problem.

Primitive root:

If g is primitive root of p, then the powers $g^1, g^2 \ldots, g^{p-1} \mod$ p, will produce all the integers congruent to numbers from 1 to p -1 in some order.

Diffie-Hellman key exchange DH:

Diffie-Hellman key exchange is a key exchange protocol that lets you generate a shared secret key over which all of your communication will be encrypted without the need to share this secret ahead of time. DH leverages the difficulty of the DLP. Imagine we have two Participants, let's call them Participant A and Participant B. We want for these participants to have safe and encrypted communication. In order to achieve this they will need some sort of symmetric encryption algorithm to be used on both sides and a shared secret key which will be used on encrypting and decrypting data they send to each other. These participants have never exchanged any information before so they don't have the ability to have shared secret key communicated upfront. This is where DH comes in. DH can be decoupled into 5 steps:

Step 1

Participants agree on large prime number p and it's primitive root g. Note that this information can be shared over insecure communication channel.

Step 2

Participants generate their private keys (for example by using random number generator) and compute their respective public keys followingly:

Secret key of participant A $= a$, Secret key of participant B $= b$

$$a = random(), b = random()$$

Public key of participant A $= A$, Public key of participant B $= B$

$$A = g^a \mod p,\ B = g^b \mod p$$

Step 3

Participants share their public keys A and B with one another. Note this can also be done over insecure communication channel.

Step 4

Now it's time to calculate secret shared key S.

Shared key for participant A $= S_A$, Shared key for participant B $= S_B$

$$S_A = B^a \mod p,\ S_B = A^b \mod p$$

$$S_A = S_B$$

Step 5

Use S for encrypting and decrypting shared content

Visual example of DH

Steps	**Participant A**	**Participant B**
1	Agree on a large prime number p and it's primitive root g	
2	Generates private key a	Generates private key b
3	Calculates $A = g^a \mod p$ and sends A to participant B	Calculates $B = g^b \mod p$ and sends B to participant A
4	Receives B and calculates $S_A = B^a \mod p$	Receives A and calculates $S_B = A^b \mod p$
5	Use S_A and S_B for encrypting and decrypting data	

Table 3: Diffie-Hellman Key Exchange Process

How does DH work?

Modulo operation is distributive. When we use it on expanded forms of S_A and S_B we can see:

$$S_A = S_B$$

$$(g^b \mod p)^a \mod p = (g^a \mod p)^b \mod p$$

$$g^{ba} \mod p = g^{ab} \mod p$$

Elliptic Curve Discrete Logarithm Problem ECDLP:

ECDLP is a transformation of a classic DLP problem taken from a perspective of EC. When talking about ECDLP we take E to be an elliptic curve over finite field with order n. Let P and Q to be points on E. ECDLP leverages these facts:

- It is relatively easy to compute Q as a result of $c \times P$ when given c and P
- However, finding c $(0 \leq c \leq n-1)$ when given only P and Q is much harder task

Elliptic Curve Diffie-Hellman key exchange ECDH:

ECDH is a transofmation of DH key exchange algorithm, which uses EC point multiplication instead of modular exponentiations. When transforming DH into ECDH we take G as a generator point of sufficiently large cyclic sub-group of all points generated by E. This G will then first be multiplied by private keys of participants A and B (generation of their public keys) and then after they share the public keys with one another, those public keys will be multiplied once again with their own private key.

Visual example of ECDH

Steps	**Participant A**	**Participant B**
1	Agree on definition of curve E and generator point G	
2	Generates private key a	Generates private key b
3	Calculates $A = a \times G$ and sends A to participant B	Calculates $B = b \times G$ and sends B to participant A
4	Receives B and calculates $S_A = a \times B$	Receives A and calculates $S_B = b \times A$
5	Use S_A and S_B for encrypting and decrypting data	

Table 4: Elliptic Curve Diffie-Hellman Key Exchange Process

Attacks on EC

Chinese remainder theorem

Suppose we have numbers $m_1, m_2, ..., m_n$ that are pairwise coprime. Then the system of n equations:

$$x = a_1 \mod m_1$$

$$x = a_2 \mod m_2$$

$$...$$

$$x = a_n \mod m_n$$

$$M = m_1 * m_2 * ... * m_n$$

has a unique solution for $x \mod M$

$$M_i = \frac{M}{m_i}$$

$$x = \sum_{i=1}^{n} a_i * M_i * M_i^{-1} \mod M$$

Not verifying that a point is on EC

Let's say that we are communicating with someone and we were really careful about which specific curve definition we should use, meaning our agreed curve E and it's respective generator point P is part of a subgroup with high enough order, that anybody who is trying to brute-force their way over our whole possible private key space doesn't stand a chance. If we take a better look at formula of addition on EC, we can notice that for such computations, we only need the value of a and p, but never b (taken from formula of general EC definition). Now let's consider a scenario where if received generator point P or for example public key of other participant, we would not bother to check if this P actually exists on our EC E. This means that theoretically someone could use this to their advantage and send us point P which is on his malicious curve E' instead of our E and forms subgroup with a lot smaller order than our carefully picked curve E. Since we don't check if P is part of E, we happily compute the multiplication process for P on E' and then share the result point with our attacker. How can attacker create such malicious curve? Answer is simple. He just needs to modify b coefficient until he finds such curve E' which forms subgroup of his desired order. Now let's say that he will generate as many malicious $E's$ and Ps as he needs to in order to reveal how many operations does it take

to get to the point $a \times P \mod r$ where r is order of his malicious subgroup and a is our private key. When attacker gets to a point where he has generated as many $E's$ and Ps as he needed in order to reveal our true a he can simply just use Chinese remainder theorem to tell him.