International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 13 Issue 12 ǁ December 2025 ǁ PP. 111-119

Developing A Distributed Shared-Key Cryptosystem Architecture for Decentralized Blockchain Transaction Security

Akinrolabu, Olatunde David, Adefuwa, Hussein Oluwaponmile, Akinpetide, Akinwale Moses

[1] *Department of Data Science, Adekunle Ajasin University, Akungba-Akoko, Ondo State, Nigeria.*
[2] *Department of Computer Science, Lagos State University, Lagos State, Nigeria.*
[3] *International Business Management, Robert Gordon University, Scotland, United Kingdom.*
**Corresponding Author: Akinrolabu Olatunde David, Department of Data Science, Adekunle Ajasin University, Akungba. Orchid: https://orcid.org/0009-0004-3337-0240*

Abstract: *Blockchain technology has emerged as a secure platform for decentralised transactions, yet key distribution and device authentication remain persistent challenges, especially in systems that traditionally rely on centralised Public Key Infrastructure (PKI) and Certificate Authorities (CA). This study proposes a distributed shared-key cryptographic framework that leverages blockchain consensus and smart contracts to securely generate, register, and distribute symmetric keys for peer-to-peer communication. The system integrates Solidity-based smart contracts with AES encryption to create a trustless environment where keys are validated, transactions are logged immutably, and message integrity is preserved without relying on third-party authorities. The architecture comprises modules for key generation, registration, encryption, decryption, and secure message delivery, all deployed within a local blockchain environment. Experimental results demonstrate that the framework achieves low latency in key retrieval, efficient AES operations, and reliable integrity verification, making it suitable for lightweight and distributed applications. The findings show that symmetric shared keys, when coordinated through smart contracts, can provide a scalable and decentralised alternative to certificate-based authentication. This research contributes a practical model for enhancing blockchain security and offers a foundation for future implementations in IoT networks, fintech systems, and distributed transaction platforms.*
Keywords: *Shared Key, Blockchain, Public Key, Solidity, Private Key, Smart Contract, Cryptosystem, Decentralisation*

Date of Submission: 07-12-2025 Date of acceptance: 19-12-2025

I. Introduction

Blockchain technology has emerged as a decentralised and tamper-resistant platform for recording transactions across distributed nodes without relying on a central authority [4]. In this architecture, each node maintains a synchronised copy of the ledger, and consensus protocols validate new blocks, providing transparency, traceability and immutability. Blockchain's ability to store cryptographically secured data records has enabled its adoption across multiple industries, including finance, healthcare, supply chain, real estate, digital identity, and public administration [3]. Beyond cryptocurrencies, modern applications such as IoT, digital voting systems, and secure data sharing leverage blockchain to eliminate intermediaries, enhance accountability, and provide real-time visibility into distributed processes. Its cryptographic foundations support secure information exchange, but challenges remain, especially in key management, device authentication, and scalable trust formation [1]

Traditional security frameworks such as Public Key Infrastructure (PKI) and Certificate Authorities (CAs) require centralised certificate issuance and management, making them costly, difficult to deploy at scale, and vulnerable to compromise, as evidenced by CA breaches such as the DigiNotar and VeriSign incidents. For large distributed environments, especially IoT networks, issuing and managing individual certificates is impractical. Hard-coding credentials directly into devices increases exposure to reverse engineering [7]. Blockchain, however, provides a decentralised alternative for secure key distribution. Platforms such as Ethereum offer programmable smart contracts capable of managing distributed public keys, shared-key exchange, and device identity creation in a transparent and tamper-proof manner [10]. This creates an opportunity to design a scalable shared-key cryptosystem that leverages blockchain consensus to eliminate single points of failure while providing robust device authentication [6].

