Grenze International Journal of Engineering and Technology, January Issue

A Unified Approach to Vulnerability Mitigation and Security Enhancement in ECDH-AES-GCM End-to-End Communication

Rajnish Tiwari[1], Prashant Kumar[2], Asmit Kumar Roy[3] and Anand Bhushan Pandey[4]
[1-4]Computer Science Engineering, Jims Engineering Management Technical Campus (JEMTEC) Noida, India
Email: rajnishtiwari89797@gmail.com, Prashant.aa04@gmail.com, royasmit345@gmail.com, pandey.mymails@gmail.com

***Abstract*— End-to-end communication protocols increasingly rely on elliptic curve Diffie–Hellman (ECDH) combined with AES-GCM for secure key establishment and data protection. While this pairing offers efficiency and strong theoretical security, practical deployments reveal persistent vulnerabilities including unauthenticated ephemeral keys, nonce/IV reuse, naïve key derivation, and weak public-key validation. These flaws enable man-in-the-middle (MITM) attacks, plaintext recovery, and session compromise in real-world systems. This work, analyze these vulnerabilities and propose a unified, implementation-friendly enhancement to the ECDH–AES-GCM framework. The protocol introduces identity-bound authentication of ephemeral keys using static digital signatures (SIGMA-style), HKDF-based key separation with explicit context binding, deterministic nonce derivation via nonceBase ⊕ seq, and strict public-key validation checks. Additionally, optional key confirmation is incorporated to ensure both parties derive identical session keys. Security analysis shows resilience against MITM, replay, and invalid-curve attacks, while preserving forward secrecy. Experimental evaluation demonstrates that the proposed enhancements introduce only minimal computational overhead compared to vanilla ECDH–AES-GCM yet significantly strengthen protocol robustness for browser and constrained environments. This work bridges the gap between theoretical cryptographic guarantees and practical implementation challenges, offering a secure and efficient framework for modern end-to-end encrypted communication.**

***Index Terms*— Elliptic Curve Diffie–Hellman (ECDH), AESGCM, Key Derivation Function (HKDF), Deterministic Nonce, Ephemeral Key Authentication, End-to-End Encryption, Man in the-Middle (MITM) Defense, Public-Key Validation.**

I. INTRODUCTION

The reliability of today's digital networks depends on secure communication protocols, which ensure privacy, data integrity, and identity confirmation across online services, mobile systems, and Internet of Things (IoT) devices. The Diffie– Hellman (DH) protocol [1] introduced an innovative way to create a shared key over an unsecured link without prior setup, forming the basis for symmetric encryption. Yet, it remains vulnerable to man-in-the-middle (MITM) attacks [2], where an attacker can intercept and alter the exchange. To boost security and efficiency, elliptic curve cryptography (ECC) emerged in the 1980s [3,4], using the elliptic curve discrete logarithm problem (ECDLP) for strong protection with smaller keys, leading to elliptic curve Diffie–Hellman (ECDH).

Grenze ID: 01.GIJET.12.1.131

This is widely used in Transport Layer Security (TLS), Virtual Private Networks (VPNs), and secure messaging due to its reduced processing needs [5].For data protection and authenticity, the Advanced Encryption Standard in Galois/Counter Mode (AESGCM) delivers an efficient authenticated encryption with associated data (AEAD) system [6]. However, its security hinges on unique nonces (initialization vectors, IVs); reusing a nonce with the same key can cause breaches like plaintext exposure or data tampering [7,8]. AES-GCM-SIV [9] addresses this by securing against nonce reuse. Key derivation has also advanced, with the HMAC-based Key Derivation Function (HKDF) [10,11] generating robust keys from shared secrets for context-specific use, as seen in TLS 1.3 [12] and Hybrid Public Key Encryption (HPKE) [13], reducing risks of key mismanagement [14,15].Authenticating ephemeral keys is key to thwarting MITM attacks and ensuring session reliability, achieved through protocols like SIGMA [16], Noise [17], EDHOC [18], and X3DH [19] using signatures to link temporary and long-term keys. Public-key validation counters threats from malformed curve points [20] and small-subgroup weaknesses [21] in ECDH. The field also tackles new challenges, with Hybrid Obfuscated Key Exchange and Key Encapsulation Mechanisms (KEMs) [22] defending against quantum risks, and GCM improvements with longer nonces [23] solving practical issues. Despite progress, ECDH–AESGCM implementations often lack a cohesive strategy integrating authenticated keys, nonce resilience, HKDF-based derivation, and strict validation, leaving systems open to MITM, replay attacks, and protocol conflicts [24]. This is critical in resource-limited settings like browsers or IoT devices, where failures can expose flaws, and weak key practices may enable downgrade attacks or session hijacks, undermining end-to-end encryption.

