Blockchain – Elliptic Curve Cryptography

REPORT BY	PRN	Roll No
GAURAV SUNTHWAL	1032232584	40
Himashra Kumar Mehra	1032233372	61

The following topics of Elliptic Curve Cryptography will be discussed here:

1. Introduction to Elliptic Curve Cryptography
2. History of Elliptic Curve Cryptography
3. Components of Elliptic Curve Cryptography
4. Elliptic Curve Cryptography Algorithms
5. Application of Elliptic Curve Cryptography
6. ECC vs RSA
7. Elliptic Curve Diffie-Hellman Protocol Implementation
8. Types of Security Attacks
9. Benefits of Elliptic Curve Cryptography
10. Limitations of Elliptic Curve Cryptography
11. Conclusion

• Aim

This report explores Elliptic Curve Cryptography (ECC) as a modern cryptographic framework, analyzing its mathematical foundations, practical implementations, and real-world applications. ECC provides high security with smaller key sizes, making it a superior alternative to traditional schemes like RSA and DSA.

The study covers the structure of elliptic curves, finite field arithmetic, and the Elliptic Curve Discrete Logarithm Problem (ECDLP)—the core of ECC's security. By comparing ECC with conventional algorithms, it highlights its advantages in efficiency, resource optimization, and resistance to modern cryptographic attacks. Additionally, the report examines ECC's role in TLS/SSL encryption, digital signatures, blockchain, IoT security, and secure messaging while addressing implementation challenges like standardization, hardware compatibility, and side-channel attacks.

Ultimately, this report provides a comprehensive evaluation of ECC's efficiency, scalability, and future viability, particularly in the face of quantum computing advancements.

- # Introduction to Elliptic Curve Cryptography

In an era where data security is paramount, cryptographic algorithms must balance robustness with efficiency. Traditional systems like RSA rely on the difficulty of factoring large primes, but their computational overhead and large key sizes make them less viable for resource-constrained environments. Enter Elliptic Curve Cryptography (ECC)—a groundbreaking approach that offers equivalent security with significantly smaller keys. ECC leverages the algebraic structure of elliptic curves over finite fields, enabling faster computations and reduced energy consumption. Its adoption in protocols like TLS 1.3 and blockchain networks underscores its growing importance. However, challenges such as mathematical complexity and quantum vulnerability persist. This report examines ECC's transformative potential, its technical underpinnings, and its role in shaping the future of cryptography.

- # Objectives

1. **Analyze Mathematical Foundations** – Explore the structure of elliptic curves, finite field arithmetic, and the Elliptic Curve Discrete Logarithm Problem (ECDLP).
2. **Compare Efficiency with RSA** – Evaluate ECC's computational efficiency, resource consumption, and key size advantages over traditional cryptographic algorithms.
3. **Assess Security Mechanisms** – Examine ECC's resistance to cryptographic attacks, including its vulnerabilities in a post-quantum computing era.
4. **Identify Real-World Applications** – Investigate ECC's role in securing TLS/SSL, blockchain, IoT, digital signatures, and secure messaging.
5. **Address Implementation Challenges** – Discuss standardization, hardware compatibility, side-channel attacks, and regulatory considerations.
6. **Evaluate Future Viability** – Analyze ECC's long-term sustainability, potential replacements, and integration with post-quantum cryptography.

- # Components of Elliptic Curve Cryptography

1. ECC keys:

Private key: ECC cryptography's private key creation is as simple as safely producing a random integer in a specific range, making it highly quick. Any integer in the field represents a valid ECC private key.

Public keys: Public keys within ECC are EC points, which are pairs of integer coordinates x, and y that lie on a curve. Because of its unique features, EC points can be compressed to a single coordinate + 1 bit (odd or even). As a result, the compressed public key corresponds to a 256-bit ECC.

2. Generator Point:

ECC cryptosystems establish a special pre-defined EC point called generator point G (base point) for elliptic curves over finite fields, which can generate any other position in its subgroup over the elliptic curve by multiplying G from some integer in the range [0...r].

The number r is referred to as the "ordering" of the cyclic subgroup.

Elliptic curve subgroups typically contain numerous generator points, but cryptologists carefully select one of them to generate the entire group (or subgroup), and is excellent for performance optimizations in calculations. This is the "G" generator.

- # Encryption algorithms:

- **Elliptic Curve Integrated Encryption Scheme (ECIES):** ECIES is a public-key authenticated encryption scheme that uses a KDF (key-derivation function) to generate a separate Medium Access Control key and symmetric encryption key from the ECDH shared secret. Because the ECIES algorithm incorporates a symmetric cipher, it can encrypt any amount of data. In practice, ECIES is used by standards such as Intelligent Transportation Systems.