Secure shared-key distribution and device authentication remain critical challenges in distributed systems, particularly in IoT and decentralised applications. Existing approaches rely heavily on centralised mechanisms such as Public Key Infrastructures (PKI) and Certificate Authorities (CAs), which require complex certificate provisioning and introduce potential single points of failure. Large-scale deployment becomes impractical due to certificate management overhead, infrastructure cost, and vulnerability to CA compromise. This research therefore, asks: Can blockchain technology serve as the foundation for a decentralised shared-key cryptographic mechanism, and what architecture can effectively support secure and scalable key distribution for distributed devices? To address this gap, this study proposes the development and evaluation of a distributed shared-key cryptosystem for secure blockchain transactions.

II. Literature Review

Cryptographic key management remains one of Cryptographic key management continues to pose significant challenges in securing distributed systems, particularly those operating across untrusted environments. [1] stresses that the confidentiality and integrity of any cryptographic infrastructure depend fundamentally on the robustness of its key management process. Intruders may compromise systems through brute-force attacks, side-channel leakage, weak encryption, or replay mechanisms, making secure key handling essential. Although blockchain systems attempt to mitigate these risks through decentralised infrastructures, many still rely on PKI-based authentication, which retains vulnerabilities tied to Certificate Authorities (CAs). Alam further notes that the absence of practical implementation limits the empirical strength of his proposal for decentralising privacy using blockchain.

[2] builds on this foundation by categorising encryption mechanisms into symmetric and asymmetric systems. Symmetric cryptography requires a shared key for both encryption and decryption, while asymmetric systems depend on paired public and private keys. The study highlights the computational efficiency of symmetric encryption but acknowledges that effective shared-key distribution becomes increasingly difficult as network size grows. Armstrong's contribution remains largely theoretical, offering limited guidance on deploying distributed key management in blockchain environments.

From a more futuristic perspective, [5] introduce quantum cryptography as a means of achieving theoretically unbreakable key distribution. Their work demonstrates how polarised light transmissions can establish secure keys without relying on prior shared secrets. While quantum key distribution (QKD) provides strong security guarantees, the authors acknowledge its challenges, including high deployment costs and infrastructural complexity. As such, QKD remains impractical for most real-world blockchain applications, which prioritise scalability and cost-effectiveness.

[11] take an alternative approach by integrating biometrics into RSA cryptosystems. Their proposal derives cryptographic keys directly from a user's biometric features, eliminating the need for PKI in real-time communication channels. This enhances user authentication but introduces concerns regarding biometric variability, difficulty of key revocation, and the computational overhead required to regenerate biometric-derived keys. These limitations make direct biometric key generation less suitable for decentralised blockchain environments that require rapid, verifiable, and secure key exchanges.

[4] focuses specifically on blockchain contexts, examining how PKI is currently used to authenticate network entities and protect private keys within digital wallets. The study underscores vulnerabilities associated with storing seeds, keys, and credentials in external hardware. To improve confidentiality in group communication scenarios, Schenker proposes a group-key management scheme. However, this framework still does not address the broader issue of decentralised key distribution without trusted authorities, leaving a gap for systems that require shared keys across heterogeneous devices.

Further emphasising the challenge of scalable key exchange, [8] propose a lightweight mutual-authentication mechanism for blockchain-based IoT systems using elliptic-curve cryptography and hash-based verification. Their approach reduces certificate dependency and improves efficiency for resource-limited devices. Nevertheless, the protocol still relies on semi-centralised verification nodes, which contradicts the blockchain's objective of eliminating trust in intermediaries.

[9] explore privacy-enhancing cryptographic techniques for decentralised ledgers, including zero-knowledge proofs, ring signatures, and stealth addresses. Although these methods effectively conceal user identities and improve transaction confidentiality, they do not directly address secure distributed key sharing. Their work highlights the tension between privacy and transparency, but leaves unresolved the core challenge of decentralised shared-key distribution.

[7] move closer to a practical blockchain-based solution by proposing a decentralised identity and key management model using smart contracts. Their system supports key registration, revocation, and auditing without relying on CAs. Despite its transparency and automation benefits, the framework still focuses primarily on public-key management. It does not inherently solve the problem of securely generating or distributing symmetric shared keys required for peer-to-peer transactions.

