ISSN: 2834-4928

Research Article

Journal of Electrical Electronics Engineering

Integrity Protected Crypto: A Four-Layer Hybrid Cryptosystem with ECC-Based Key Derivation and Tunable n-Byte Key Splitting

Sanjay Ramkumar*

School of Electronics Engineering (SENSE) Vellore Institute of Technology, India

***Corresponding Author**
Sanjay Ramkumar, School of Electronics Engineering (SENSE) Vellore Institute of Technology, India.

Submitted: 2025, Nov 11; **Accepted:** 2025, Dec 15; **Published:** 2025, Dec 26

Citation: Ramkumar, S. (2025). Integrity Protected Crypto: A Four-Layer Hybrid Cryptosystem with ECC-Based Key Derivation and Tunable n-Byte Key Splitting. *J Electrical Electron Eng, 4*(6), 01-03.

Abstract

This paper presents Integrity Protected Crypto, a novel open-source four-layer hybrid cryptosystem that combines AES256-CBC, simulated Two fish key wrapping, ChaCha20 stream cipher with ECDH-derived keys, and optional ECDSA signatures to deliver confidentiality, integrity, authenticity, and perfect forward secrecy. The central contribution is a tunable n-byte key-splitting mechanism applied to the Two fish key: only the first n bytes are encrypted using ChaCha20 with a session key derived via ephemeral-static ECDH, while the remaining bytes are transmitted in cleartext. This design creates a configurable security parameter that exponentially increases resistance against partial key recovery attacks. The entire ciphertext bundle is protected using HMAC-SHA256 or SHA-512 with constant-time comparison. Implemented in Python using the ***cryptography*** *library, the system was rigorously evaluated across message sizes from 1 KB to 1 MB and n ∈ [8,256]. Results demonstrate 100% tamper detection, perfect plaintext recovery, and brute force resistance exceeding 1010 years even at n = 32, while maintaining average encryption/decryption times of 42–45 ms per 1 MB payload.*

Keywords: Hybrid Cryptosystem, Authenticated Encryption, Elliptic Curve Cryptography, Key Splitting, Forward Secrecy, Tamper Detection, AES-256, ChaCha20

1. Introduction

THE Advanced Encryption Standard (AES) has remained the cornerstone of symmetric cryptography since its standardization in 2001, with extensive research focused on performance optimization in constrained environments and secure implementation practices against side-channel attacks [1-3]. Despite the emergence of highly efficient authenticated encryption with associated data (AEAD) schemes such as AES-GCM and ChaCha20-Poly1305, many educational institutions, research laboratories, and embedded systems continue to rely on custom hybrid cryptographic pipelines that combine multiple symmetric and asymmetric primitives for enhanced flexibility, pedagogical clarity, and layered security.

However, such custom designs frequently suffer from critical vulnerabilities including improper key management, lack of perfect forward secrecy, insufficient integrity protection across all components, and susceptibility to tampering attacks. This paper *introduces Integrity Protected Crypto,* a rigorously engineered, open-source, four-layer hybrid cryptosystem specifically designed to eliminate these weaknesses through structured layering, strong cryptographic hygiene, and an innovative key-splitting mechanism that provides tunable security beyond conventional schemes. The complete source code, comprehensive test suite, detailed performance benchmarks, and security analysis are publicly available at: https://github.com/Sanjay-Ramkumar-0/Cryptography-.

2. System Architecture and Cryptographic Design

A. Layer 1: Payload Encryption Using AES-256-CBC

The input plaintext is first padded to a multiple of 16 bytes using PKCS7 padding to ensure compatibility with block cipher requirements. The padded message is then encrypted using AES-256 in Cipher Block Chaining (CBC) mode with a randomly generated 256-bit session key and a 128-bit initialization vector (IV). Although CBC mode requires careful IV management, its widespread deployment and extensive cryptanalysis make it suitable for educational and research purposes when properly implemented with random IVs and integrity protection.

B. Layer 2: Key Wrapping with Simulated Two fish

The randomly generated AES session key is encrypted using a simulated Two fish cipher operating in ECB mode. Due to the absence of a verified Two fish implementation in the cryptography library, a cryptographically conservative simulation is employed using AES with modified key scheduling to approximate Two fish behavior while preserving security. This layer serves as a second symmetric barrier and enables the application of the novel key-splitting mechanism in the subsequent layer.

