



# Design and Implementation of a Dual Signature Trust Model for Secure Transaction in E-Banking Systems

Okechi Onyedimekwu<sup>\*1</sup>, Moses O. Onyesolu<sup>2</sup>, Ahiakwo L. Nwabriaije<sup>3</sup>

<sup>1</sup>Department of Computer and Robotics Education,  
Federal College of Education (Technical), Omoku, Nigeria  
E-mail: okechi@fctomoku.edu.ng ORCID iD: <https://orcid.org/0000-0001-7663-784X>

<sup>2</sup>Department of Computer Science, Nnamdi Azikiwe University, Awka, Nigeria  
E-mail: mo.onyesolu@unizik.edu.ng ORCID iD: <https://orcid.org/0000-0003-3357-4847>

<sup>3</sup>Department of Computer and Robotics Education,  
Federal College of Education (Technical), Omoku, Nigeria  
E-mail: luxiluv@gmail.com

## ABSTRACT

Emerging e-banking systems require transaction-level confidentiality, integrity, authentication, and non-repudiation beyond what transport-layer security alone can guarantee. Although TLS 1.3 provides secure communication channels, it does not inherently enforce end-to-end payload confidentiality across multi-tier infrastructures, nor does it provide explicit transaction-level joint authorization. This study presents the design and implementation of a Dual Signature Trust Model integrated within SecureCipher, a secure middleware architecture that encrypts each API payload using AES-256-GCM and authenticates transactions using device-bound ECDSA P-384 digital signatures. Each transaction carries two coordinated signatures: Sig\_P, generated locally on the client device using a device-bound private key, and Sig\_S, appended by the middleware after independent verification, replay validation, and policy enforcement. Sig\_P provides cryptographic proof of origin and user authorization, ensuring message integrity before submission, while Sig\_S establishes institutional attestation and co-authorization. Ephemeral session keys are derived through ECDH over secp384r1 and expanded using HKDF-SHA384 to ensure forward secrecy. Communication channels are protected using mutual TLS 1.3 with ephemeral ECDHE cipher suites. Private keys remain confined to user devices, eliminating centralized key custody risks and aligning with zero-trust architectural principles. A prototype implementation demonstrates modest performance overhead of approximately 5% throughput reduction and 10–15% latency increase while significantly strengthening confidentiality, integrity, replay resistance, and regulatory compliance. Evaluation against open-banking frameworks, including the Central Bank of Nigeria (CBN) guidelines and PSD2 requirements, confirms regulatory alignment and enhanced transaction accountability.

**Keywords:** SecureCipher, Dual Signature Trust Model, AES-256-GCM, TLS 1.3, ECDSA, Transaction-Level Encryption

## INTRODUCTION

Modern online banking systems must reconcile usability with increasingly stringent security requirements. Traditional TLS-encrypted APIs protect data in transit but do not provide intrinsic transaction-level non-repudiation or payload confidentiality once data reaches intermediary components such as reverse proxies, load balancers, or API gateways. In many open banking deployments, messages are decrypted at multiple service layers, thereby expanding the trust boundary and increasing exposure to insider threats or infrastructure compromise. Regulatory frameworks such as the Central Bank of Nigeria's Operational Guidelines for Open Banking (Central Bank of Nigeria, 2023) and the European PSD2 directive now emphasize strong customer authentication and message-level security mechanisms. While JSON Web Encryption and JSON Web Signatures provide structured message protection, they do not inherently enforce coordinated multi-party authorization or prevent unilateral modification within distributed infrastructures.

To address this structural limitation, this study proposes a Dual Signature Trust Model implemented within SecureCipher. The conceptual foundation draws inspiration from earlier dual-signature mechanisms such as those introduced in the Secure Electronic Transaction protocol (Sarkar & Tripathi, 2014), where cryptographic binding ensured that related transaction components could not be altered independently. SecureCipher generalizes this principle beyond card-based workflows to modern API-driven e-banking systems. Rather than separating commercial and financial data domains, the model enforces coordinated cryptographic endorsement over the complete transaction payload. The client signs the encrypted transaction context using a device-bound private key, producing Sig\_P, and the middleware appends Sig\_S only after successful cryptographic validation and policy enforcement. Acceptance by the banking API requires verification of both signatures, thereby ensuring joint integrity and authorization.

