SRI JAYACHAMARAJENDRA COLLEGE OF ENGINEERING

A Constituent College of

JSS SCIENCE & TECHNOLOGY UNIVERSITY

Additive Cipher , Multiplicative Cipher & Elliptic Curve Cryptography

Mini project report submitted in partial fulfillment of curriculum prescribed for the : Cryptography and Network Security (20CS552) course for the award of the degree of

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING

PROJECT REPORT

Submitted by

S.No.	NAME	ROLLNO.	USN
1	Adithya Deepthi Kumar	28	01JST21CS005
2	E Shreyas Herale	35	01JST21CS037
3	Eshwar J	8	01JCE21CS033
4	Harsha N P	10	01JCE21CS038

D-SECTION

Under the Guidance of

Shwethashree G C

Assistant Professor

Dept.of CS & E,

SJCE, JSSSTU, Mysuru

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

November 2023

SRI JAYACHAMARAJENDRA COLLEGE OF ENGINEERING
A Constituent College of
JSS SCIENCE & TECHNOLOGY UNIVERSITY

CERTIFICATE

This is to certify that the work entitled“ **Additive Cipher, Multiplicative Cipher,Elliptic Curve Cryptography ”** is a bonafied work carried out by **Adithya Deepthi Kumar, E Shreyas Herale, Eshwar J, Harsha N P** in partial fulfillment of the award of the degree of **Bachelor of Engineering in Computer Science and Engineering of JSS Science and Technology University, Mysuru during the year 2023**. It is certified that all corrections / suggestions indicated during CIE have been incorporated in the report. The mini project report has been approved as it satisfies the academic requirements in respect of mini project work for event 1 prescribed for the Cryptography and Network Security (20CS552) course.

Course in Charge and Guide

Shwethashree G C
Assistant Professor
Dept.of CS & E,
SJCE, JSSSTU, Mysuru

Place: Mysuru **Date : 28/11/2023**

ABSTRACT

This report offers a comprehensive comparative analysis of the Additive Cipher, Multiplicative Cipher, and Elliptic Curve Cryptography. The classical Additive and Multiplicative Ciphers, rooted in modular arithmetic, are examined for historical significance, algorithmic simplicity, and susceptibility to attacks. In contrast, Elliptic Curve Cryptography (ECC) present a modern and secure encryption approach, leveraging the mathematical foundation of elliptic curves. The report emphasizes ECC's security advantages, computational efficiency, and resistance to specific attacks. Comparative strengths and weaknesses are discussed, highlighting trade-offs in computational complexity, key management, and resistance to cryptographic attacks. The findings aim to guide practitioners, researchers, and decision-makers in making informed choices for diverse security contexts.security.

TABLE OF CONTENTS

S.No.	Content	Page No.
1.	Introduction	1
1.1	Intoduction to Cryptography	1
1.2	Cipher	2
1.3	Historical Significance	3
1.4	Symmetric key Cipher	5
2	Additive Cipher	6
2.1	Additive Cipher Implementation	7
2.2	Advantages and Disadvantages of Additive Cipher	10
2.3	Applications of Additive Cipher	11
3	Multiplicative Cipher	12
3.1	Multiplicative Cipher Implementation	13
3.2	Advantages and Disadvantages of Multiplicative Cipher	16
3.3	Applications of Multiplicative Cipher	17
4	Elliptic-curve cryptography Algorithm	18
4.1	Elliptic-curve cryptography Implementation	23
4.2	Pros, Cons and Application of ECC	28
5	Conclusion	31
6	References	31

Elliptic Curve Cryptography

Elliptic-curve cryptography (**ECC**) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. ECC allows smaller keys compared to non-EC cryptography to provide equivalent security.

Elliptic curves are applicable for key agreement, digital signatures, pseudo-random generators and other tasks. Indirectly, they can be used for encryption by combining the key agreement with a symmetric encryption scheme. They are also used in several integer factorization algorithms that have applications in cryptography, such as Lenstra elliptic-curve factorization.

Although RSA and ElGamal are secure asymmetric-key cryptosystems, their security comes with a price, their large keys. Researchers have looked for alternatives that give the same level of security with smaller key sizes. One of these promising alternatives is the elliptic curve cryptosystem (ECC). The system is based on the theory of elliptic curves.

Elliptic Curves over Real Numbers:

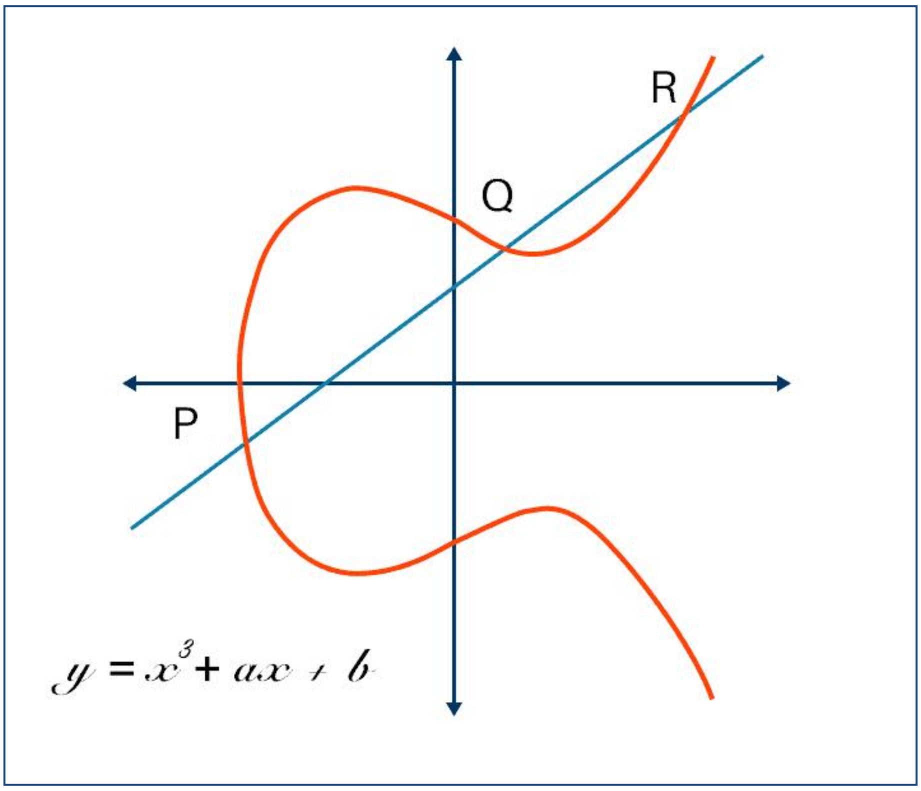
Elliptic curves, which are not directly related to ellipses, are cubic equations in two variables that are similar to the equations used to calculate the length of a curve in the circumference of an ellipse.

