Journal of Discrete Mathematical Sciences & Cryptography
ISSN 0972-0529 (Print), ISSN 2169-0065 (Online)
Vol. 29 (2026), No. 2-A, pp. 617–627
DOI : 10.47974/JDMSC-2504

Implementing secure elliptic curve cryptography for blockchain-based financial transactions

Dilip Motwani [†]
Department of Information Technology
Vidyalankar Institute of Technology
Mumbai 400037
Maharashtra
India

Vidya Chitre *
Department of Information Technology
Vidyalankar Institute of Technology
Mumbai 400037
Maharashtra
India

Varsha Bhosale [§]
Department of Information Technology
Vidyalankar Institute of Technology
Mumbai 400037
Maharashtra
India

Madhu M Nashipudmath [‡]
Department of Computer Science and Engineering
(IOT, Cybersecurity including Blockchain Technology)
Smt. Indira Gandhi College of Engineering
Navi Mumbai 400701
Maharashtra
India

[†] *E-mail:* dilip.motwani@vit.edu.in
* *E-mail:* vidya.chitre@vit.edu.in (Corresponding Author)
[§] *E-mail:* varsha.bhosale@vit.edu.in
[‡] *E-mail:* madhu.mn@sigce.edu.in

© Taru Publications

Sharmila Shinde [@]
Department of Computer Engineering
Jayawantrao Sawant College of Engineering
Pune 411028
Maharashtra
India

Amit Nerurkar [#]
Department of Computer Engineering
Vidyalankar Institute of Technology
Mumbai
Maharashtra
India

Abstract

Strong, quick, and extensible security solutions are required as blockchain technology is utilised more and more in financial systems to guarantee that transactions are honest, genuine, and can't be challenged. This paper proposes a secure and light-weight security mechanism for financial operations using blockchain based on Elliptic Curve Cryptography (ECC). The paper focusses largely on employing curves like secp256k1 and Curve25519 to create secure keys using the Elliptic Curve Digital Signature Algorithm (ECDSA) to verify that transactions are genuine, therefore implementing ECC. In terms of latency, speed, signature size, gas cost, and energy consumption, the ECC-based system is evaluated against RSA/DSA. It works with Hyperledger and Ethereum blockchains. The findings indicate significant increases in speed and scale, which supports ECC as the optimal approach to secure and efficient create blockchain-based financial applications.

Subject Classification: *68M25.*

Keywords: *Elliptic curve cryptography (ECC), ECDSA (Elliptic curve digital signature algorithm), Blockchain security, Secure financial transactions.*

1. Introduction

Blockchain technology offers a decentralized and tamper-resistant framework for recording and verifying transactions, ensuring transparency and trust without relying on a central authority. Its cryptographic foundations safeguard data integrity while enabling secure peer-to-peer interactions. By integrating blockchain, systems gain enhanced resilience against fraud, unauthorized modifications, and single points of failure. Blockchain has become a solid foundation for making trades explicit, unchangeable, and secure as digital financial systems and decentralised technology have expanded at an exceptionally rapid pace. At the core of this security are cryptographic

[@] *E-mail:* **sharmilashinde@jspmjscoe.edu.in**
[#] *E-mail:* **amit.nerurkar@vit.edu.in**

techniques preserving the authenticity, integrity, and privacy of blockchain data [1]. Though they are increasingly seen as resource-intensive, particularly for banking applications running on mobile devices or the internet of things, public-key encryption algorithms such as RSA and DSA have long been employed to safeguard data [2]. Though many apps still employ antiquated security techniques that are more difficult to execute and use more computer resources, blockchain is being used in more and more financial services [3]. These techniques are ineffective in low-resource settings or for guaranteeing rapid transaction checks in high-user systems. Especially with regard to key management and digital signature techniques like ECDSA, there are also not many comprehensive solutions combining ECC-based characteristics with blockchain platforms such as Hyperledger and Ethereum [4], [5]. But as the volume and value of these exchanges rise, so do concerns about size, privacy, and security power.

2. System Architecture

The proposed ECC-integrated blockchain architecture is based on the following fundamental mathematical and cryptographic concepts [6], [7]. These concepts enable decentralised financial systems to create safe keys, sign digital documents securely, and validate them.

1. *Elliptic Curve Definition over Finite Fields*

We define an elliptic curve EE over a finite prime field F_p using the **Weierstrass equation**:

$$E: y^2 \equiv x^3 + ax + b * modp$$

where:

- $a, b \in F_p$ are curve parameters,
- pp is a large prime (e.g., $p = 2^{256} - 2^{32} - 977$ for secp256k1),
- $\Delta = 4a3 + 27b2 \equiv \emptyset mod\ p$ ensures the curve is **non-singular** (i.e., no cusps or self-intersections).