Cryptographically, SecureCipher employs AES-256 in Galois/Counter Mode for payload protection, following the recommendations in NIST Special Publication 800-38D (National Institute of Standards and Technology [NIST], 2011). GCM provides authenticated encryption, guaranteeing confidentiality and integrity in a single primitive while generating an authentication tag that detects tampering. Digital signatures are implemented using ECDSA on the NIST P-384 curve in accordance with FIPS 186-4 (NIST, 2013), providing strong security strength with efficient computation. Session keys are derived from ephemeral ECDH exchanges using HKDF-SHA384 as defined in RFC 5869 (Krawczyk & Eronen, 2010), ensuring forward secrecy. The overall design aligns with Zero Trust Architecture principles (NIST, 2020), eliminating implicit trust in network boundaries and enforcing cryptographic validation at every transaction stage.

## METHODOLOGY

The methodology adopted in this study follows a design science and implementation-oriented security engineering approach grounded in formally standardized cryptographic primitives and layered trust enforcement. The objective was to construct and experimentally validate a transaction-level security framework that operates independently of, yet complementarily to, transport-layer protection. The development process began with the specification of a secure session establishment protocol between the client application and the SecureCipher middleware. At session initiation, both entities perform a mutually authenticated TLS 1.3 handshake employing ephemeral Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) key exchange over the secp384r1 curve. The use of ephemeral parameters ensures forward secrecy, such that compromise of long-term keys does not retroactively expose past session data. From the negotiated handshake secrets and associated cryptographic nonces, both parties derive application-layer symmetric keys using the HMAC-based Extract-and-Expand Key Derivation Function (HKDF) with SHA-384 as the underlying hash function (Krawczyk & Eronen, 2010). These derived keys are cryptographically independent from TLS record-layer keys and are reserved exclusively for payload-

level encryption, thereby enforcing separation of concerns between channel security and transaction-level protection.

For each API request, the client constructs the transaction payload in structured JSON format and applies authenticated encryption using AES-256 in Galois/Counter Mode in accordance with NIST Special Publication 800-38D (National Institute of Standards and Technology [NIST], 2011). To preserve the security guarantees of GCM, a unique initialization vector is deterministically generated for every message using a predefined IV base combined with a strictly monotonic counter maintained within the session context. This construction eliminates nonce reuse, which is critical for preventing forgery attacks under GCM. The encryption operation produces both ciphertext and a 128-bit authentication tag that ensures integrity and authenticity of the encrypted content. The resulting protected payload is transmitted through the established TLS channel. Even in scenarios where TLS termination occurs at reverse proxies or gateway services, the application-layer encryption preserves confidentiality and tamper resistance between the client and the SecureCipher middleware.

After encrypting the payload, the client generates a digital signature denoted as Sig\_P using a device-bound Elliptic Curve Digital Signature Algorithm key on the P-384 curve, compliant with FIPS 186-4 (NIST, 2013). The private signing key is created during user registration and securely stored within the device's protected keystore or hardware-backed enclave, ensuring that it is non-exportable and inaccessible to server infrastructure. The signature is computed over the encrypted transaction payload and associated contextual parameters, thereby cryptographically binding the transaction data to the user's device identity. This operation establishes proof of origin, message integrity, and non-repudiation at the point of submission.

The methodology further incorporates integrity-protected audit logging mechanisms within the middleware layer. Rather than storing full transaction contents, the system records metadata such as transaction digests, timestamps, and user identifiers. These logs are digitally signed to ensure tamper-evidence while minimizing exposure of sensitive financial information. Throughout the protocol lifecycle, symmetric session keys derived via HKDF remain ephemeral and confined to the session boundary, and private signing keys never leave their designated trust domains. This strict separation of cryptographic responsibilities ensures containment of compromise, reduces systemic risk associated with centralized key storage, and operationalizes zero-trust security principles within the transaction processing pipeline (NIST, 2020).

### **The Developed System**

SecureCipher model is implemented as an integrated client–middleware–bank architecture grounded in the Dual-Signature Trust Model, combining device-bound cryptographic identity with transaction-level protection. The client application (Android/iOS) generates and securely stores a non-exportable ECDSA P-384 private key within the device keystore during registration, while the corresponding public key is registered with the middleware over a TLS 1.3 channel. For each transaction, the application serializes the transaction payload, encrypts it using AES-256-GCM with session keys derived from ephemeral ECDH and HKDF-SHA384, and produces the client signature (Sig\_P) binding the entire transaction context. The middleware verifies Sig\_P, enforces replay protection and policy checks, and appends its institutional signature (Sig\_S) before forwarding the request to the banking API. This layered design ensures end-to-end confidentiality, integrity, non-repudiation, and joint cryptographic authorization without exposing private keys to server-side infrastructure