The general equation for an elliptic curve is

$$y^2 + b_1xy + b_2y = x^3 + a_1x^2 + a_2x + a_3$$

Elliptic curves over real numbers use a special class of elliptic curves of the form

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



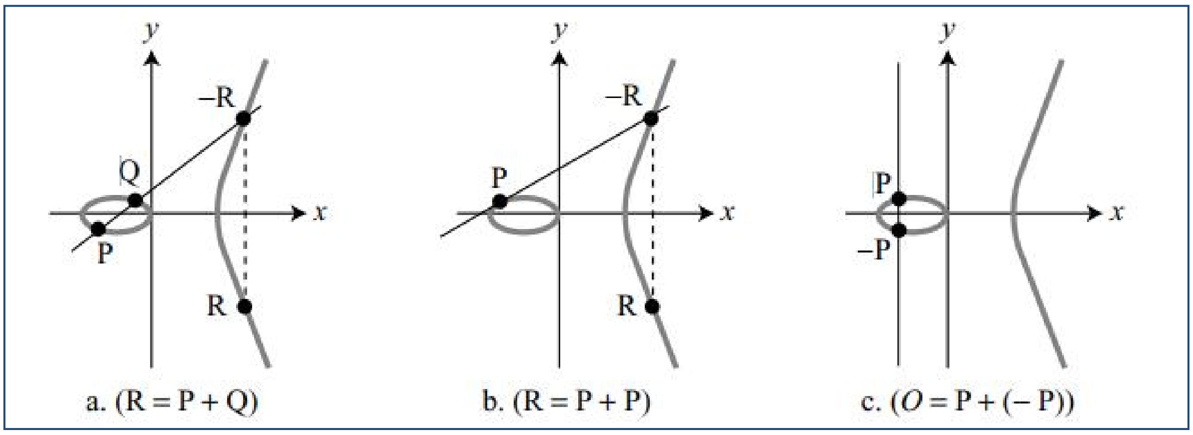
Addition operation in ECC:

The operation is the addition of two points on the curve to get another point on the curve

$$R = P + Q, \text{ where } P = (x_1, y_1),\ Q = (x_2, y_2), \text{ and } R = (x_3, y_3)$$

Three adding cases in an elliptic curve:

\



- In the first case, the two points P = (x1, y1) and Q = (x2, y2) have different x-coordinates and y-coordinates (x1 ≠ y1 and x2 ≠ y2), as shown in Figure(a). The line connecting P and Q intercepts the curve at a point called −R. R is the reflection of −R with respect to the x-axis. The coordinates of the point R, x3 and y3, can be found by first finding the slope of the line, λ, and then calculating the values of x3 and y3, as shown below:

$$\lambda = (y2 - y1) / (x2 - x1)$$

$$x3 = \lambda 2 - x1 - x2 \qquad y3 = \lambda\ (x1 - x3) - y1$$

- In the second case, the two points overlap (R = P + P), as shown in Figure (b). In this case, the slope of the line and the coordinates of the point R can be found as shown below:

$$\lambda = (3x1\ 2 + a)/(2y1)$$

$$x3 = \lambda 2 - x1 - x2 \qquad y3 = \lambda\ (x1 - x3) - y1$$

- In the third case, the two points are additive inverses of each other as shown in Figure(c). If the first point is P = (x1, y1), the second point is Q = (x1, −y1). The line connecting the two points does not intercept the curve at a third point. Mathematicians say that the intercepting point is at infinity; they define a point O as the point at infinity or zero point, which is the additive identity of the group.

Elliptic Curves over GF(p)

Steps for Elliptic Curve Operations over GF(p):

1.Elliptic Curve Definition:
Define an elliptic curve group over the finite field GF(p) as Ep(a, b), where ‘p’ is the modulus, and ‘a’ and ‘b’ are coefficients in the equation $y^2 = x^3 + ax + b$.

2. Inverse Calculation:
Determine the inverse of a point (x, y) as (x, -y), where -y is the additive inverse of y. For example, if p = 13, the inverse of (4, 2) is (4, 11).

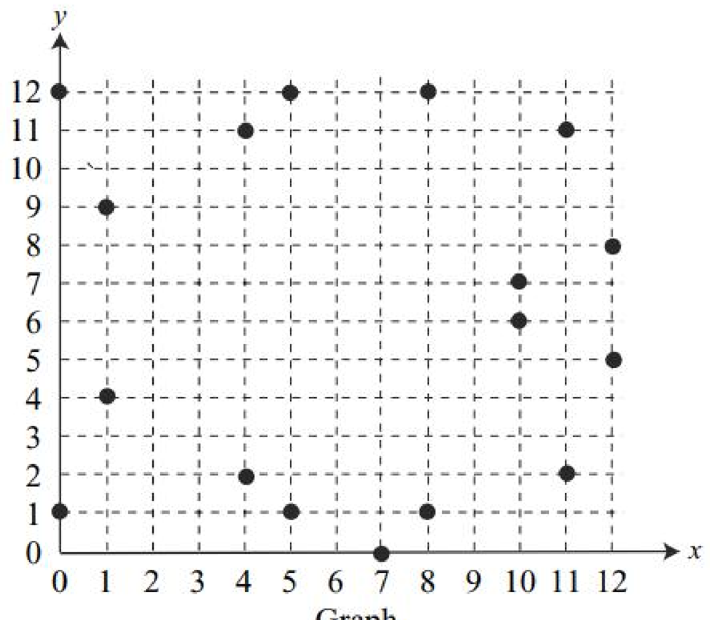
3. Finding Points on the Curve:
Algorithm shows the pseudocode for finding the points on the curve Ep(a,b)

```
ellipticCurve_points ($p, a, b$)                          // $p$ is the modulus
{
    $x \leftarrow 0$
    while ($x < p$)
    {
        $w \leftarrow (x^3 + ax + b) \bmod p$                    // $w$ is $y^2$
        if ($w$ is a perfect square in $\mathbf{Z}_p$) output ($x, \sqrt{w}$) ($x, -\sqrt{w}$)
        $x \leftarrow x + 1$
    {
}
```

An elliptic curve $E_{13}(1, 1)$. The equation is $y^2 = x^3 + x + 1$ and the calculation is done modulo 13. Points on the curve can be found as shown in Figure.

(0, 1)	(0, 12)
(1, 4)	(1, 9)
(4, 2)	(4, 11)
(5, 1)	(5, 12)
(7, 0)	(7, 0)
(8, 1)	(8, 12)
(10, 6)	(10, 7)
(11, 2)	(11, 11)
(12, 5)	(12, 8)

Points



Graph

4.Adding Two Points:
We use the elliptic curve group defined earlier, but calculations are done in GF(p). Instead of subtraction and division, we use additive and multiplicative inverses.

Let us add two points, R = P + Q, where P = (4, 2) and Q = (10, 6).

a. $\lambda = (6 - 2) \times (10 - 4)^{-1} \bmod 13 = 4 \times 6^{-1} \bmod 13 = 5 \bmod 13.$

b. $x = (5^2 - 4 - 10) \bmod 13 = 11 \bmod 13.$

c. $y = [5\,(4 - 11) - 2] \bmod 13 = 2 \bmod 13.$

d. R = (11, 2), which is a point on the curve

5. Multiplying a Point by a Constant:
In arithmetic, multiplying a number by a constant k means adding the number to itself k times. The situation here is the same. Multiplying a point P on an elliptic curve by a constant k means adding the point P to itself k times. For example, in E_{13} (1, 1), if the point (1, 4) is multiplied by 4, the result is the point (5, 1). If the point (8, 1) is multiplied by 3, the result is the point (10, 7).

These steps outline the fundamental operations in elliptic curve cryptography over the finite field GF(p), encompassing point calculations, addition, and multiplication by constants.