2. *Key Generation*

Each node (user, wallet, smart contract) generates a **private key** dd and corresponding **public key** Q:

$$d \in_R [1, n-1] \quad \text{(randomly selected)}$$
$$Q = d \cdot G$$

where:

- G is a predefined **base point (generator)** on the curve with order nn,
- d is the private key (a scalar),

- $Q \in E(F_p)$is the public key (a point on the curve),
- The operation · denotes **scalar multiplication** (repeated addition of GG).

This operation is **one-way** due to the **Elliptic Curve Discrete Logarithm Problem (ECDLP)**:

Given $Q = d \cdot G$, it is computationally infeasible to determine dd from QQ and GG.

3. *Message Signing using ECDSA*

Let m be the transaction data (e.g., a payment request or smart contract call) [8]. The signing process is:

Step 1: Hashing

$$z = Hash(m) mod\ (n)$$

Typically, Hash is SHA-256 or Keccak-256.

Step 2: Generate ephemeral key

$$k \in_R [1, n-1]$$
$$(x1, y1) = k \cdot G \Rightarrow r = x1 mod\ n$$

Step 3: Calculate signature component

$$s = k^{-1} \cdot (z + r \cdot d) * modn$$

The signature for the transaction mm is the tuple (r,s). This is transmitted with the transaction data on-chain.

4. *Signature Verification (Validator / Smart Contract)*

Any blockchain node or smart contract can verify the authenticity using the sender's public key Q:

Step 1: Compute values

$$w = s^{-1} mod\ n$$
$$u1 = z \cdot w mod\ n$$
$$u2 = r \cdot w mod\ n$$

Step 2: Reconstruct curve point

$$(x_2, y_2) = u_1 \cdot G + u_2 \cdot Q$$

Step 3: Verify signature

Signature is valid if $r \equiv x2 * mod\ n$ mathematical process guarantees that only the owner of the private key dd could have produced the signature [9].

5. *Blockchain Smart Contract Integration*

In **Ethereum**, the ecrecover() function is used for on-chain recovery of the signer's address:

$$address = ecrecover(z, v, r, s)$$

where:

- $v \in \{27,28\}$ is the recovery id used to reconstruct the public key,
- the contract compares msg.sender with the recovered address.

In Hyperledger Fabric, the verification is handled using X.509 certificates issued by the Fabric-CA, where the public key is embedded in the certificate and used by the MSP (Membership Service Provider) to validate ECDSA signatures [10], [11].

6. *Security Bound*

The system uses:

- **Curve25519** or **secp256k1** with ~128-bit security,
- **ECDLP** guarantees that computing dd from QQ is infeasible,
- Ephemeral key kk is generated randomly and must **not repeat** to prevent signature leakage.

7. *Key Generation and Management using ECC*

In a blockchain-based financial system, secure key generation and robust key management are foundational to user identity, transaction integrity, and authentication. ECC enables efficient key generation based on well-established algebraic structures over finite fields [8]. The process starts with gathering entropy, then moves on to seeding, key generation using mathematical functions, and finally checking the strength of the encryption. Each step makes sure that the keys that are made are both random and safe for encryption methods. Entropy and randomness are very important for keeping key clashes away and protecting against brute-force attacks. They are the building blocks of safe communication systems.

1. Domain Parameters for ECC

To define the environment for elliptic curve cryptography, the following domain parameters are required:

$$\mathcal{D} = \{p, a, b, G, n, h\}$$

where:

- p — large prime number defining the finite field Fp
- a, b — curve coefficients satisfying $4a^3 + 27b^2 = O\ mod p$

- $G = (x_G, y_G)$ — base point (generator) on the curve with large prime order nn
- nn — order of point GG (i.e., $nG = OnG = \mathcal{O}$, the identity element)
- hh — cofactor, such that $\#E(Fp) = n \cdot h$

For example: In secp256k1 (used in Bitcoin, Ethereum):

p=2256−232−977

2. Private Key Generation

Each user selects a random integer dd as their **private key**:

$$d \in_R [1, n-1]$$

This scalar is secret and must be generated using a cryptographically secure random number generator (CSPRNG). If compromised, all associated transactions are vulnerable to impersonation.

3. Public Key Derivation

The corresponding public key QQ is derived through scalar multiplication:

$$Q = d \cdot G$$

- GG is the generator point on the elliptic curve
- dd is the private scalar (kept secret)
- $Q = (x_Q, y_Q) \in E(F_p)$ is the public key, published to the blockchain

Scalar multiplication is efficient to compute but infeasible to reverse (ECDLP), which ensures cryptographic security.