- **EC-based ElGamal Elliptic Curve Cryptography:** ElGamal Elliptic Curve Cryptography is the public key cryptography equivalent of ElGamal encryption schemes that employ the Elliptic Curve Discrete Logarithm Problem. ElGamal is an asymmetric encryption algorithm that is used to send messages securely over long distances. Unfortunately, if the encrypted message is short enough, the algorithm is vulnerable to a Meet in the Middle attack.

- # Application of Elliptic Curve Cryptography

Diffie-Hellman: The basic public-key cryptosystem suggested for secret key sharing is the Diffie-Hellman protocol. If A (Alice) and B (Bob) initially agree on a given curve, field size, and mathematical type. They then distribute the secret key in the following manner. We can see that all we need to build the Diffie-Hellman protocol is scalar multiplication.

Elliptic Curve Digital Signature Algorithm (ECDSA): ECC is one of the most widely utilized digital signature implementation approaches in cryptocurrencies. In order to sign transactions, both Bitcoin and Ethereum use the field inverse multiplication, but also arithmetic multiplication, inverse function, and modular operation.

Online application: Moreover, ECC is not limited to cryptocurrencies. It is an encryption standard that will be utilized by most online apps in the future due to its reduced key size and efficiency. Most commonly used in cryptocurrencies such as Bitcoin and Ethereum, along with single-way encryption of emails, data, and software.

Blockchain application: The cryptocurrency Bitcoin employs elliptic curve cryptography. Ethereum 2.0 makes heavy use of elliptic curve pairs with BLS signatures, as stated in the IETF proposed BLS specification, to cryptographically ensure that a specific Eth2 validator has really verified a specific transaction.

- # Key Agreement algorithm:

Criteria	ECDH	FHMQV
Mutual Authentication (M)	No built-in authentication; vulnerable to MITM attacks	Provides implicit authentication via signature-like key validation
Robustness (R)	Secure but lacks resilience against active attacks	More resilient due to the fully hashed mechanism
Efficiency (E)	Simple and computationally efficient	Slightly more complex but optimized for security

- **ECC vs RSA**

Feature	ECC	RSA
Key Size	Smaller	Larger
Security	Higher per bit	Lower per bit
Computational Speed	Faster	Slower
Energy Efficiency	More efficient	Less efficient
Usage	IoT, Blockchain, TLS	Legacy systems, Banking

- **ECC vs RSA: Key Length Comparison:**

Security(in Bits)	RSA key length required	ECC key length required
80	1024	160-223
112	2048	224-255
128	3072	256-383
192	7680	384-511
256	15360	512+

- **Install tinyec in Python:**

pip install tinyec

```
# Importing required libraries used
# to perform arithmetic operations
# on elliptic curves
from tinyec import registry
import secrets

# Function to calculate compress point
# of elliptic curves
def compress(publicKey):
 return hex(publicKey.x) + hex(publicKey.y % 2)[2:]

# The elliptic curve which is used for the ECDH calculations
curve = registry.get_curve('brainpoolP256r1')

# Generation of secret key and public key
Ka = secrets.randbelow(curve.field.n)
X = Ka * curve.g
print("X:", compress(X))
Kb = secrets.randbelow(curve.field.n)
Y = Kb * curve.g
print("Y:", compress(Y))
print("Currently exchange the publickey (e.g. through Internet)")

# (A_SharedKey): represents user A
# (B_SharedKey): represents user B
A_SharedKey = Ka * Y
print("A shared key :",compress(A_SharedKey))
B_SharedKey = Kb * X
print("(B) shared key :",compress(B_SharedKey))
print("Equal shared keys:", A_SharedKey == B_SharedKey)
```

```
from tinyec import registry
import secrets

def compress(publicKey):
 return hex(publicKey.x) + hex(publicKey.y % 2)[2:]

curve = registry.get_curve('brainpoolP256r1')
Ka = secrets.randbelow(curve.field.n)
X = Ka * curve.g
print("X:", compress(X))
Kb = secrets.randbelow(curve.field.n)
Y = Kb * curve.g
print("Y:", compress(Y))
print("Currently exchange the publickey (e.g. through Internet)")
#(A): represents person A
#(B): represents person B
A_SharedKey = Ka * Y
print("Person A shared key :",compress(A_SharedKey))
B_SharedKey = Kb * X
print("Person B shared key :",compress(B_SharedKey))
print("Whether shared keys are equal:", A_SharedKey == B_SharedKey)
```

```
X: 0x24a00fe479b09b7a2117bc5d800d0491068e216b4eb172738f91dd9690d7e74d0
Y: 0x5bc45a276989a685af69d39d49ab8ce3492a73ac65a921958bf8db22b28e3bd91
Currently exchange the publickey (e.g. through Internet)
Person A shared key : 0x8bc97592d3d342cfd30633b3556c7b182ba92aaa781091c7117c7238aab02c101
Person B shared key : 0x8bc97592d3d342cfd30633b3556c7b182ba92aaa781091c7117c7238aab02c101
Whether shared keys are equal: True
```

- **Explanation:**

The following Python code generates an ECC private-public key pair for the recipient of the message (based on the brainpoolP256r1 curve), then derives a secret shared key (for encryption) and an ephemeral cipher – text key (for ECDH) from the recipient's public key, and then derives same secret key pair (for decryption) from the recipient's secret key and the previously generated ephemeral ciphertext public key.