Despite progress, current ECDH→AES-GCM deployments lack a unified framework ensuring: (1) authenticated ephemeral keys, (2) nonce-misuse resilience, (3) HKDF-based context bound key derivation, and (4) strict key validation. This paper presents an enhanced protocol integrating signed ephemeral keys, HKDF scheduling, deterministic IVs, and rigorous public-key checks, compatible with browsers and constrained devices, addressing long-standing weaknesses in existing systems.

II. Related Work

The development of secure communication protocols stems from pioneering work in key exchange and encryption, with ongoing efforts tackling vulnerabilities and adapting to new challenges. Diffie and Hellman [1] introduced the Diffie–Hellman (DH) protocol, enabling a shared secret via g^ab mod" " p, where gis a generator, pis a prime, and a,bare private exponents, forming K=g^ab mod" " p. However, Merkle [2] noted its susceptibility to man-in-the-middle (MITM) attacks due to unverified exchanges. To improve security and efficiency, Koblitz [3] and Miller [4] proposed elliptic curve cryptography (ECC), leading to elliptic curve Diffie–Hellman (ECDH) with kPon curve Eover F_q[5], widely adopted in TLS, VPNs, and messaging as per Menezes et al. [5].

For encryption, AES-GCM [6] offers an authenticated encryption with associated data (AEAD) scheme, computed as C=E_k (P⊕CTR)⊕H(IV,A), where Cis ciphertext, Pis plaintext, and His a hash. Yet, nonce reuse [7,8] allows plaintext recovery via C_1⊕C_2=P_1⊕P_2, addressed by AES-GCM-SIV [9] for misuse resistance. Key derivation advanced with HKDF [10,11], using K="Extract"(S,"salt")‖"Expand"(PRK,"info")for secure keys in TLS 1.3 [12] and HPKE [13], though flaws like cross-protocol issues persist [14,15].

Authentication is crucial, with Krawczyk [16]'s SIGMA using signatures, extended by Perrin [17]'s Noise, Selander et al. [18]'s EDHOC for IoT, and Marlinspike and Perrin [19]'s X3DH for messaging. Public-key validation counters invalid curve attacks (P∉E(F_q)) [20] and ECDH failures [21]. Emerging threats drive innovations like hybrid KEMs [22] against quantum risks and GCM enhancements with longer nonces [23]. Despite this, ECDH–AES-GCM lacks a unified framework [24], leaving gaps in authentication and nonce handling, especially in resource-constrained browsers and IoT, where our work aims to improve.

III. Methodology

This section presents the proposed security enhancement framework for the ECDH–AES-GCM protocol, focusing on resolving vulnerabilities in key authentication, derivation, and nonce handling. The methodology defines a structured flow comprising secure key generation, authenticated handshake, HKDF-based key expansion, and deterministic IV computation, ensuring confidentiality, integrity, and forward secrecy in end-to-end communication.