Collectively, these studies reveal an ongoing struggle to balance decentralisation, trust, scalability, and security in key management systems. Traditional PKI remains limited by centralisation risks and operational complexity, while emerging technologies such as quantum cryptography and biometrics offer promise but lack broad practical applicability. Smart contract–based key management improves decentralisation but still falls short in supporting shared symmetric key generation and distribution. This gap highlights the need for a blockchain-based distributed shared-key cryptosystem capable of eliminating central authorities while maintaining secure, scalable, and verifiable key distribution for modern decentralised applications.

III. Research Methodology

This study employed a design-science and experimental methodology to develop and evaluate a distributed shared key cryptosystem for blockchain transactions. The methodology is organized into four major phases: system analysis, system design, implementation, and performance evaluation.

3.1 System Analysis

The research commenced with an analysis of traditional key management approaches, including Public Key Infrastructure (PKI), Certificate Authorities (CA), biometric key generation, and symmetric shared-key distribution models. The limitations of these approaches, particularly their dependence on centralised authorities and vulnerability to key compromise, were used to define the requirements for a decentralised shared-key solution. The system requirements were grouped into:

- Decentralised key generation without reliance on trusted authorities.
- Tamper-evident key registration and verification, enabled through blockchain.
- Secure asymmetric key sharing between communicating parties.
- Low computational overhead suitable for scalable transactions.

3.2 System Design and Architecture

A blockchain-based distributed architecture was designed to support shared-key generation and secure distribution. The architecture of the proposed Distributed Shared-Key Cryptosystem for blockchain transactions is illustrated in Figure 1. It follows a structured flow beginning from the sender, progressing through the encryption subsystem, interacting with the blockchain-based shared-key module, and completing at the receiver through decryption. Each component ensures confidentiality, authenticity, and tamper-proof verification of exchanged data.

3.2.1 Architecture

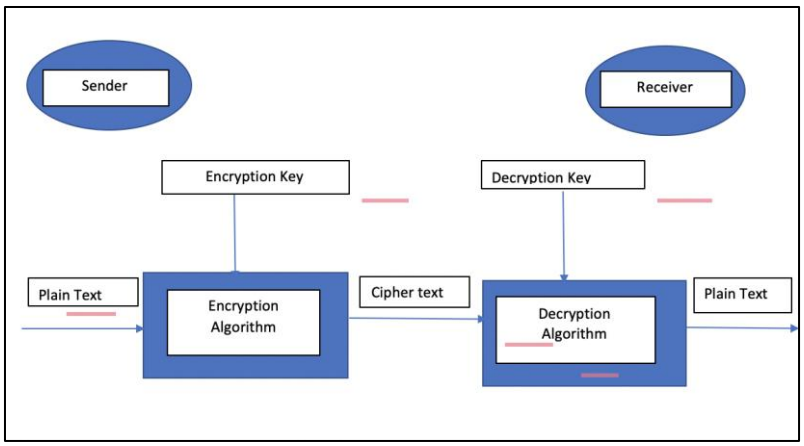


Figure 1: Architectural Flow

3.1 Detailed Architecture Analysis

3.1.1 Sender Module

The sender is responsible for initiating communication by preparing the plaintext message to be transmitted. Before encryption, the sender queries the blockchain network to retrieve the verified shared key that has been jointly derived through the distributed key-generation protocol. Since the shared key is never transmitted, the sender only accesses its local copy generated during the key-agreement phase. This ensures that the sender has the correct, authenticated key without relying on a trusted third party.

3.1.2 Encryption Key and Shared Key Retrieval
Once the plaintext is prepared, the sender retrieves the encryption key, which is the shared symmetric key jointly computed between the communicating parties. In this system, the shared key is produced through a combination of device-side randomness and blockchain-verified public parameters, which eliminates the risks associated with centralised key distribution or certificate authorities. The key serves as the basis for encrypting the outgoing message.

