

**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**

**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 (napísať po slovensky súhrn práce)

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

Figure 1	Elliptic curves (over R)	11
Figure 2	Elliptic curves (over finite fields)	11
Figure 3	Visual representation of point negation	12
Figure 4	Visual representation of point addition (case 1)	13
Figure 5	Visual representation of point addition (case 2)	13
Figure 6	Visual representation of point addition (case 3)	14
Figure 7	Intersection of two cubic curves	16
Figure 8	Visual representation of lines E_1, E_2, E_3 and V_1, V_2, V_3	17
Table 1	Time to Break vs Key Sizes	9
Table 2	Binary Representation and Double-and-Add Method for $41 \times P$.	15
Table 3	Diffie-Hellman Key Exchange Process	22
Table 4	Elliptic Curve Diffie-Hellman Key Exchange Process	23

List of Algorithm

1	Double and add algorithm for point multiplication	16
---	---	----

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.

Key Terms and Definitions

Algebraic curve:

An *algebraic curve* (over field K) is set of points (x, y) in the plane that satisfy a non-constant polynomial equation in two variables. A *non-singular algebraic curve* is algebraic curve without any singular points. *Singular point* is a point at which curve derivative is equal to 0, therefore having not well-defined tangent line.

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

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. This point is also inverse point of itself and has no affine (x, y) point representation.

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

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

$$-\mathcal{O} = \mathcal{O}$$

Elliptic curve:

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

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

As stated previously, our requirement on elliptic curves is that they have to be non-singular. Cubic polynomial which defines such curves can't have repeated roots meaning $\Delta = 4a^3 + 27b^2 \neq 0$ where $a, b \in K$

Examples over R :

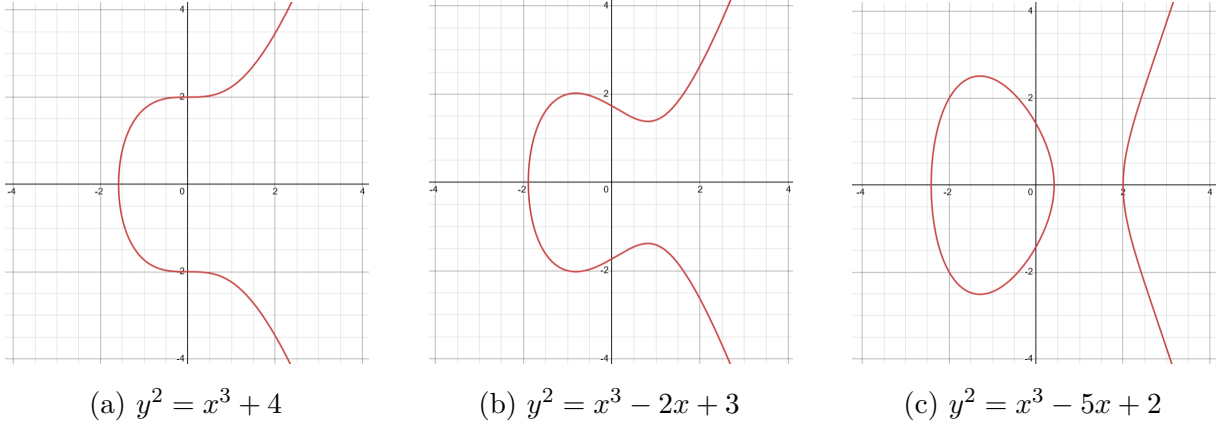


Figure 1: Elliptic curves (over R)

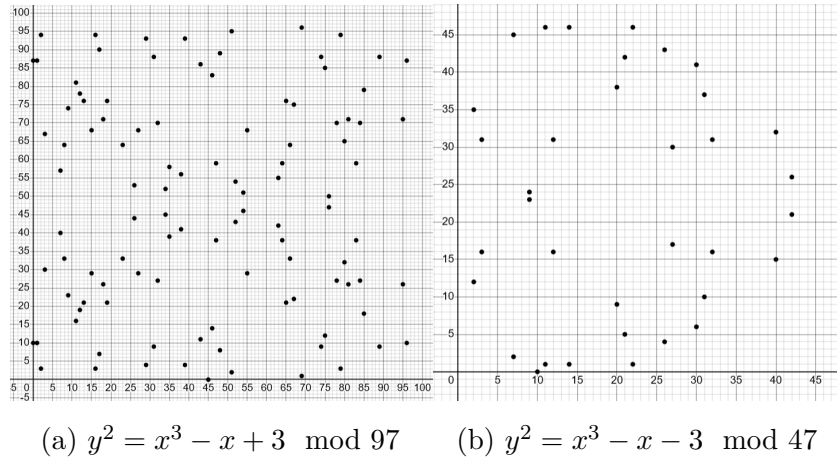


Figure 2: Elliptic curves (over finite fields)