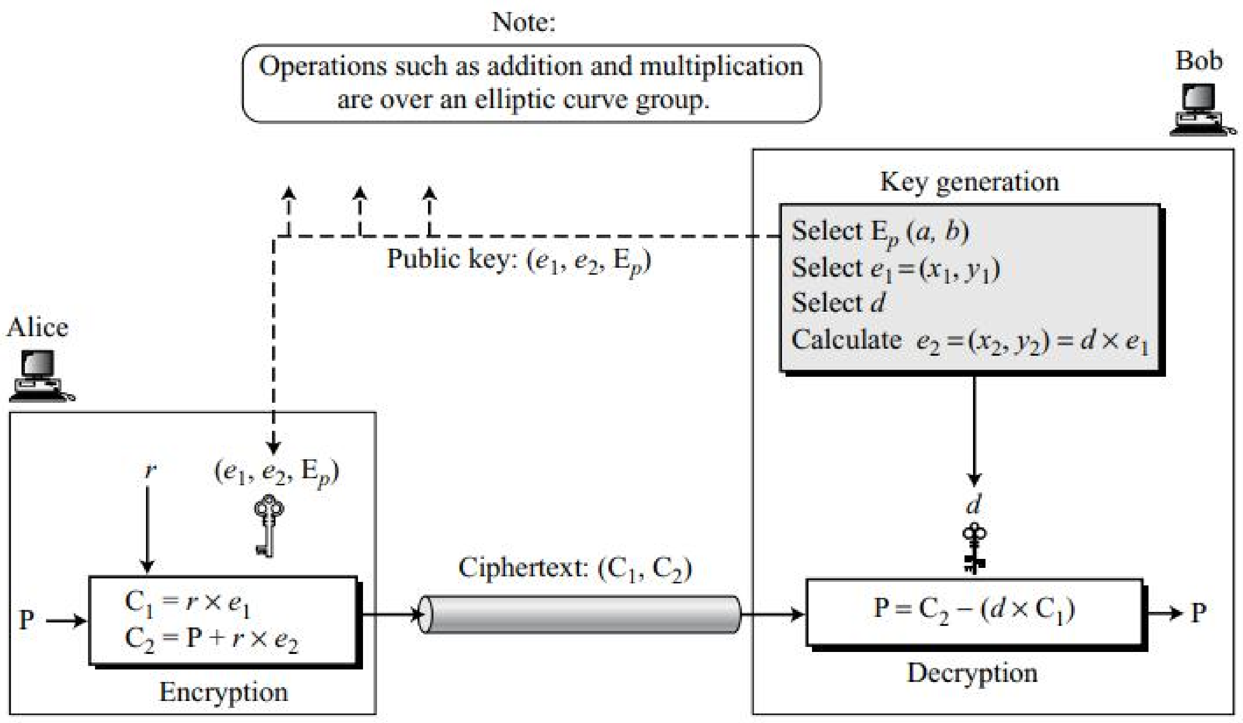
Attacks on Elliptic Curve Cryptography:

Elliptic Curve Cryptography (ECC) faces several potential attacks. **Brute force attacks** involve systematically trying all possible private key values to find the correct one. **Pollard's Rho attack** efficiently solves the elliptic curve discrete logarithm problem, particularly when the key space is small. **Side-channel attacks** exploit information leaks during cryptographic operations, while **fault injection attacks** intentionally induce errors to reveal sensitive information. **Elliptic curve factorization attacks** focus on factorizing the order of the curve to obtain the private key. **Invalid curve attacks** manipulate curve parameters or use invalid curves to exploit vulnerabilities in ECC implementations. Finally, the emergence of quantum computers poses a threat, with Shor's algorithm potentially undermining ECC security by efficiently solving the discrete logarithm problem. To counteract these threats, best practices in key generation, management, and the exploration of post-quantum cryptographic alternatives are crucial.

Elliptic Curve Cryptography Simulating ElGamal

Several methods have been used to encrypt and decrypt using elliptic curves. The common one is to simulate the ElGamal cryptosystem using an elliptic curve over GF(p)

ElGamal cryptosystem using the elliptic curve


Note:
Operations such as addition and multiplication are over an elliptic curve group.
Bob
Key generation
Select $E_p(a, b)$
Select $e_1 = (x_1, y_1)$
Select d
Calculate $e_2 = (x_2, y_2) = d \times e_1$
Public key: (e_1, e_2, E_p)
Alice
r
(e_1, e_2, E_p)
d
P
$C_1 = r \times e_1$
$C_2 = P + r \times e_2$
Encryption
Ciphertext: (C_1, C_2)
$P = C_2 - (d \times C_1)$
P
Decryption

Generating Public and Private Keys

1. Bob chooses E(a, b) with an elliptic curve over **GF**(p) or **GF**(2^n).
2. Bob chooses a point on the curve, $e_1(x_1, y_1)$.
3. Bob chooses an integer d.
4. Bob calculates $e_2(x_2, y_2) = d \times e_1(x_1, y_1)$. Note that multiplication here means multiple addition of points as defined before.
5. Bob announces E(a, b), $e_1(x_1, y_1)$, and $e_2(x_2, y_2)$ as his public key; he keeps d as his private key.

Encryption

Alice selects P, a point on the curve, as her plaintext, P. She then calculates a pair of points on the text as ciphertexts:

$$\mathbf{C_1 = r \times e_1} \qquad \mathbf{C_2 = P + r \times e_2}$$

Decryption

Bob, after receiving C_1 and C_2, calculates P, the plaintext using the following formula.

$\mathbf{P = C_2 - (d \times C_1)}$ The minus sign here means adding with the inverse.

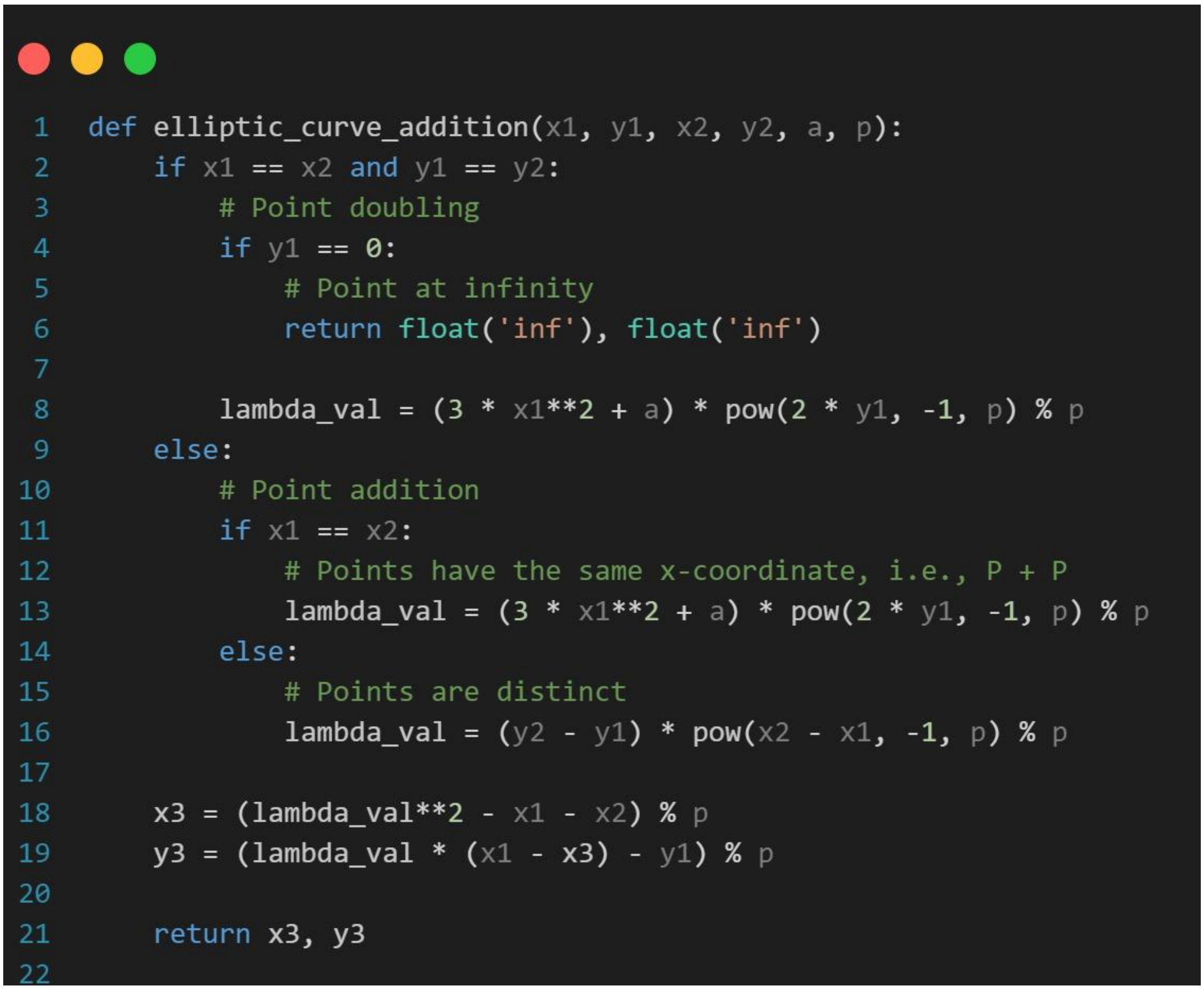
We can prove that the P calculated by Bob is the same as that intended by Alice, as shown below:

$$P + r \times e_2 - (d \times r \times e_1) \;=\; P + (r \times d \times e_1) - (r \times d \times e_1) \;=\; P + O \;=\; P$$

P, C_1, C_2, e_1, and e_2 are all points on the curve. Note that the result of adding two inverse points on the curve is the zero point.

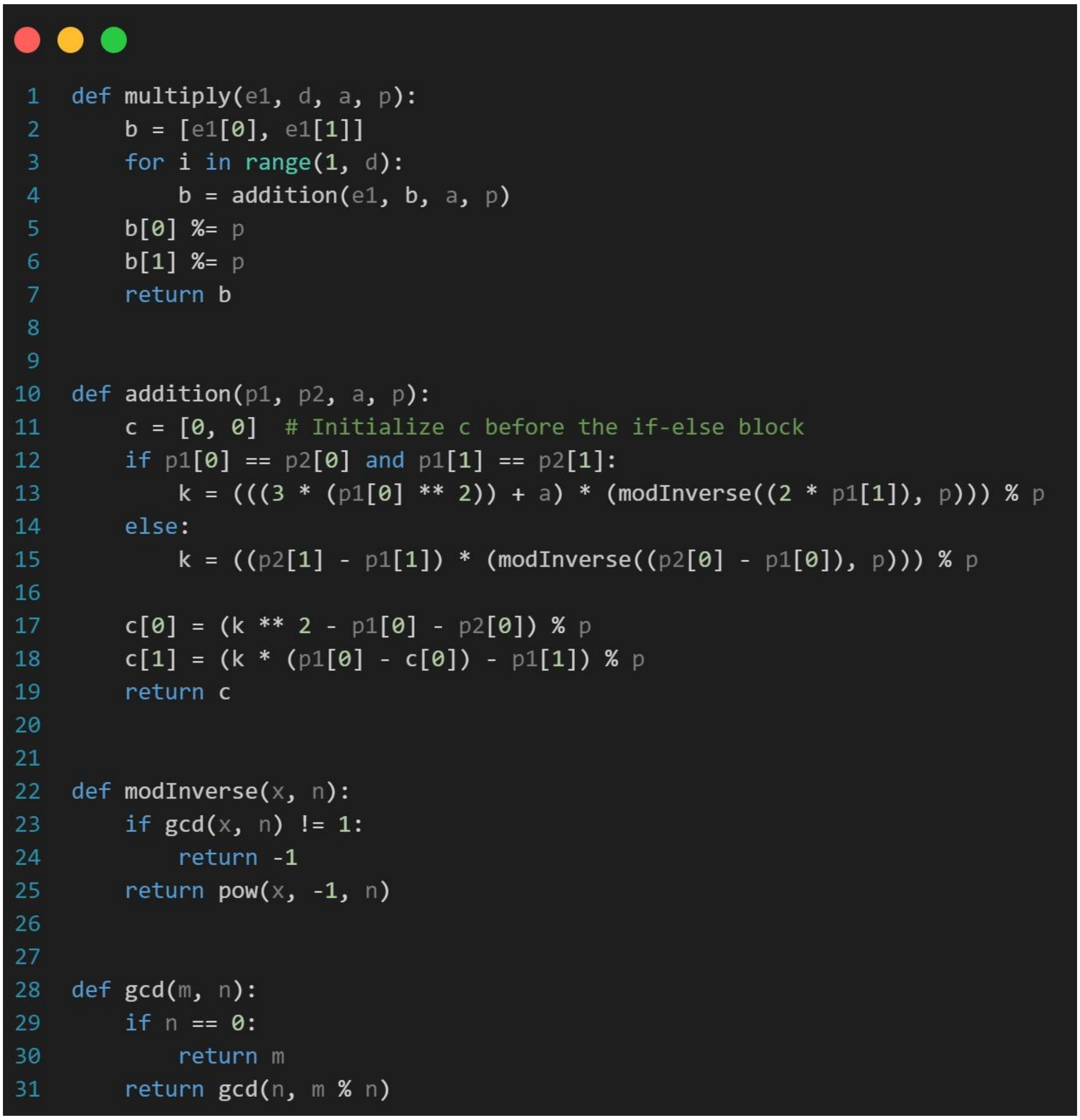
Elliptic Curve Cryptography Implementation:

Elliptical Curve Addition:



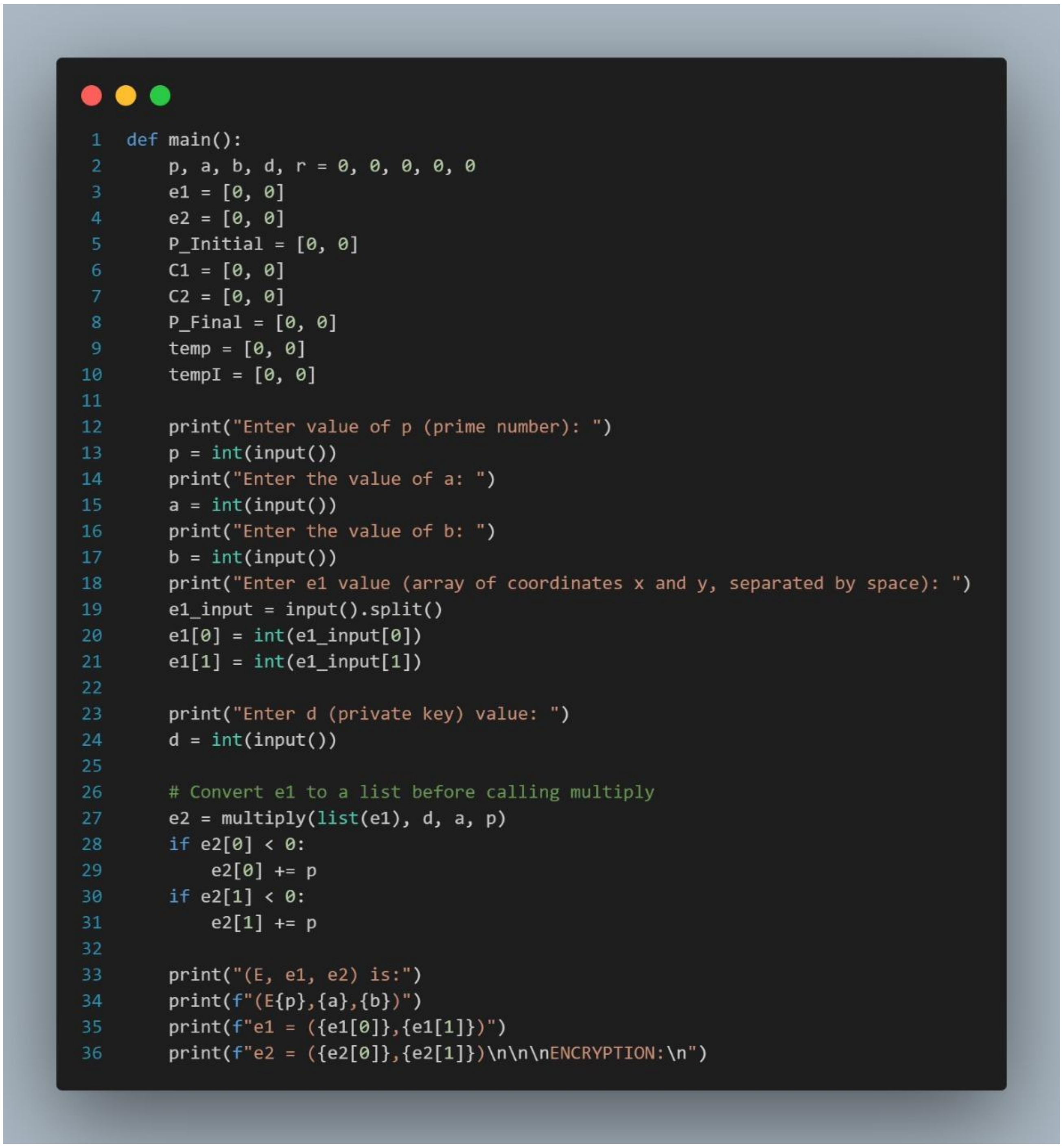
```python
def elliptic_curve_addition(x1, y1, x2, y2, a, p):
    if x1 == x2 and y1 == y2:
        # Point doubling
        if y1 == 0:
            # Point at infinity
            return float('inf'), float('inf')

        lambda_val = (3 * x1**2 + a) * pow(2 * y1, -1, p) % p
    else:
        # Point addition
        if x1 == x2:
            # Points have the same x-coordinate, i.e., P + P
            lambda_val = (3 * x1**2 + a) * pow(2 * y1, -1, p) % p
        else:
            # Points are distinct
            lambda_val = (y2 - y1) * pow(x2 - x1, -1, p) % p

    x3 = (lambda_val**2 - x1 - x2) % p
    y3 = (lambda_val * (x1 - x3) - y1) % p

    return x3, y3

```

Driver Function:

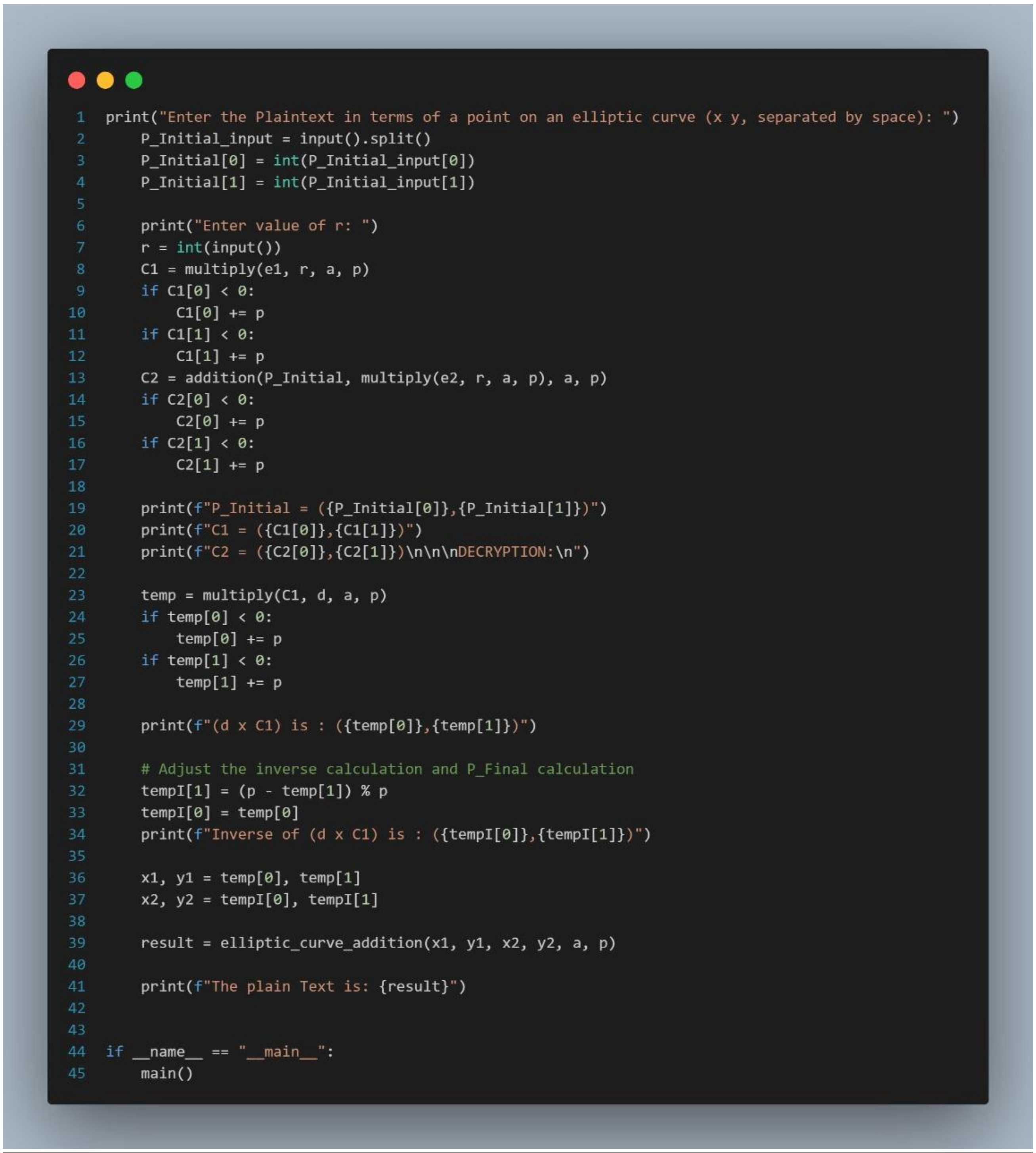


```python
def multiply(e1, d, a, p):
    b = [e1[0], e1[1]]
    for i in range(1, d):
        b = addition(e1, b, a, p)
    b[0] %= p
    b[1] %= p
    return b


def addition(p1, p2, a, p):
    c = [0, 0]  # Initialize c before the if-else block
    if p1[0] == p2[0] and p1[1] == p2[1]:
        k = (((3 * (p1[0] ** 2)) + a) * (modInverse((2 * p1[1]), p))) % p
    else:
        k = ((p2[1] - p1[1]) * (modInverse((p2[0] - p1[0]), p))) % p

    c[0] = (k ** 2 - p1[0] - p2[0]) % p
    c[1] = (k * (p1[0] - c[0]) - p1[1]) % p
    return c


def modInverse(x, n):
    if gcd(x, n) != 1:
        return -1
    return pow(x, -1, n)


def gcd(m, n):
    if n == 0:
        return m
    return gcd(n, m % n)
```