In an integrated encryption scheme, these keys will be utilized for data encryption and decryption. If you execute the code, the above output will differ (due to the randomness used to produce ciphertextPrivKey), but the decryption and encryption keys will remain the same.

The process for producing a shared ephemeral secret key based on an ECC key pair described above is an example of a KEM (a key encapsulating mechanism) based on the ECC and ECDH.

- **Limitations of Elliptic Curve Cryptography**

Large encryption size: ECC increases the size of the encrypted message significantly more than RSA encryption. The default key length for ECC private keys is 256 bits, but many different ECC key sizes are conceivable depending on the curve.

A more complex: The ECC algorithm is more complete and more difficult to implement than RSA. Algorithms cost have been computed from the computation of the elliptic curve operation and finite field operations that determine the running time of the scalar multiplication integer sub-decomposition (ISD) method.

Complex security: Complicated and tricky to implement securely, mainly the standard curves. If the key size used is large enough, ECC is regarded to be highly secure. For internal communications, the US government needs ECC with a key size of either 256 or 384 bits, depending on the sensitivity level of the material being communicated.

Binary curves: Processing of binary curves is costly. Elliptic curve cryptography (ECC) employs elliptic curves over finite fields Fp (where p is prime and p > 3) or F2m (where the field size p = 2 m_). This means that the field is a p x p square matrix, and the points on the curve can only have integer locations within the field.

Types of Security Attacks

- **Side-channel attack:** Side-channel attacks in elliptic curve cryptography are caused by unintended information leaking during processing. The computation of n*P, where n is a positive number and P is a location on the elliptic curve E, is a critical operation.

- **Backdoor attack:** Concerns have been made by cryptographic specialists that the National Security Agency has installed a kleptographic backdoor into at least one elliptic curve-based pseudo-random generator. According to one investigation of the potential backdoor, an attacker in possession of the algorithm's secret key might access encryption keys provided only 32 bytes of outputs.

- **Quantum computing attacks:** By calculating discrete logarithms on a hypothetical quantum computer, Shor's technique can be used to break elliptic curve cryptography. The most recent quantum resource estimates are 2330 qubits and 126 billion Toffoli gates for cracking a curve with only a 256-bit modulus (128-bit security level).

Benefits of Elliptic Curve Cryptography

- Fast key generation: ECC cryptography's key creation is as simple as securely producing a random integer in a specific range, making it highly quick. Any integer in the range represents a valid ECC secret key. The public keys in the ECC are EC points, which are pairs of integer coordinates x, and y that lie on a curve.

- Smaller key size: Cipher text, signatures, and Elliptic-curve cryptography (ECC) is a public-key encryption technique based on the algebraic structure of elliptic curves with finite fields. Compared to non-EC encryption (based on ordinary Galois fields), ECC allows for fewer keys to guarantee equal security.

- Low latency: Signatures can be computed in two stages, allowing latency much lower. By computing signatures in two stages, ECC achieves lower latency than the inverse throughout. ECC has robust protocols for authorized key exchange, and the technology has widespread adoption.

- **Conclusion**

Encryption strength: The main distinction between RSA and ECC certificates is the encryption strength. When compared to other approaches, such as RSA, ECC can provide a level of security that uses fewer processing resources to encrypt and decrypt data.

ECC Keys feature: With a lower key length, Elliptic Curve Cryptography (ECC) delivers the same level of encryption strength as the RSA.ECC and other public key encryption systems use a mathematical technique to combine two separate keys and then use the resulting output to encrypt and decrypt data. One is a public key that anybody can see, and the other is a private key that only the sender and receiver of the data can see.

ECC certificates: As a result, for Public Key Infrastructure, an ECC certificate provides more speed and security than an RSA certificate. Elliptic Curve Cryptography (ECC) provides an equivalent level of encryption strength to the RSA algorithm with a shorter key length.

ECC curves: The elliptic curve over a finite area gives us more security. For contemporary ECC purposes, an elliptic curve is a plane curve over a finite field composed of points fitting the equation: Any point on the curve in this elliptic curve cryptography example can be mirrored over the x-axis and the curve will remain unchanged.

Use of prime number: Zp (where p is a prime number) elliptic curve When p is a huge prime integer, it indicates that the cipher text is extremely tough to crack. The public and shared keys are both 257 bits long (65 hexadecimal digits, 256 bits due to key compression). The private keys KA and KB are different due to randomness, but the estimated shared secret key across (A) and (B) will always be the same.