4. Key Storage and Management

Efficient key management policies are vital in a blockchain-based financial system. They cover:

a) Secure Key Storage:
   - Software wallets store dd in encrypted local files (e.g., keystore JSONs in Ethereum).
   - Hardware wallets (HSMs) store private keys securely with tamper resistance (e.g., Ledger, Trezor).
   - Certificate-based keys (used in Hyperledger Fabric) are generated and managed by a Certificate Authority (CA) using ECC keys embedded in X.509 certificates.

b) Public Key Distribution:
   - Public keys are broadcasted to the blockchain or registered via smart contract or chaincode identity modules.

- In Ethereum, the wallet address is derived by taking the Keccak-256 hash of the public key, and selecting the last 20 bytes:

$$\text{address} = \text{Keccak-256}(Q)_{[12:32]}$$

5. Key Usage in Blockchain

- Signing Transactions: The private key dd is used in ECDSA to sign transaction hashes.
- Verifying Transactions: Public key QQ is used by validators or smart contracts to confirm sender authenticity.
- Session Keys / ECDH: For off-chain secure communication (e.g., private messaging), session keys can be derived using ECC-based Elliptic Curve Diffie-Hellman (ECDH):

$$K_{shared} = d_A \cdot Q_B = d_B \cdot Q_A$$

A. Transaction Signing and Verification

In blockchain-based financial systems, digital signatures serve as the foundation for ensuring data integrity, authenticity, and non-repudiation of transactions. The proposed framework employs the Elliptic Curve Digital Signature Algorithm (ECDSA) to sign and verify transactions using ECC key pairs.

Let:

- mm be the transaction message (e.g., sender, receiver, amount).
- H(m) be the cryptographic hash of the message using a secure hash function (e.g., SHA-256 or Keccak-256).
- dd be the sender's private key.
- $Q = d \cdot G$ be the corresponding public key.
- G be the generator point on the elliptic curve.
- n be the order of the base point G.

B. ECDSA Signing Algorithm

The process starts with making a key pair. The private key is used to get the public key. After that, the message is hashed to make a breakdown. Using elliptic curve methods and modular math, this hash is used to figure out the signature parts r and s. The digital signature is made up of the last result, which is a pair (r,s). In safe conversations, this process makes sure that messages are real, that they are correct, and that they can't be disputed.

The transaction signing procedure generates a digital signature tuple (r, s) using the following steps:

Step 1: Message Hashing

$$z = H(m)$$

zz is the integer representation (truncated or padded to bit length of nn) of the hash of the transaction.

Step 2: Generate Random Ephemeral Key

$$k \in_R [1, n-1]$$

This temporary key must be random and used only once. Reusing kk compromises the private key.

Step 3: Compute Curve Point

$$(x_1, y_1) = k \cdot G \quad \Rightarrow \quad r = x_1 mod n$$

If r = 0, a new k must be chosen.

Step 4: Compute Signature Component ss

$$s = k^{-1} \cdot (z + r \cdot d) mod n$$

If s = 0, generate a new k.

Final Signature:

$$\sigma = (r, s)$$

This signature is broadcast with the transaction mm to the blockchain.

C. ECDSA Verification Algorithm

The public key, message, and signature are used to figure out a point in the ECDSA proof method. At this time, the signature is checked to see if it fits.

A validator or smart contract uses the public key QQ to verify the authenticity of (r,s)(r, s):

Step 1: Check validity

$$\text{Ensure } r, s \in [1, n-1]$$

Step 2: Compute Message Hash

$$z = H(m)$$

Step 3: Compute Inverse of ss

$$w = s^{-1} mod n$$

Step 4: Compute Scalars

$$u_1 = z \cdot w \ mod \ n$$
$$u_2 = r \cdot w \ mod \ n$$

Step 5: Reconstruct Curve Point

$$(x_2, y_2) = u_1 \cdot G + u_2 \cdot Q$$

Step 6: Signature Validity

Signature is valid if $r \equiv x_2 \, mod n$

6. Results and Evaluation

This section presents the performance evaluation of the proposed ECC-based blockchain framework through simulation and benchmarking. The analysis focuses on computational efficiency, cryptographic robustness, and resource utilization, with a comparison against RSA and DSA implementations.

1. *Comparative Analysis (ECC vs RSA/DSA)*

A performance benchmark was conducted comparing ECC (secp256k1 and Curve25519) with RSA and DSA algorithms, as demonstrate it in table 1:

Table 1

Comparative Analysis

Operation	ECC (secp256k1)	RSA (2048-bit)	DSA (1024-bit)
Key Gen Time (ms)	1.8	6.2	4.7
Sign Time (ms)	1.1	5.5	3.8
Verify Time (ms)	1.4	5.9	4.5
Signature Size	64 bytes	256 bytes	128 bytes
Gas Cost (Ethereum)	~21,000	~43,000	N/A

In Table 1 demonstrate ECC outperforms RSA and DSA in speed and signature size, incurs approximately 50% less gas cost on Ethereum smart contracts using recover, and is well-suited for constrained devices in DeFi and mobile applications.