**Fig. 1.1** SecureCipher Dual Signature Trust Model Architecture

The developed system, SecureCipher model, implements a Dual Signature Trust Model (DSTM) as a cryptographically enforced middleware architecture for secure e-banking transactions. The architecture is logically divided into three principal domains: the client device, the SecureCipher middleware, and the banking API. On the client side, each user device generates and securely stores a device-bound private key, ensuring cryptographic key sovereignty at the endpoint rather than the server. For every transaction, the payload is first encrypted using an application-layer authenticated encryption scheme (AES-256-GCM) and digitally signed with the client's long-term ECDSA P-384 private key, producing the first signature (Sig\_P). This design ensures confidentiality, integrity, and non-repudiation at the transaction level— independent of transport-layer protection. Communication between the client and middleware occurs over a TLS 1.3 encrypted channel with ephemeral ECDH key exchange (secp384r1), and session keys are derived using HKDF-SHA384. While TLS protects the communication channel, the additional payload-level encryption and signature guarantee end-to-end security even in the presence of reverse proxies, intermediaries, or partially trusted infrastructure components.

Upon receipt of a transaction request, the SecureCipher middleware performs structured cryptographic validation before any forwarding operation occurs. First, it verifies the client's digital signature (Sig\_P) against the registered public key, ensuring that the transaction originates from the legitimate device-bound identity. Replay protection is enforced using per-message nonces derived from a base initialization vector combined with a monotonic counter, preventing duplication or reordering attacks. Once the client signature is validated and the payload integrity confirmed, the middleware applies a second digital signature (Sig\_S) using its own ECDSA P-384 private key. This second signature constitutes institutional attestation—confirming that the transaction has passed middleware-level validation, policy checks, and security controls. The resulting dual-signed request is then transmitted to the banking API. Unlike conventional banking systems that rely primarily on TLS and token-based authorization (e.g., OAuth or JWT), the Dual-Signature Trust Model ensures that neither the client nor the middleware can unilaterally alter transaction content without detection. The bank verifies both Sig\_P and Sig\_S before processing the transaction, establishing joint cryptographic accountability.

The core contribution of this model lies in its elimination of single-point trust dependency and its formalization of shared transaction authorization. Traditional e-banking architectures predominantly

employ single-party authorization mechanisms, where user authentication (passwords, OTP, biometrics) implicitly authorize transactions once validated by the server. In contrast, SecureCipher introduces explicit dual cryptographic endorsement at the transaction layer. This provides stronger guarantees of origin authenticity, middleware approval, tamper-evidence, and regulatory auditability. Because private keys remain device-bound and are never centrally stored, the architecture significantly reduces systemic exposure associated with centralized key management. The middleware functions not merely as a routing component but as a cryptographic co-signer and policy enforcement authority. By combining ephemeral key exchange, strong key derivation, authenticated encryption, and dual ECDSA signatures, the proposed Dual-Signature Trust Model establishes a verifiable chain of custody for each transaction. This approach enhances resilience against credential theft, insider manipulation, replay attacks, and session hijacking—thereby advancing the state of secure transaction processing in modern e-banking systems.

## RESULTS AND DISCUSSION

Prototype deployment demonstrated that AES-256-GCM encryption of a 1 kB payload required approximately 0.2 milliseconds on contemporary mobile hardware, while ECDSA P-384 signature generation required approximately 1.5 milliseconds. Middleware signature verification averaged approximately 0.5 milliseconds. Enabling transaction-level encryption and dual signatures resulted in an approximate 5% reduction in throughput and a latency increase between 10% and 15% compared with TLS-only communication. These performance impacts remained below perceptible user thresholds and are consistent with the efficiency characteristics of ECDSA and GCM implementations.

Security evaluation confirms that the model prevents replay attacks through nonce-counter enforcement and ensures tamper detection through authenticated encryption and signature validation. Because private keys are device-bound and never stored server-side, compromise of backend infrastructure does not yield signing credentials. The dual-signature requirement ensures that neither the client nor the middleware can independently alter transaction content without detection. The architecture complies with NIST cryptographic guidance, Zero Trust principles, and CBN Open Banking mandates, providing enhanced non-repudiation and regulatory auditability.

## CONCLUSION

This study has presented the design and implementation of a Dual Signature Trust Model for secure e-banking transactions within the SecureCipher architecture. By integrating AES-256-GCM authenticated encryption, device-bound ECDSA P-384 signatures, ephemeral ECDH key exchange, HKDF-SHA384 key derivation, replay protection mechanisms, and TLS 1.3 transport security, the framework enforces confidentiality, integrity, authentication, and non-repudiation at the transaction layer. Unlike conventional API security models that rely predominantly on transport-layer protection and token-based authorization, the proposed architecture introduces explicit cryptographic co-authorization between the client and middleware. Experimental results demonstrate operational feasibility with modest overhead while significantly strengthening transaction accountability and minimizing centralized trust exposure. Future work will focus on formal security proofs under IND-CCA2 and EUF-CMA models, scalability testing under high-throughput conditions, and exploration of threshold or post-quantum signature extensions to enhance long-term resilience.