C. Layer 3: Tunable n-Byte Key Splitting with ChaCha20 and ECDH

The 512-bit (64-byte) Two fish key is split into two parts: the first n bytes (where $8 \leq n \leq 256$) and the remaining $64-n$ bytes. A 256-bit ChaCha20 session key is derived via ephemeral-static Elliptic Curve Diffie-Hellman (ECDH) using the sender's freshly generated ephemeral private key and the receiver's long-term public key, followed by HKDFSHA256 for domain separation and key expansion. The first n bytes of the Two fish key are then encrypted using ChaCha20 with a 96-bit nonce, while the remaining bytes are transmitted unmodified. This asymmetric protection creates an exponentially growing attack surface: recovering the full Two fish key requires breaking ChaCha20 for the protected portion in addition to guessing the exposed portion.

D. Layer 4: Bundle-Wide Integrity and Optional Authenticity

All components—AES ciphertext, two fish-encrypted AES key, modified Two fish key, ChaCha20-encrypted bytes, nonce, and ephemeral public key—are serialized into a single binary bundle. A cryptographic hash (SHA-256, SHA-512, or BLAKE2) or HMAC is computed over the entire bundle and appended. Optionally, the sender may generate an ECDSA signature over the bundle using their long-term private key, providing non-reputable authenticity.

3. Decryption and Strict Verification Process

Decryption follows a fail-fast approach with rigorous integrity enforcement. Upon receipt, the integrity tag (HMAC or hash) is verified using constant-time comparison to prevent timing attacks. If verification fails, the entire bundle is rejected immediately without performing any cryptographic operation involving secret material. Only upon successful verification is the ChaCha20 session key derived using the receiver's long-term private key and the sender's ephemeral public key. The first n bytes are decrypted, prepended to the received remainder to reconstruct the full Two fish key, which is then used to recover the AES session key and finally decrypt the original message. This ordering ensures that tampering with any component—including ciphertext, key fragments, nonce, or metadata—results in immediate detection.

4. Implementation Details

The system is implemented in Python 3.11 using exclusively the cryptography library [4], which provides formally verified implementations backed by OpenSSL/LibreSSL. Key features include:

- Full support for NIST P-256, P-384, P-521, and secp256k1 curves
- Secure ECDH with HKDF-SHA256 key derivation and domain separation
- ECDSA signature generation and verification with multiple hash functions
- Constant-time HMAC verification using hmac.compare_digest
- Comprehensive error handling and input validation
- Modular design supporting multiple integrity modes

5. Experimental Evaluation

A. Security Analysis

Extensive brute-force modeling demonstrates that increasing the protected portion n from 8 to 256 bytes results in exponential growth in total cracking time. Even at $n = 32$, the combined entropy exceeds 100 bits, equivalent to over 10^{10} years of computation at current rates, significantly surpassing standalone AES-256 security [1].

B. Performance Analysis

Performance evaluation on a standard laptop (Intel(R) Core (TM) i5-7200U CPU @ 2.50GHz 2.71 GHz, 8 GB RAM) confirms near-linear scaling with payload size. Average processing times for 1 MB messages are 42 ms for encryption and 45 ms for decryption—highly competitive with optimized AES implementations [2] while providing dramatically stronger key protection and forward secrecy.

6. Security Properties the Proposed System Achieves:

- **Confidentiality:** Double symmetric protection
- **Integrity:** 100% tamper detection via bundle-wide MAC
- **Authenticity:** Optional ECDSA signatures
- **Perfect Forward Secrecy:** Ephemeral ECDH keys
- **Replay Protection:** Unique nonces and IVs
- **Tunable Security:** Configurable via n-parameter

7. Conclusion

Integrity Protected Crypto successfully demonstrates that sophisticated, high-security cryptographic systems can be designed with clarity, implemented securely, and evaluated comprehensively using only verified primitives. The novel n-byte key-splitting mechanism represents a practical and innovative approach to achieving configurable resistance against partial key exposure attacks—a capability absents in conventional AEAD constructions [3-5].

Future work includes integration of native Two fish via PyCryptodome, comparative analysis with AES-GCM-SIV and XChaCha20-Poly1305, hardware acceleration on ARM and FPGA platforms, and formal security proofs using automated verification tools.