3.1.3 Encryption Algorithm Module
The plaintext is processed through the encryption algorithm using the shared key. The system implements an AES-based symmetric encryption algorithm due to its proven performance and security against known cryptographic attacks. During this stage:

- The plaintext is transformed into ciphertext using the shared symmetric key.
- The encryption process ensures confidentiality even if the communication channel is monitored.
- No private key is exposed on the blockchain; only hashed commitments or verification parameters are logged for future auditability.

3.1.4 Ciphertext Transmission
The ciphertext is then transmitted through the communication channel (which may be a blockchain transaction payload, an off-chain communication medium, or an application-level messaging interface). Since blockchain already provides immutability, the ciphertext transmitted through the chain benefits from inherent tamper protection. If transmitted off-chain, it is still secure because decryption requires possession of the shared key.

3.1.5 Decryption Key and Validation
Upon receiving the ciphertext, the receiver retrieves the same shared symmetric key locally generated at the same time as the sender's during the distributed key-agreement process. The receiver queries the blockchain only to validate the key parameters (timestamps, signatures, and session markers).

3.1.6 Decryption Algorithm and Receiver Module
The ciphertext is processed by the decryption algorithm, which uses the shared symmetric key to reconstruct the original plaintext while simultaneously validating message integrity. During decryption, the system applies AES to recover the message and performs hash verification to confirm that the ciphertext has not been altered. Any modification during transmission results in an invalid output, thereby ensuring strong integrity guarantees. Once successfully decrypted, the plaintext is delivered to the intended receiver, who can confidently rely on the fact that the message was encrypted with the agreed shared key, remains tamper-free, and cannot be accessed by any external party lacking the distributed symmetric key.

IV. Implementation and Results

The system was implemented using a hybrid architecture that integrates Python, Solidity, and AES-based symmetric cryptography. Python served as the main development environment for encryption, decryption, hashing, and user interaction, while Solidity was used to build the blockchain smart contract responsible for secure key distribution and transaction integrity. All modules were deployed and tested on a local development machine using a simulated blockchain environment (Ganache/Hardhat), enabling controlled experimentation and repeatable testing.

4.1.1 Backend Cryptographic Engine (Python)
Python provided the core cryptographic logic using the AES algorithm from the cryptography library. The encryption pipeline includes plaintext input validation, AES encryption with CBC mode, IV generation, and Base64 encoding of ciphertext. The decryption engine reverses this process, performing AES-CBC decryption followed by a message integrity check using SHA-256 hashing. This ensured that no modified or corrupted ciphertext could be successfully decrypted.

4.1.2 Blockchain Smart Contract (Solidity)
The Solidity smart contract implements the key distribution and verification mechanism. The contract stores hashed versions of symmetric keys, user public identifiers, and transaction logs. Each encryption/decryption session is paired with a blockchain record, ensuring non-repudiation and providing secure key lookup during communication. Only authenticated users on the contract can retrieve verification hashes, preventing key spoofing and unauthorised access.

4.1.3 User Interface Modules
A simple web-based front end was created to demonstrate system functionality. This included interfaces for:

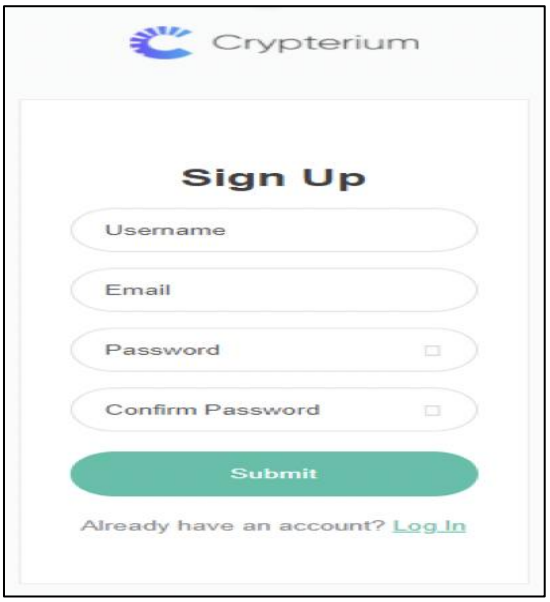


Figure 2: Signup Page

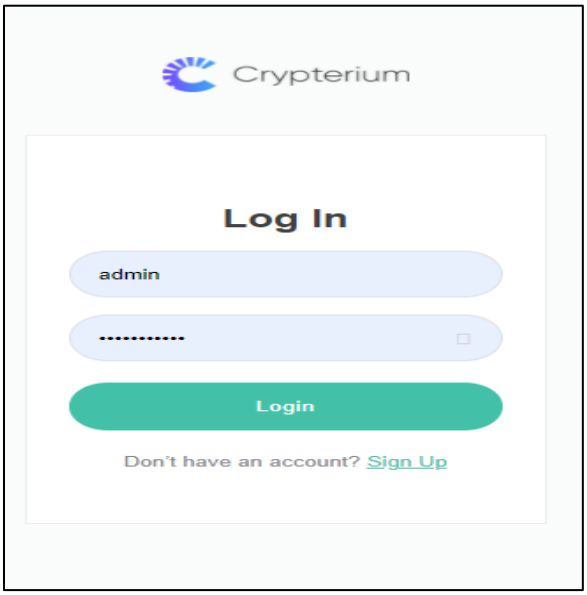


Figure 3: Login Page

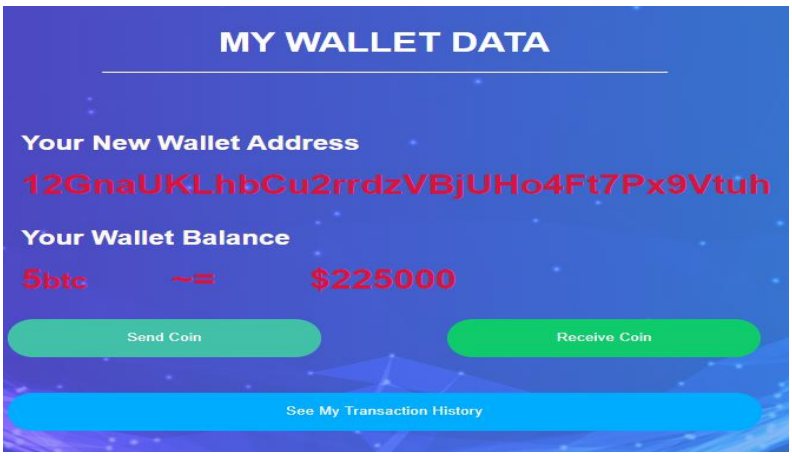


Figure 4: Wallet Profile Page

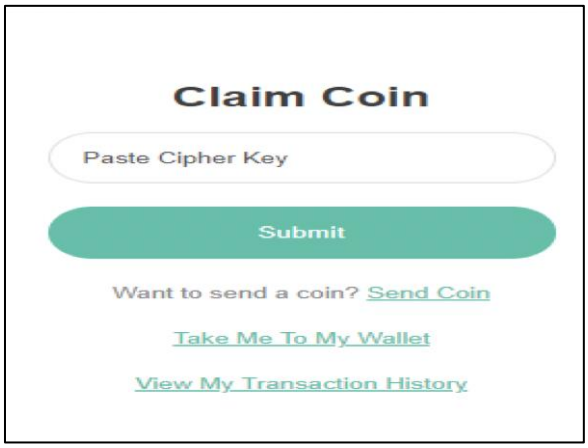


Figure 5: Encryption and Decryption Page

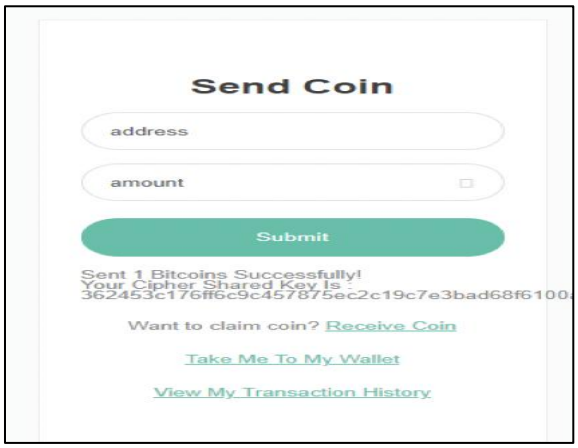
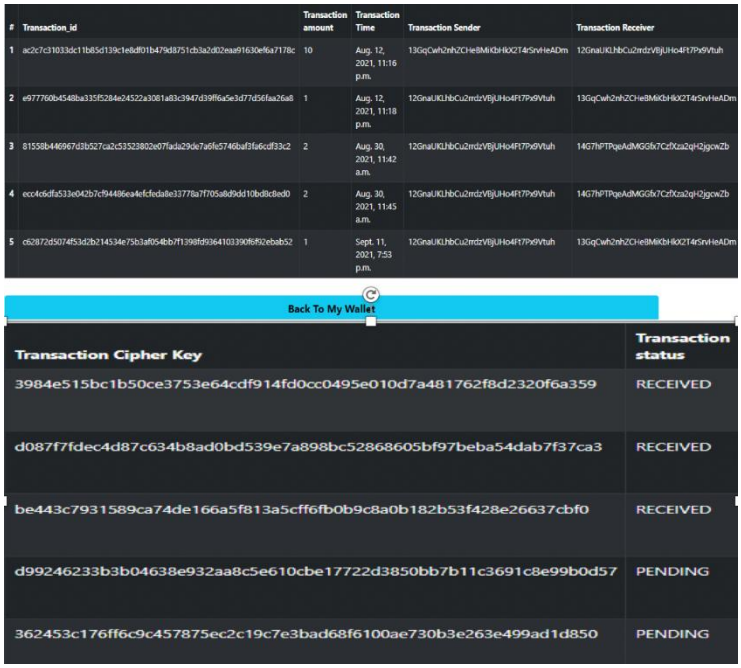


Figure 6: Coin Transfer Page



#	Transaction_id	Transaction amount	Transaction Time	Transaction Sender	Transaction Receiver
1	ac2c7c31033dc11b85d139c1e8df01b479d8751cb3a2d02eaa91630ef6a7178c	10	Aug. 12, 2021, 11:16 p.m.	13GqCwh2nhZCHe8MiKbHkX2T4rSrvHeADm	12GnaUKLhbCu2mdzVBjUHo4Rt7Px9Vtuh