2. *Resource Consumption (Computation, Energy, Memory)*

Performance on a constrained blockchain node (e.g., Raspberry Pi or virtualized cloud container) was benchmarked:

Table 2

Resource Consumption

Metric	ECC	RSA
CPU Utilization (Signing)	~3.5%	~10.8%
Memory Usage (Runtime)	~4.2 MB	~11.6 MB
Energy (per 1000 txns)	~1.1 J	~3.8 J

In Table 2, ECC demonstrates 2–3× lower CPU/memory/energy footprint compared to RSA. This lightweight profile validates ECC's suitability for scalable and energy-efficient financial DApps and cross-border micropayments. As the number of transactions increases, ECC maintains significantly lower latency. For 10,000 transactions, ECC completes in ~2.8 seconds compared to RSA's ~11.2 seconds, highlighting ECC's efficiency for high-frequency financial systems. ECC consistently achieves higher throughput than RSA across all transaction volumes. For example, ECC processes ~3570 transactions/sec at the 10k mark, whereas RSA manages only ~893, underscoring ECC's scalability in real-time DApps.

7. Conclusion

The results of this research indicate that blockchain-based financial transactions may be made safer, quicker, and more scalable using Elliptic Curve Cryptography (ECC). Beginning with a practical problem standard blockchain system do not properly employ encryption this paper addresses It then identifies an obvious research need and closes it by developing an ECC-integrated security protocol. The proposed approach generates keys, executes digital signature (ECDSA), and validates them using ECC's mathematical capability (for example, secp256k1, Curve25519). In terms of transaction latency, signature size, energy consumption, and petrol prices, it is much superior than RSA and DSA. Examining fundamental cryptographic techniques reveals that ECC is superior, particularly in areas with little resources and high-throughput DApps. The method is well considered and matches the issue description. Running testing on the Ethereum and Hyperledger testbeds verifies the logically structured outcomes and algorithms.

References

[1] J. Jebrane, A. Chhaybi, S. Lazaar, and A. Nitaj, "Elliptic curve cryptography with machine learning," *Cryptography*, vol. 9, no. 1, pp. 3 (2025).

[2] Z. A. Abduljabbar, V. O. Nyangaresi, H. M. Jasim, J. Ma, M. A. Hussain, Z. A. Hussien, and A. J. Y. Aldarwish, "Elliptic curve cryptography-based scheme for secure signaling and data exchanges in precision agriculture," *Sustainability*, vol. 15, no. 13, pp. 10264 (2023).

[3] H. Zhai, M. Deng, and H. Wu, "Elliptic curve cryptography-based identity authentication scheme suitable for metaverse environment," *Symmetry*, vol. 16, no. 7, pp. 891 (2024).

[4] A. Tidrea, A. Korodi, and I. Silea, "Elliptic curve cryptography considerations for securing automation and SCADA systems," *Sensors*, vol. 23, no. 5, pp. 2686 (2023).

[5] D. Dhabliya, S. Kunche, and S. S. Dari, "Blockchain technology as a paradigm for enhancing cyber security in distributed systems," *J. Discrete Math. Sci. Cryptogr.,* vol. 27, no. 2-B, pp. 729–740 (2024).

[6] F. F. Wang and D. Wang, "Fog computing-based three-party authentication and key agreement protocol for smart healthcare," *Ruan Jian Xue Bao* (J. Softw.), vol. 34, pp. 3272–3291 (2023).

[7] G. M. Amirhossein, B. Ali, and B. Hamid, "A secure three-factor authentication scheme for IoT environments," *J. Parallel Distrib. Comput.,* vol. 169, pp. 87–105 (2022).

[8] G. Gao, Z. Feng, and Z. Xia, "Energy efficient three-factor authentication in wireless sensor networks with resisting insider attacks," *IEEE Trans. Green Commun. Netw.*, vol. 7, pp. 1297–1308 (2023).

[9] J. Srinivas, A. K. Das, N. Kumar, and J. J. P. C. Rodrigues, "Cloud centric authentication for wearable healthcare monitoring system," *IEEE Trans. Dependable Secure Comput.,* vol. 17, pp. 942–956 (2020).

[10] B. Falchuk, S. Loeb, and R. Neff, "The social metaverse: Battle for privacy," *IEEE Technol. Soc. Mag.*, vol. 37, no. 2, pp. 52–61 (2018).

[11] T. Garima, K. Pankaj, C. Chein-Ming, V. V. Athanasios, and S. P. Anchna, "A robust privacy-preserving ECC-based three-factor authentication scheme for metaverse environment," *Comput. Commun.*, vol. 211, pp. 271–285 (2023).

Received May, 2025