## REFERENCES

- Al-Zubaidie, M., Zhang, Z., & Zhang, J. (2019). Efficient and secure ECDSA algorithm and its applications: A survey. *International Journal of Communication Networks and Information Security*, 11(1), 7–35. [https://research.usq.edu.au/download/fc0a146c1dd9d6da09882b6ccfc134e6c84d5402d9b21d404e1fd291d0afc439/615211/ECDSA\\_survey.pdf](https://research.usq.edu.au/download/fc0a146c1dd9d6da09882b6ccfc134e6c84d5402d9b21d404e1fd291d0afc439/615211/ECDSA_survey.pdf)
- Central Bank of Nigeria. (2023). *Operational guidelines for open banking in Nigeria*. <https://www.cbn.gov.ng>
- Genc, Y., & Afacan, E. (2021). Design and implementation of an efficient elliptic curve digital signature algorithm (ECDSA). In 2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS). <https://doi.org/10.1109/IEMTRONICS52119.2021.9422589>

- Gidney, C., & Ekerå, M. (2021). How to factor 2048-bit RSA integers in 8 hours using 20 million noisy qubits. *Quantum*, 5, 433. <https://doi.org/10.22331/q-2021-04-15-433>
- Gour, A., Singh Malhi, S., Singh, G., & Kaur, G. (2024). Hybrid cryptographic approach for secure data communication using block cipher techniques. *E3S Web of Conferences*, 556, 01048. <https://doi.org/10.1051/e3sconf/202455601048>
- Jamroz, Z., Ullah, I., Hassan, B., Ul Amin, N., Khan, M. A., Lorenz, P., & Innab, N. (2023). An optimal authentication scheme through dual signature for the Internet of Medical Things. *Future Internet*, 15(8), 258. <https://doi.org/10.3390/fi15080258>
- Krawczyk, H., & Eronen, P. (2010). *HMAC-based extract-and-expand key derivation function (HKDF) (RFC 5869)*. Internet Engineering Task Force.
- National Institute of Standards and Technology. (2011). *Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC (SP 800-38D)*.
- National Institute of Standards and Technology. (2013). *Digital signature standard (DSS) (FIPS PUB 186-4)*.
- National Institute of Standards and Technology. (2020). *Zero trust architecture (SP 800-207)*.
- Sarkar, A., & Tripathi, S. (2014). Design of a dual signature scheme using ECDSA in SET protocol. *International Journal of Computer Applications*, 101(13), 1–7.
- Upadhyay, S. (2025). Architecting secure, scalable, and real-time transaction systems. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 11(1), 2322–2332.
- Visa Developer. (n.d.). Message Level Encryption. Retrieved from [https://developer.visa.com/pages/encryption\\_guide](https://developer.visa.com/pages/encryption_guide).

#### Authors' Profiles



Okechi Onyedimekwu is a Computer Science Lecturer and Examination Officer at the Federal College of Education (Technical), Omoku, Nigeria. He holds a B.Eng. in Elect/Elect Engineering (IT Options) from ABU, Zaria, PGDE, M.Sc. in Information Technology, and is about concluding his Ph.D. in Computer Science (Software Engineering) at Nnamdi Azikiwe University, Awka, Nigeria. A member of Computer Professionals Registration Council of Nigeria (CPN) and Teachers Registration Council of Nigeria (TRCN). His expertise is in Software Engineering Design & Modeling, Data Analytics and Applied Cryptography.



Moses Okechukwu Onyesolu, Professor, Department of Computer Science, Nnamdi Azikiwe University, Awka-Nigeria. His research interest is mainly in software engineering which revolves around modeling/simulation, computer security, and artificial intelligence. He is a member Nigerian Computer Society (NCS), Computer Professionals (Registration Council of Nigeria) (CPN), and International Association of Engineers (IAENG), International Association of Computer Science and Information Technology (IACSIT) and European Association for Programming Languages and Systems (EAPLS).



Ahiakwo Lucky Nwabriaije is a Senior Lecturer and the current Academic Advisor in the Department of Computer & Robotics Science, Federal College of Education (Technical), Omoku, Rivers State. He holds a Bachelors and Masters Degrees in Automation and Control Systems Engineering (Computer Engineering Option) of the Faculty of Computer Engineering, Kuban State University of Technology, Krasnodar, Russia Federation. He also obtained a Masters Degree in Information and Telecommunications Engineering at the Center for Information and Telecommunications Engineering, University of Port Harcourt, Choba, Rivers; where he is also currently a Doctorate Degree student in Information Systems Engineering. He is a member of several professional bodies.