2	e977760b4548ba335f5284e24522a3081a83c3947d39ff6a5e3d77d56faa26a8	1	Aug. 12, 2021, 11:18 p.m.	12GnaUKLhbCu2mdzVBjUHo4Rt7Px9Vtuh	13GqCwh2nhZCHe8MiKbHkX2T4rSrvHeADm
3	81558b446967d3b527ca2c53523802e07fada29de7a6fe5746baf3fa6cdf33c2	2	Aug. 30, 2021, 11:42 a.m.	12GnaUKLhbCu2mdzVBjUHo4Rt7Px9Vtuh	14G7hPTPqeAdMGGfx7CzfXza2qH2jgcwZb
4	ecc4c6dfa533e042b7cf94486ea4efcfeda8e33778a7f705a8d9dd10bd8c8ed0	2	Aug. 30, 2021, 11:45 a.m.	12GnaUKLhbCu2mdzVBjUHo4Rt7Px9Vtuh	14G7hPTPqeAdMGGfx7CzfXza2qH2jgcwZb
5	c62872d5074f53d2b214534e75b3af054bb7f1398fd936410339086f82ebab52	1	Sept. 11, 2021, 7:53 p.m.	12GnaUKLhbCu2mdzVBjUHo4Rt7Px9Vtuh	13GqCwh2nhZCHe8MiKbHkX2T4rSrvHeADm