EC as Abelian Group

We will require E with some binary operation $+$ to behave as additive Abelian group. Because we are dealing with additive group, we will be marking inverses of out points with sign $-$. For Abelian group behaviour we are going to need:

1. Closure: $\forall P_i, P_j \in E, P_i + P_j \in E$
2. Associativity: $\forall P_i, P_j, P_k \in E, (P_i + P_j) + P_k = P_i + (P_j + P_k)$
3. Identity element: $\forall P_i \in E, \exists \mathcal{O} : P_i + \mathcal{O} = P_i$
3. Identity element: $\exists \mathcal{O} : \mathcal{O} \in E, \forall P_i \in E, P_i + \mathcal{O} = P_i$
4. Inverse element: $\forall P_i \in E, \exists (-P_i) : (-P_i) \in E, P + (-P) = \mathcal{O}$
5. Commutativity: $\forall P_i, P_j \in E, P_i + P_j = P_j + P_i$

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.

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

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

Negating a point:

$$P = (x_P, y_P)$$

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

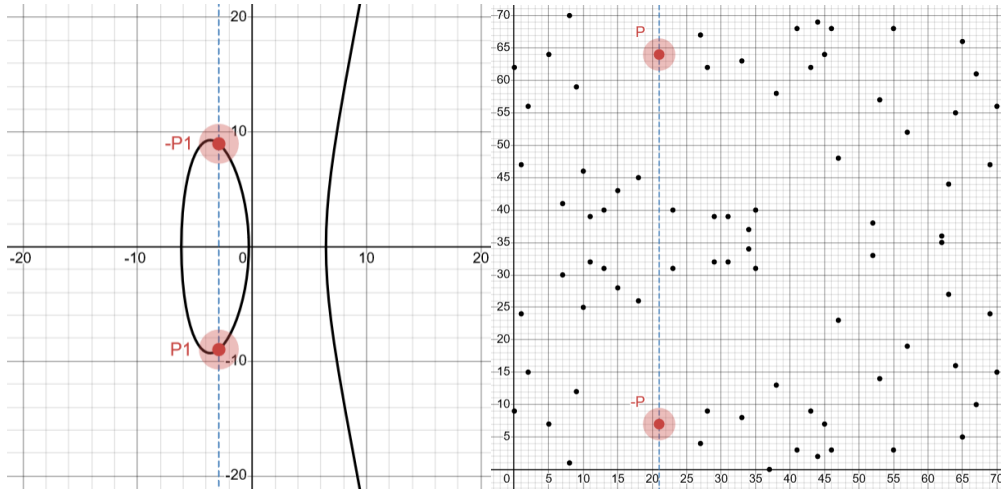


Figure 3: Visual representation of point negation

Addition:

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

$$\lambda = \frac{y_{P_2} - y_{P_1}}{x_{P_2} - x_{P_1}} = (y_{P_2} - y_{P_1})(x_{P_2} - x_{P_1})^{-1}$$