Main Function:

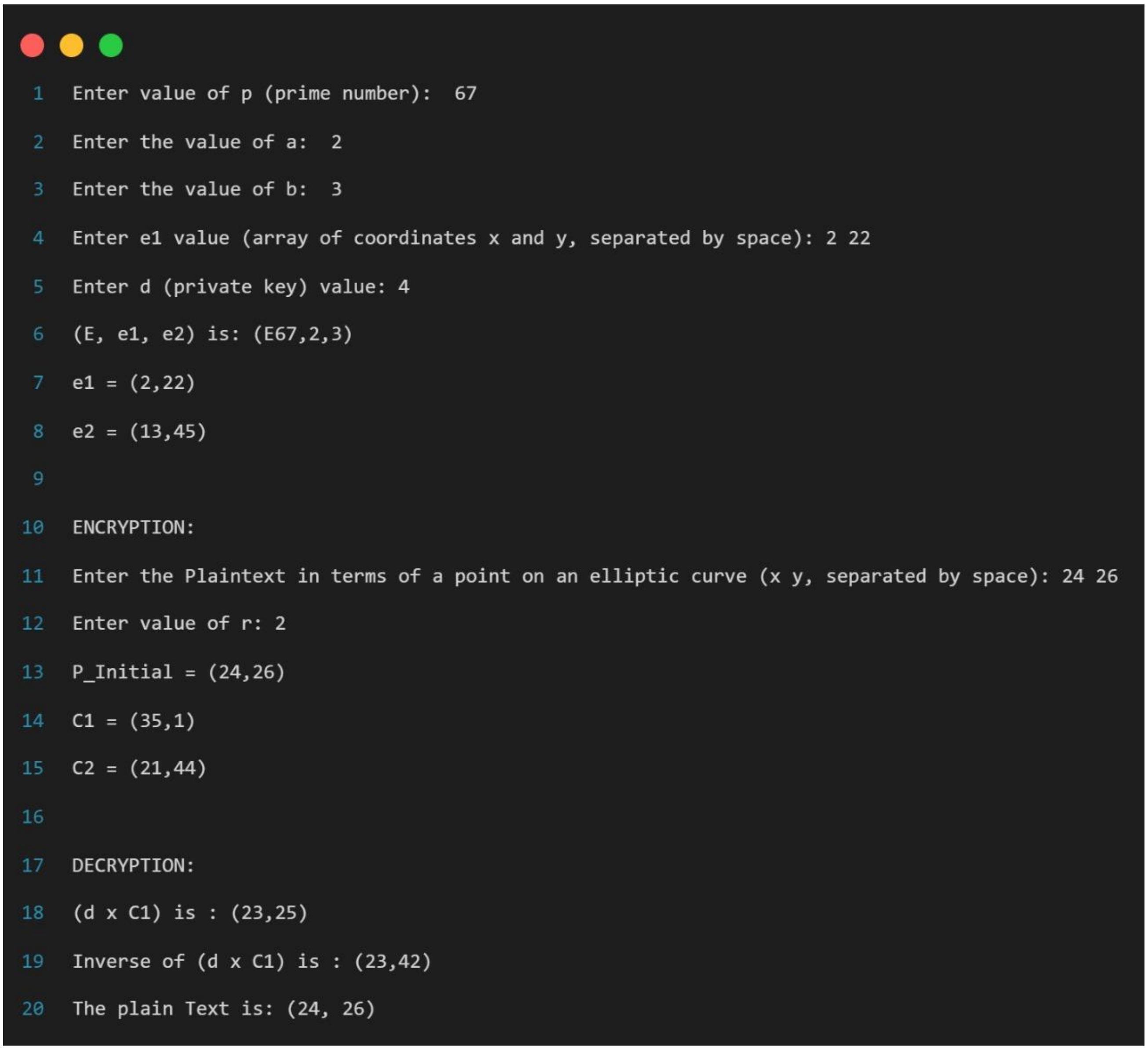


```python
def main():
    p, a, b, d, r = 0, 0, 0, 0, 0
    e1 = [0, 0]
    e2 = [0, 0]
    P_Initial = [0, 0]
    C1 = [0, 0]
    C2 = [0, 0]
    P_Final = [0, 0]
    temp = [0, 0]
    tempI = [0, 0]

    print("Enter value of p (prime number): ")
    p = int(input())
    print("Enter the value of a: ")
    a = int(input())
    print("Enter the value of b: ")
    b = int(input())
    print("Enter e1 value (array of coordinates x and y, separated by space): ")
    e1_input = input().split()
    e1[0] = int(e1_input[0])
    e1[1] = int(e1_input[1])

    print("Enter d (private key) value: ")
    d = int(input())

    # Convert e1 to a list before calling multiply
    e2 = multiply(list(e1), d, a, p)
    if e2[0] < 0:
        e2[0] += p
    if e2[1] < 0:
        e2[1] += p

    print("(E, e1, e2) is:")
    print(f"(E{p},{a},{b})")
    print(f"e1 = ({e1[0]},{e1[1]})")
    print(f"e2 = ({e2[0]},{e2[1]})\n\n\nENCRYPTION:\n")
```



```
print("Enter the Plaintext in terms of a point on an elliptic curve (x y, separated by space): ")
    P_Initial_input = input().split()
    P_Initial[0] = int(P_Initial_input[0])
    P_Initial[1] = int(P_Initial_input[1])

    print("Enter value of r: ")
    r = int(input())
    C1 = multiply(e1, r, a, p)
    if C1[0] < 0:
        C1[0] += p
    if C1[1] < 0:
        C1[1] += p
    C2 = addition(P_Initial, multiply(e2, r, a, p), a, p)
    if C2[0] < 0:
        C2[0] += p
    if C2[1] < 0:
        C2[1] += p

    print(f"P_Initial = ({P_Initial[0]},{P_Initial[1]})")
    print(f"C1 = ({C1[0]},{C1[1]})")
    print(f"C2 = ({C2[0]},{C2[1]})\n\n\nDECRYPTION:\n")

    temp = multiply(C1, d, a, p)
    if temp[0] < 0:
        temp[0] += p
    if temp[1] < 0:
        temp[1] += p

    print(f"(d x C1) is : ({temp[0]},{temp[1]})")

    # Adjust the inverse calculation and P_Final calculation
    tempI[1] = (p - temp[1]) % p
    tempI[0] = temp[0]
    print(f"Inverse of (d x C1) is : ({tempI[0]},{tempI[1]})")

    x1, y1 = temp[0], temp[1]
    x2, y2 = tempI[0], tempI[1]

    result = elliptic_curve_addition(x1, y1, x2, y2, a, p)

    print(f"The plain Text is: {result}")


if __name__ == "__main__":
    main()
```

Snapshots of Implementation:



```
Enter value of p (prime number):  67

Enter the value of a:  2

Enter the value of b:  3

Enter e1 value (array of coordinates x and y, separated by space): 2 22

Enter d (private key) value: 4

(E, e1, e2) is: (E67,2,3)

e1 = (2,22)

e2 = (13,45)


ENCRYPTION:

Enter the Plaintext in terms of a point on an elliptic curve (x y, separated by space): 24 26

Enter value of r: 2

P_Initial = (24,26)

C1 = (35,1)

C2 = (21,44)


DECRYPTION:

(d x C1) is : (23,25)

Inverse of (d x C1) is : (23,42)

The plain Text is: (24, 26)
```

Advantages and Disadvantages of Elliptic Curve Cryptography:

Elliptic Curve Cryptography (ECC) is a public key cryptography technique that uses elliptic curves over finite fields for securing communications. Here are some advantages and disadvantages of ECC:

Advantages:

1.**Security Strength**:
ECC provides strong security with much shorter key lengths compared to other asymmetric encryption algorithms like RSA. This makes ECC more efficient in terms of computational resources while maintaining a high level of security.