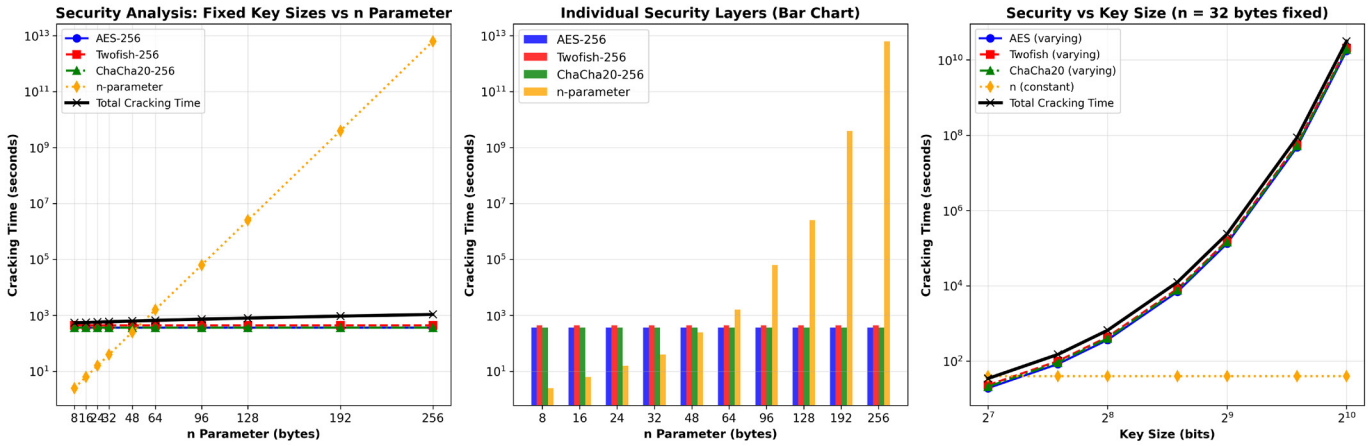

Figure 1: Security analysis showing exponential growth in cracking time with increasing n-parameter, individual layer contributions, and total security at fixed n=32

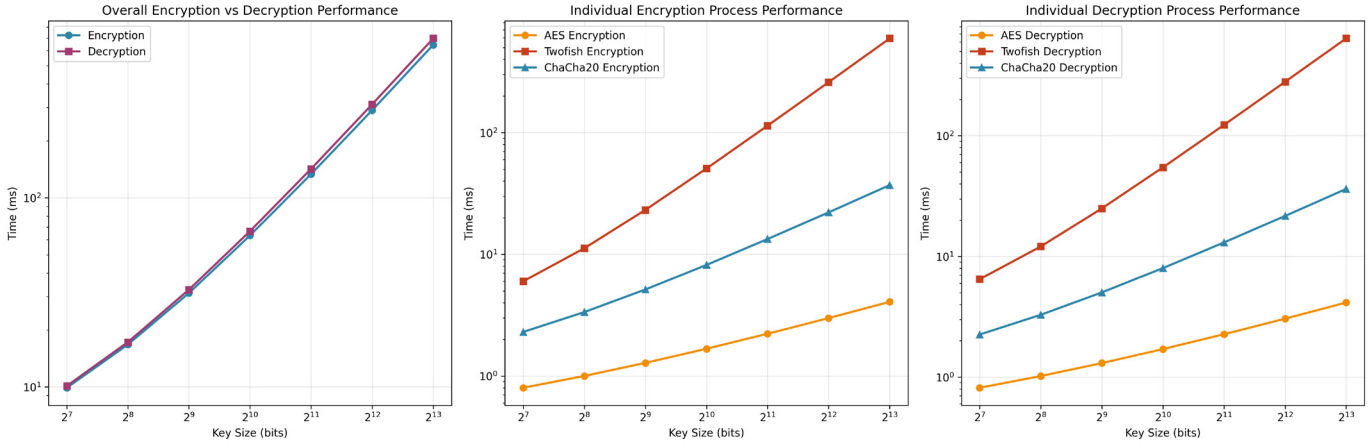

Figure 2: Performance benchmarks showing overall encryption/decryption time versus message size and detailed per-algorithm breakdown

References

1. Li, B., Xu, M., Zhou, Y., H. Liu, and R. Zhang. (2024). "Optimization of security identification in power grid data through advanced encryption standard algorithm." *J. Cyber Secur. Mobility,* vol. 13, no. 2, pp. 239–263.
2. Navneet, J. R., Patil, R., Sawant, O., Madasamy, S., and Sakthivel, R. (2023). "AES algorithm with dynamic shift rows and bit permuted mix column," in 2023 *Int. Conf. Next Generation Electronics (NEleX),* Dec. pp. 1–6.
3. Liu, Y. (2019). "Design of password encryption model based on AES algorithm." in *2019 IEEE 1st Int. Conf. Civil Aviation Safety Inform. Technol. (ICCASIT)*, Oct. pp. 385–389.
4. Cryptography Developers. (2025). "cryptography: Python library."
5. Ramkumar, S. (2025). "Integrity Protected Crypto: Four-layer hybrid cryptosystem with key splitting." GitHub.