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

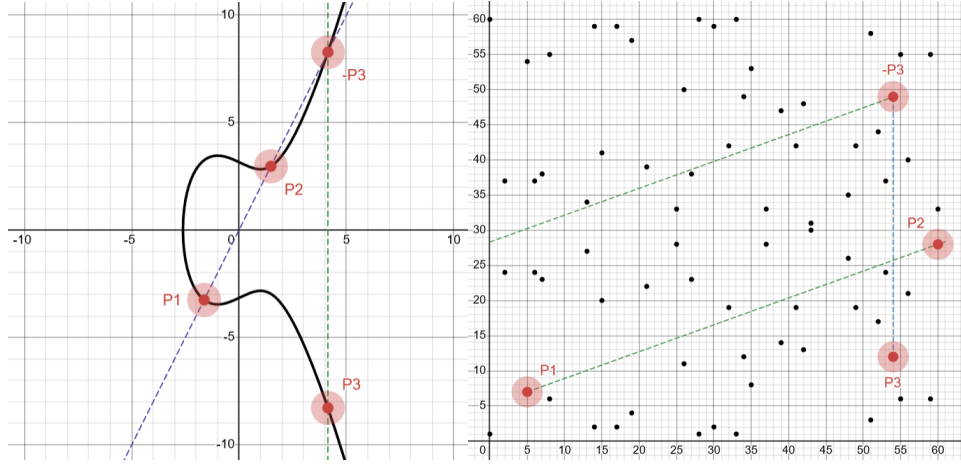


Figure 4: Visual representation of point addition (case 1)

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

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

$$n = m^2 - 2x_{P_1}$$

$$P_1 + P_2 = (n, m(x_{P_1} - n) - y_{P_1}) = P_3$$

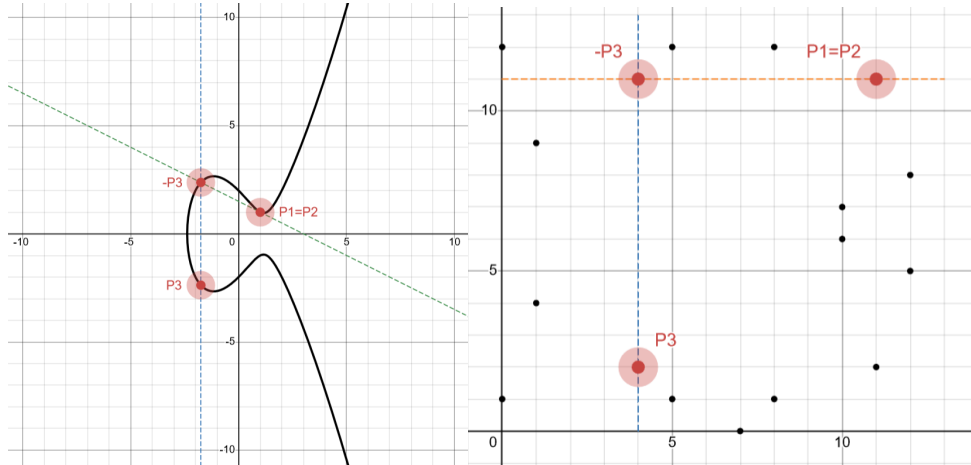


Figure 5: Visual representation of point addition (case 2)

3.) if $P_1 = P_2 \wedge y_{P_1}, y_{P_2} = 0 \vee P_1 = -P_2$

$$P_1 + P_2 = \mathcal{O} = P_3$$

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

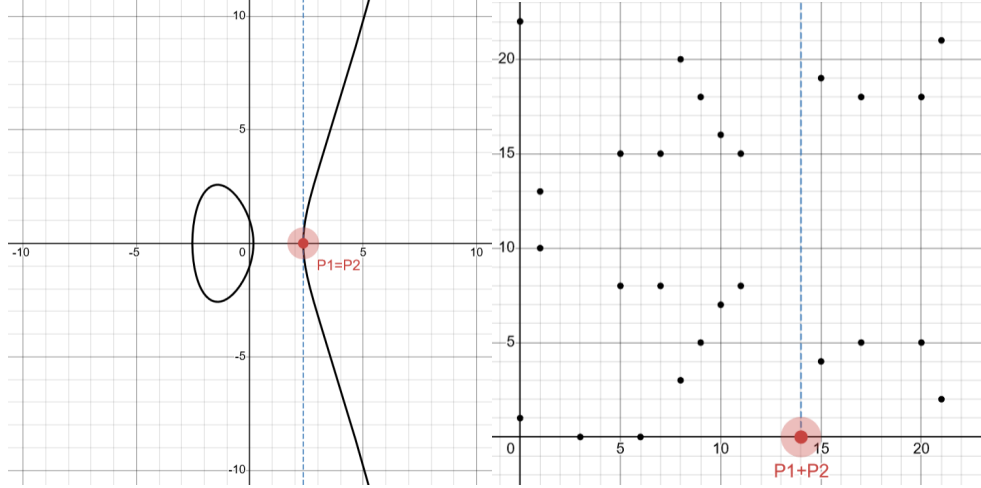


Figure 6: Visual representation of point addition (case 3)

$$P_1 + P_2 = P_1 = P_3$$

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.