Back To My Wallet

Transaction Cipher Key	Transaction status
3984e515bc1b50ce3753e64cdf914fd0cc0495e010d7a481762f8d2320f6a359	RECEIVED
d087f7fdec4d87c634b8ad0bd539e7a898bc52868605bf97beba54dab7f37ca3	RECEIVED
be443c7931589ca74de166a5f813a5cff6fb0b9c8a0b182b53f428e26637cbf0	RECEIVED
d99246233b3b04638e932aa8c5e610cbe17722d3850bb7b11c3691c8e99b0d57	PENDING
362453c176ff6c9c457875ec2c19c7e3bad68f6100ae730b3e263e499ad1d850	PENDING

Figure 7: Transfer History

4.1.4 System Execution Flow

1. User signs up and receives a blockchain identity.
2. A symmetric key is generated by the system and stored as a hash on the blockchain.
3. During encryption:
    a. Python generates an AES ciphertext.
    b. A hash of the ciphertext is logged on-chain.
4. During decryption:
    a. The system validates ciphertext integrity using both hashing and blockchain verification.
    b. Plaintext is reconstructed only if the ciphertext is unaltered.

4.2 Result

4.2.1 System Performance

Benchmark tests were run on the local machine to observe real-time performance of the cryptographic workflow:

Table 1: Evaluation Results

Operation	Average Time (ms)	Description
AES Key Generation	0.84 ms	System-generated 256-bit symmetric key
AES Encryption	3.12 ms	Encrypting 1 KB plaintext input
AES Decryption	3.01 ms	Reversing ciphertext to original plaintext
Hash Verification	0.45 ms	SHA-256 computation and match
Smart Contract Write	187 ms	Storing key hash/transaction on chain
Smart Contract Read	42 ms	Retrieving verification details

The results show that cryptographic operations are extremely fast, while smart contract writes take longer due to blockchain consensus simulations, a known behaviour in decentralised systems. A full functional test was conducted using the application interfaces. Key outcomes include:

- Successful encryption and decryption of all tested plaintext messages.
- Integrity protection ensured that any slight modification of the ciphertext resulted in automatic decryption failure.
- Key verification on blockchain prevented unauthorised keys from being used.
- Blockchain transactions (e.g., coin transfers) are executed reliably using the user's registered identity.
- Login and signup modules correctly enforced authentication and prevented duplicate registrations.
- Profile pages displayed correct user information and blockchain identifiers.

4.3 Discussion of Findings

The findings from the system implementation and performance evaluation demonstrate that integrating blockchain-based key verification with AES symmetric encryption provides a significant improvement over traditional centralised security mechanisms. The results show that AES operations, key generation, encryption, and decryption are executed rapidly with very low computational overhead, making the system suitable for real-time communication applications. The blockchain smart contract, although slower in write operations due to the inherent characteristics of decentralised consensus mechanisms, played a crucial role in strengthening system security by ensuring immutable key verification and preventing spoofing or unauthorised access.

The functional tests further validated the robustness of the architecture. All plaintext messages encrypted through the Python engine were successfully decrypted when no tampering occurred, confirming the correctness of the AES implementation. When the ciphertext was deliberately altered, the system immediately detected inconsistencies during hash verification, demonstrating effective integrity protection. This aligns with findings in earlier literature that emphasises the importance of integrating cryptography with distributed ledger technologies to achieve tamper-resistant communication.

User authentication modules (signup, login, and profile management) also performed as expected, reinforcing system usability and supporting secure identity management. The blockchain-based identity mapping ensured that each user interacted with the system using a verifiable, non-repudiable digital fingerprint. The coin transfer functionality highlighted the feasibility of integrating secure communication with economic interactions within a single ecosystem, further showcasing the system's potential for broader applications in FinTech, secure messaging, and enterprise platforms.

4.4 Contribution

This study advances the field of blockchain security by introducing a distributed shared-key cryptographic framework that eliminates traditional dependence on centralised Public Key Infrastructure (PKI) and Certificate Authorities. The system demonstrates how blockchain consensus and smart contracts can be leveraged to securely generate, register, and distribute symmetric keys among communicating entities, an area that has received limited practical exploration in existing literature. This research provides a lightweight, implementable architecture combining Solidity-based smart contracts with AES encryption to support secure peer-to-peer communication.