2.**Key Size Efficiency**:
ECC keys are shorter than RSA keys for equivalent security levels. Shorter keys result in faster computations for key generation, encryption, and decryption, making ECC more efficient in terms of processing power and bandwidth.

3.**Performance**:
ECC operations are generally faster than traditional public key algorithms like RSA. This is particularly beneficial in resource-constrained environments such as mobile devices and IoT devices.

4.**Bandwidth Efficiency**:
The shorter key lengths of ECC also lead to more efficient use of bandwidth, making it suitable for applications where bandwidth is a critical factor, such as in mobile networks.

5.**Lower Computational Power Requirements**:
ECC requires less computational power, making it suitable for devices with limited processing capabilities. This is important for battery-powered devices and systems with constrained resources.

6.**Scalability**:
ECC is easily scalable to higher security levels by increasing the key size, without a proportional increase in computational complexity. This makes it adaptable to evolving security requirements.

Disadvantages:

1.**Implementation Complexity**:
Implementing ECC correctly can be more complex than other cryptographic algorithms. Improper implementations can lead to vulnerabilities. It requires careful attention to details such as point validation and side-channel attack resistance.

2.**Lack of Standardization**:
While ECC is widely used, there is less standardization compared to algorithms like RSA. This can lead to interoperability issues and concerns about the long-term support of ECC standards.

3.**Patent Concerns**:
In the past, certain ECC techniques were covered by patents, which could affect their widespread adoption. However, many of these patents have expired, and ECC is more freely usable now.

4. **Resistance to Quantum Attacks**:
While ECC is considered secure against classical computers, its resistance to quantum attacks is an area of ongoing research. Quantum computers have the potential to break some of the underlying mathematical problems that ECC relies on, although practical quantum attacks are not yet realized.

5.**Public Misperception**:
Some people may be unfamiliar with ECC and may be more comfortable with traditional algorithms like RSA. The lack of understanding or awareness can sometimes be a barrier to adoption.

In summary, ECC offers several advantages in terms of security, efficiency, and scalability. However, proper implementation and potential future developments in quantum computing are factors to consider.

Applications of Elliptic Curve Cryptography:

Elliptic Curve Cryptography (ECC) is widely used in various applications where secure communication, data integrity, and confidentiality are crucial. Some of the key applications of ECC include:

1.**Public Key Cryptography**:
ECC is commonly used for public key cryptography, where it provides the foundation for secure key exchange, digital signatures, and other cryptographic operations. It is used in protocols like SSL/TLS for securing web communication and in secure email systems.

2.**Digital Signatures**:
ECC is employed for creating and verifying digital signatures. Digital signatures are used to ensure the authenticity and integrity of digital messages. ECC's efficiency in signature generation and verification makes it suitable for applications where computational resources are limited.

3.**Secure Communication Protocols**:
Many secure communication protocols, such as those used in VPNs, secure chat applications, and secure file transfer protocols, leverage ECC for key exchange and encryption. Its efficiency makes it particularly suitable for resource-constrained environments.

4.**Mobile and IoT Security**:
ECC is well-suited for securing communication in mobile devices and Internet of Things (IoT) devices due to its smaller key sizes and lower computational requirements. This is important for devices with limited processing power and battery life.

5.**Blockchain and Cryptocurrencies**:
ECC is widely used in blockchain technology and cryptocurrencies. It is employed for creating digital wallets, signing transactions, and securing the underlying cryptographic protocols. Bitcoin, for example, uses the elliptic curve secp256k1 for its cryptographic operations.

6.**Smart Cards and Secure Elements**:
ECC is used in smart cards and secure elements for tasks such as secure key storage, authentication, and digital signatures. The efficiency of ECC is beneficial in scenarios where the computational resources on the card are limited.

7.**Government and Military Applications**:
ECC is utilized in various government and military applications for securing sensitive information, communications, and digital identities. Its efficiency is advantageous in scenarios where real-time encryption and decryption are critical.

8.**Secure Messaging and Email Encryption**:
ECC is employed in secure messaging systems and email encryption protocols to ensure the confidentiality and integrity of communication. It is used in conjunction with other cryptographic techniques to provide end-to-end encryption.

9.**Wireless Communication Security**:
ECC is used in securing wireless communication protocols, such as those used in Wi-Fi networks and cellular networks. Its efficiency in key exchange and encryption is beneficial in the context of wireless communication.

10.**Secure Remote Access**:
ECC is used in secure remote access solutions, such as Virtual Private Networks (VPNs) and remote desktop protocols, to establish secure connections between users and remote servers.

In summary, ECC is a versatile cryptographic technique that finds applications in a wide range of scenarios, from securing internet communication to blockchain transactions and IoT devices.

CONCLUSION :

In conclusion, this report has delved into three distinct cryptographic techniques: Additive Cipher, Multiplicative Cipher, and Elliptic Curve Cryptography (ECC). Each method offers unique approaches to securing information, catering to different needs and technological landscapes.

The Additive and Multiplicative Ciphers, both falling under the category of classical symmetric encryption, provide straightforward methods for encoding and decoding messages. The Additive Cipher, or Caesar Cipher, involves shifting characters by a fixed amount, offering simplicity but susceptibility to brute-force attacks. The Multiplicative Cipher, on the other hand, introduces multiplication as an operation, adding complexity to the encryption process. However, both methods share vulnerabilities due to their reliance on fixed key values, making them less suitable for modern security standards.

In contrast, Elliptic Curve Cryptography emerges as a modern and powerful asymmetric encryption technique. Leveraging the mathematics of elliptic curves over finite fields, ECC offers strong security with shorter key lengths, making it computationally efficient and suitable for resource-constrained environments. ECC finds applications in various domains, including secure communication, mobile devices, IoT, and blockchain technologies. However, challenges such as proper implementation complexity and potential vulnerabilities to quantum attacks necessitate careful consideration in its deployment.

In summary, while the classical Additive and Multiplicative Ciphers provide a historical perspective on encryption methods, Elliptic Curve Cryptography represents a contemporary solution addressing the security demands of the digital age. The choice between these cryptographic techniques depends on the specific requirements of the application, the desired level of security, and the computational resources available. As technology evolves, the importance of robust encryption techniques remains paramount in safeguarding sensitive information across diverse domains.