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

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:

Group formed by EC is closed under $+$ operation. We can also notice that:

$$P + P = 2 \times P$$

$$2 \times P + 2 \times P = 4 \times P$$

...

In summary, adding a point to itself is the same operation as doubling the point and with our group having closure property, we are guaranteed to stay on EC even when doubling a point repeatedly. Now we have effective way for doubling P using simple addition.

When calculating $k \times P$ we progressively double 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 $2 \times P$ to the accumulated result. Here's an example using $k = 41$:

$$41 \times P$$

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

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 $\log_2(n)$ point multiplications and on average $\frac{1}{2}\log_2(n)$ point 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:    $k \gg 1$ 
9: end while
10: return  $P_3$ 
```

We can minimize number of point additions by converting number k into balanced ternary numeral system with coefficients $\{-1, 0, 1\}$.

Bézout's theorem for curves

If two algebraic curves, defined by polynomials, intersect in the projective plane, the number of intersection points (counted with multiplicities + points at infinity) is the product of the degrees of the two curves. For example two cubic curves C_1, C_2 intersect at exactly $\deg(C_1) * \deg(C_2) = 3 * 3 = 9$ total points of intersection.

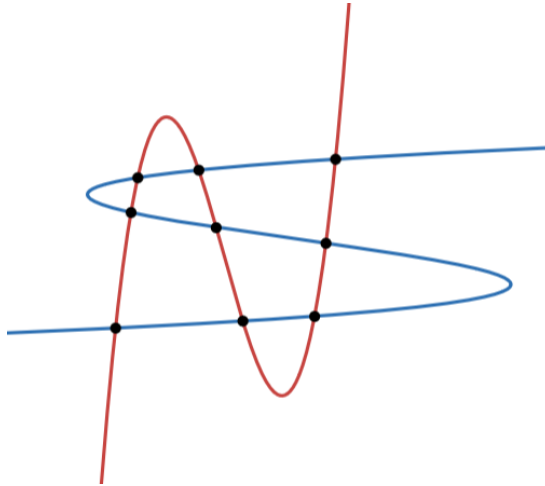


Figure 7: Intersection of two cubic curves

Cayley-Bacharach theorem

If $P_1 \dots P_8$ are distinct points in projective plane, then there is a 9th point P_9 such that any cubic through $P_1 \dots P_8$ also passes through P_9 , if in $P_1 \dots P_8$, there are no 4 points on a line, and no 7 on a conic

Associativity of EC groups

Our main goal here is to show that:

$$\forall A, B, C \in E, (A + B) + C = A + (B + C)$$

We will prove same statement negated on both sides instead

$$\forall A, B, C \in E, -((A + B) + C) = -(A + (B + C))$$

Let's define 6 lines, where each defined by 3 points intersecting with EC like so:

$$E_1 = \text{line given by points: } \{A, B, -(A + B)\}$$

$$E_2 = \text{line given by points: } \{A + B, C, -((A + B) + C)\}$$

$$E_3 = \text{line given by points: } \{B + C, O, -(B + C)\}$$

$$V_1 = \text{line given by points: } \{A + B, O, -(A + B)\}$$

$$V_2 = \text{line given by points: } \{B, C, -(B + C)\}$$

$$V_3 = \text{line given by points: } \{A, B + C, -(A + (B + C))\}$$

Lines defined above are represented visually in the following picture:

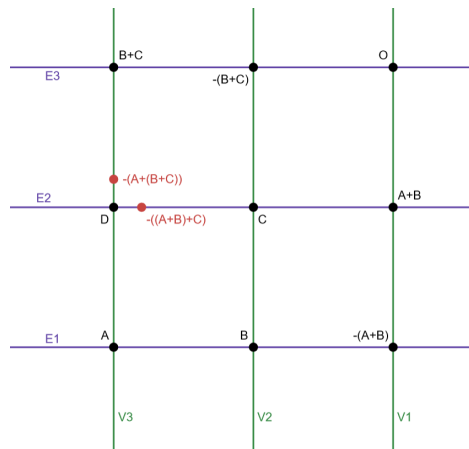


Figure 8: Visual representation of lines E_1, E_2, E_3 and V_1, V_2, V_3

Notice we labeled new point D where lines E_2 and V_1 intersect. We don't know for sure if E passes through D . For all of the other points displayed in the picture above, we know for sure that E passes through them. We know that $-(A + (B + C)) \in V_3$ and $-((A + B) + C) \in E_2$. We need $-((A + B) + C) = -(A + (B + C)) = D$. Let's consider these two cubic curves:

$$C_1 : E_1 E_2 E_3 = 0$$

$$C_2 : V_1 V_2 V_3 = 0$$

We know that C_1 and C_2 share these 8 points

$$\{A, B, -(A + B), C, A + B, B + C, -(B + C), O\}$$

Cayley-Bacharach theorem states, that there must be 9th point in which they intersect. Bézout's theorem states that this point is the only point left in which C_1 and C_2 will intersect. Since V_3 is a factor of $C_2, D \in C_2$. We can now see what points lie on $C_2 \cap E$. It's points:

$$\{A, B, -(A + B), C, A + B, B + C, -(B + C), O, -(A + (B + C)), D\}$$

Above set of points has 10 elements, which contradicts with Bézout's theorem, meaning two of those points must be equal. By definition we know that the starting set of 8 points are all distinct, so the only duplicate point we have in this set is $D = -(A + (B + C))$. Similarly for E_2 being a factor of $C_1, D \in C_1$.

$C_1 \cap E$:

$$\{A, B, -(A + B), C, A + B, B + C, -(B + C), O, -((A + B) + C), D\}$$

The duplicate point we have in this set is $D = -((A + B) + C)$. Not only we have now proof that $D \in E$ but also that $D = -(A + (B + C)) = -((A + B) + C)$. With this we also covered the associativity requirement for EC to behave as additive Abelian group.

Projective coordinates

Affine coordinates are commonly used to represent points on EC mostly because they are easy to work with and understand, but there is one problem of using such coordinates, especially when working over algebraically closed fields. This problem is that for every point addition we want to perform, we need to calculate $(x_{P_1} + x_{P_2})^{-1}$ using *extended Euclidean algorithm* which is costly operation. To avoid this we can represent points of EC with projective coordinates. To convert point from affine (x, y) to projective coordinates (X, Y, Z) , we take non-zero value of Z and convert:

$$(x, y) \equiv (xZ, yZ, Z)$$

Example:

$$(1, 2) \equiv (3, 6, 3) = (4, 8, 4) = (5, 10, 5) \dots$$

To convert projective coordinates to affine coordinates:

$$(x, y) = \left(\frac{X}{Z}, \frac{Y}{Z}\right)$$

Definition of EC achieved by converting affine coordinates to projective ones:

$$E := \{(X : Y : Z) \in P^2(K) \mid Y^2Z = X^3 + aXZ^2 + bZ^3\}$$

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 a finite field can form one or more finite cyclic subgroups. There are cases where the EC will generate only one subgroup, which is of the same order as the EC itself (n). If EC forms multiple subgroups, it is given that these subgroups are all nonoverlapping 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 fundamental 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 $g^1, g^2, \dots, g^{p-1} \mod p$, 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 its 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

$$a = \text{random}(), b = \text{random}()$$

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:

$$\begin{aligned} S_A &= S_B \\ (g^b \bmod p)^a \bmod p &= (g^a \bmod p)^b \bmod p \\ g^{ba} \bmod p &= g^{ab} \bmod p \end{aligned}$$

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 transformation 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, \dots, m_n that are pairwise coprime. Then the system of n equations:

$$x = a_1 \pmod{m_1}$$

$$x = a_2 \pmod{m_2}$$

...

$$x = a_n \pmod{m_n}$$

$$M = m_1 * m_2 * \dots * m_n$$

has a unique solution for $x \pmod{M}$

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

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

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 P s as he needs to in order to reveal how many operations does it take

to get to the point $a \times P \bmod r$ 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 P s as he needed in order to reveal our true a he can simply just use Chinese remainder theorem to tell him.