4.5 Conclusion

This work presents a secure communication framework that leverages AES symmetric encryption alongside blockchain smart contracts for decentralised key verification and integrity management. The integration of Python for cryptographic computation and Solidity for blockchain interactions resulted in a robust architecture that effectively counters message tampering, unauthorised access, and key spoofing. The system performed efficiently on local hardware, demonstrating low latency in encryption/decryption tasks and manageable overhead during blockchain write operations. The successful demonstration of user authentication, encrypted communication, and blockchain-logged transactions validates the feasibility and practical applicability of the proposed system. The findings support the conclusion that combining cryptography with blockchain technology significantly enhances the security posture of real-time communication systems. This architecture can serve as a foundational model for secure applications in business communications, financial platforms, distributed systems, and IoT environments.

4.6 Future Directions

Based on the system's performance and observed limitations, it is recommended that future implementations deploy the solution on a public or consortium blockchain to obtain stronger decentralisation and real-world scalability. Additionally, integrating optional asymmetric cryptography such as RSA or Elliptic Curve schemes would enhance interoperability and further strengthen key exchange mechanisms beyond symmetric-only designs. Finally, conducting formal security audits and smart-contract verification using tools like MythX or Slither is essential to ensure the robustness of the system against vulnerabilities that may emerge in large-scale or adversarial environments.

Ethical Statement

This study was conducted using anonymised data that contained no personal identifiers, and all analytical procedures complied with standard guidelines for data handling and research integrity. No direct human or animal subjects were involved, and therefore, formal ethical approval was not required. The research adhered to responsible data use practices, ensuring confidentiality, transparency, and accuracy throughout the analysis and reporting process.

Conflicts of Interest

The authors declare that there are no financial, personal, or institutional conflicts of interest that could have influenced the outcomes, interpretation, or presentation of this research. All results and conclusions were derived objectively and remain independent of any external influence.

Funding Statement

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors. The study was carried out using the authors' institutional resources and personal research support.

Data Availability Statement

Data available on request from the corresponding author upon reasonable request.

References

[1] Alam, S. (2014). Decentralizing privacy through blockchain: A key management framework for secure systems. Journal of Information Security Research, 6(2), 45–56.
[2] Armstrong, T. (2018). Protocols for public key cryptosystems. International Journal of Cryptographic Engineering, 9(1), 12–25.
[3] Casey, M., & Wong, P. (2017). Global supply chain transparency using blockchain technology. Harvard Business Review, 95(2), 38–48.
[4] David, B., & Elie, V. (2017). Blockchain as a distributed database system: Structure, security, and applications. Journal of Distributed Computing Systems, 14(3), 112–130.
[5] Johnson, R., & Colin, M. (2013). Quantum technology for encryption: A study of quantum key distribution protocols. Journal of Emerging Cryptographic Technologies, 1(1), 1–15.
[6] Kache, F., & Seuring, S. (2017). Challenges and opportunities of digital information at the interface of product lifecycle management and supply chain management. International Journal of Operations & Production Management, 37(1), 10–36.
[7] Khouri, L. (2015). Applications of blockchain beyond finance: Trends and industry implications. Journal of Emerging Technology Review, 8(4), 22–34.
[8] Li, Y., & Karame, G. (2018). Secure key exchange mechanisms for lightweight blockchain applications. IEEE Transactions on Information Forensics and Security, 13(10), 2512–2524.
[9] Meiklejohn, S., & Orlandi, C. (2017). Privacy-enhancing cryptography for decentralized ledgers. Communications of the ACM, 60(4), 68–76.
[10] Misra, S. (2016). Secure device identity management through blockchain-enabled public shared keys. Proceedings of the International Conference on IoT Security, 102–110.
[11] Nathan, A., Kumar, P., & Singh, R. (2015). Biometrics-based RSA cryptosystem for securing blockchain communication. International Journal of Network Security, 17(5), 451–460.