REFERENCES :

- Cryptography and Network Security By Behrouz A Forouzan
- https://www.geeksforgeeks.org
- https://en.wikipedia.org/wiki/Cryptography
- https://www.techtarget.com/searchsecurity/definition/cryptography
- https://www.khanacademy.org/computing/computer-science/cryptography
- https://brilliant.org/wiki/caesar-cipher/
- https://en.wikipedia.org/wiki/Elliptic-curve_cryptography
- https://avinetworks.com/glossary/elliptic-curve-cryptography/
- https://www.geeksforgeeks.org/blockchain-elliptic-curve-cryptography/

Steps Involved in Encryption and Decryption process :

Additive Cipher:

Plain text:"encrypt this message" , Key=15.

Encryption :

Plain Text	Plain Text No.	Addition	Cipher Text No.	Cipher Text
e	4	4+15=19	19	T
n	13	13+15=28	2	C
c	2	2+15=17	17	R
r	17	17+15=32	6	G
y	24	24+15=39	13	N
p	15	15+15=30	4	E
t	19	19+15=34	8	I
t	19	19+15=34	8	I
h	7	7+15=22	22	W
i	8	8+15=23	23	X
s	18	18+15=33	7	H
m	12	12+15=27	1	B
e	4	4+15=19	19	T
s	18	18+15=33	7	H
s	18	18+15=33	7	H
a	0	0+15=15	15	P
g	6	6+15=21	21	V
e	4	4+15=19	19	T

Decryption:

Cipher Text	Cipher Text No.	Subtraction	Plain Text No.	Plain Text
T	19	19-15=4	4	e
C	2	2-15=-13	13	n
R	17	17-15=2	2	c
G	6	6-15=-9	17	r
N	13	13-15=-2	24	y
E	4	4-15=-11	15	p
I	8	8-15=-7	19	t
I	8	8-15=-7	19	t
W	22	22-15=7	7	h
X	23	23-15=8	8	i
H	7	7-15=-8	18	s
B	1	1-15=-14	12	m
T	19	19-15=4	4	e
H	7	7-15=-8	18	s
H	7	7-15=-8	18	s

P	15	15-15=0	0	a
V	21	21-15=6	6	g
T	19	19-15=4	4	e

Plain text : “encrypt this message”
Cipher text : “TCRGNEIIWXHBTHHPVT”

Multiplicative Cipher:

Plain Text : ”top secret message”, key=17

Encryption :

Plain Text	Plain text No	Multiplication	Cipher Text No	Cipher Text
t	19	323	11	L
o	14	238	4	E
p	15	255	21	V
s	18	306	20	U
e	4	68	16	Q
c	2	34	8	I
r	17	289	3	D
e	4	68	16	Q
t	19	323	11	L
m	12	204	22	W
e	4	68	16	Q
s	18	306	20	U
s	18	306	20	U
a	0	0	0	A
g	6	102	24	Y
e	4	68	16	Q

Decryption:

K^{-1} in Z_{26} , that is 17^{-1} in 26

K^{-1} obtained by Euclidean Algorithm

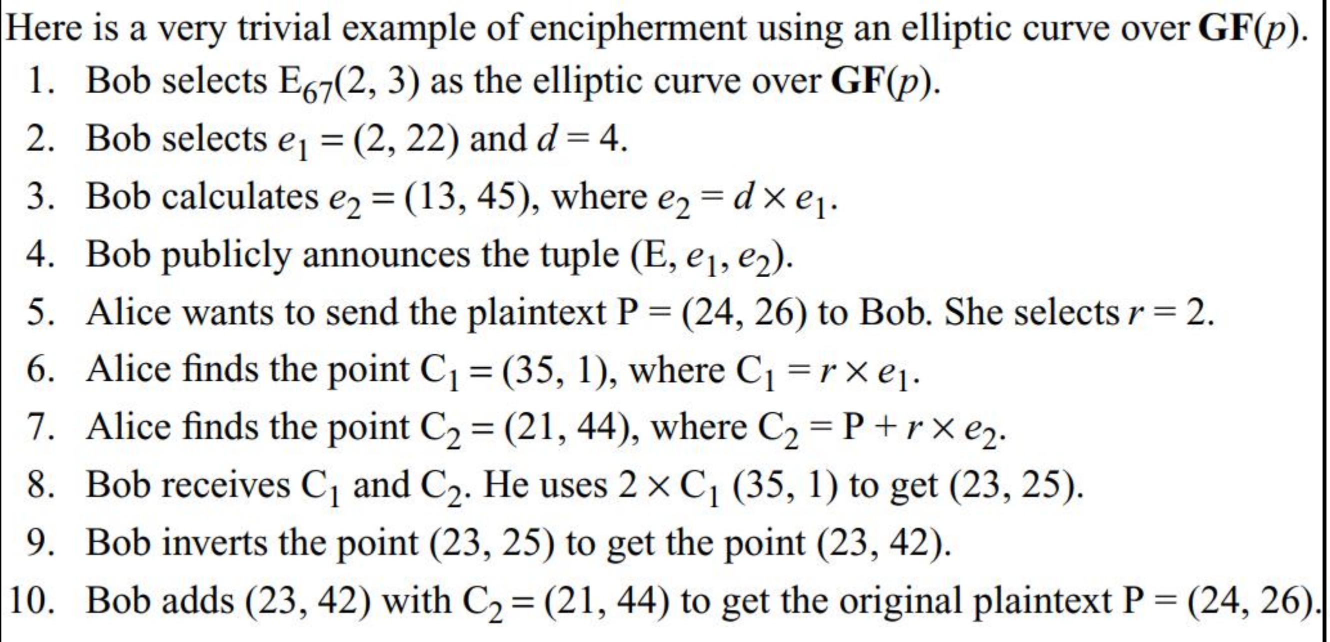
q	r1	r2	r	t1	t2	t
1	26	17	9	0	1	-1
1	17	9	8	1	-1	2
1	9	8	1	-1	2	-3
8	8	1	0	2	-3	26
	⇨ 1			⇨ -3		

17^{-1} in Z_{26} = -3+26= **23**

Cipher Text	Cipher Text No	Multiplication	Plain Text No	Plain Text
L	11	253	19	t
E	4	92	14	o
V	21	483	15	p
U	20	460	18	s
Q	16	368	4	e
I	8	184	2	c
D	3	69	7	r
Q	16	368	4	e
L	11	253	19	t
W	22	506	12	m
Q	16	368	4	e
U	20	460	18	s
U	20	460	18	s
A	0	0	0	a
Y	24	552	6	g
Q	16	368	4	e

Plain text : “top secret message”
Cipher text : “LEVUQIDQLWQUUAYQ”

Elliptic Curve Cryptography Encryption and Decryption:


Here is a very trivial example of encipherment using an elliptic curve over **GF**(p).

1. Bob selects $E_{67}(2, 3)$ as the elliptic curve over **GF**(p).
2. Bob selects $e_1 = (2, 22)$ and $d = 4$.
3. Bob calculates $e_2 = (13, 45)$, where $e_2 = d \times e_1$.
4. Bob publicly announces the tuple (E, e_1, e_2).
5. Alice wants to send the plaintext P = (24, 26) to Bob. She selects $r = 2$.
6. Alice finds the point $C_1 = (35, 1)$, where $C_1 = r \times e_1$.
7. Alice finds the point $C_2 = (21, 44)$, where $C_2 = P + r \times e_2$.
8. Bob receives C_1 and C_2. He uses $2 \times C_1$ (35, 1) to get (23, 25).
9. Bob inverts the point (23, 25) to get the point (23, 42).
10. Bob adds (23, 42) with $C_2 = (21, 44)$ to get the original plaintext P = (24, 26).

Plain text:(2,22)
Cipher text: (35,1),(21,44)