Methodology Flow

- Key Generation
- Authentication

- Share Secret Derivation
- HKDF-Based Key Expansion
- Deterministic Nonce Computation
- Secure Message Transmission

A. Design goals and assumptions

TABLE I. DESIGN GOALS AND ASSUMPTIONS

Design goals	Security assumptions
Confidentiality, integrity, authenticity of messages.	ECDLP / ECDH hardness for chosen EC curves (P-256 / P-384 / X25519 as implementation choice).
Forward secrecy and (practical) post-compromise recovery.	HMAC-SHA-256 is a secure PRF.
Strong key separation and context binding.	AES-GCM is secure under unique-IV assumption.
Deterministic, replay-safe IV discipline suitable for browsers and multi-process clients.	Signature scheme (ECDSA or Ed25519) is EUF-CMA secure.
Interoperability with WebCrypto API and reasonable performance on mobile/IoT.	Local storage (IndexedDB, secure enclave) available for persistent monotonic counters/keys (subject to device compromise caveat).

Each session performs an authenticated ECDH handshake where ephemeral ECDH public keys are signed by long-term identity keys (SIGMA-style). The raw ECDH shared secret is processed via HKDF-Extract (with per-session salt) and HKDF-Expand with explicit info labels to produce a family of independent subkeys: Kenc, Kmac, Knoncebase, Kconf, and rekey seeds. For each message we derive a deterministic IV = nonceBase XOR encode(seq) where seq is a monotonic per-direction counter persisted robustly; encryption uses AES-GCM with the derived Kenc and associated data bound to session/context; an optional HMAC over the transcript (using Kmac) provides explicit key confirmation and additional integrity. Rekeying (DH or HKDF-based) is performed on configurable thresholds (message count, time, or data volume). Public keys are strictly validated (point format, on-curve checks), and all labels are canonicalized to avoid cross-protocol confusion.

TABLE II. NOTATIONS AND CONSTANTS

Notation / Constant	**Description**
G	EC base point (curve: P-256 / P-384 / X25519)
d_x	Private key of party X
$Q_x = d_x G$	Public key of party X
S	ECDH shared secret (raw point or x-coordinate)
HKDF(·)	HMAC-based Extract→Expand (HKDF-SHA256)
HKDF_Extract(salt, IKM)	Produces PRK
HKDF_Expand(PRK, info, L)	Produces OKM (L bytes)
K_{enc}	AES-256 key (32 bytes)
K_{mac}	HMAC-SHA-256 key (32 bytes) (optional extra MAC)
nonceBase	12-byte base nonce from HKDF
seq	Monotonic per-direction counter (uint64 or uint96)
IV = nonceBase ⊕ pad12(seq)	Initialization vector
AAD	Associated Auth Data (sessionID || senderID || seq || protocolVersion)
Tag_GCM	AES-GCM 128-bit tag
Sig	Signature over ephemeral public key (signature scheme = Ed25519/ECDSA)

B. Handshake

Precondition: Each party has a long-term identity key pair (sk_{id},pk_{id}) registered/obtained via PKI / TOFU / QR code. Session has a unique sessionId (random 32 bytes).

Ephemeral key generation (both sides)

- Alice: generate ephemeral key pair

(d_A^e , Q_A^e) - ECDH ephemeral.

- Bob: generate ephemeral key pair (dBe , QBe)

Sign ephemeral public keys

- Alice forms message M_A = concat("ephemeral",Q_A^e , sessionID, timestamp).
- Alice computes $\sigma_A = Sign(sk_{id,A}, H(M_A))$.
- Send to Bob: (Q_A^e , $pk_{id,B}$, σ_A)

Receiver verifies

- Bob verifies the static public key $pk_{id,A}$ via PKI/TOFU.
- Verify σ_A over M_A . If verification fails, abort.

Symmetric for Bob

- Bob sends (Q_B^e , $pk_{id,B}$, σ_B) to Alice; Alice verifies.

Compute raw ECDH shared secret

- Alice: compute S= KDFInput(ECDH(d_A^e , Q_A^e)). Use x-coordinate or canonical encoding (per curve) as IKM.
- Bob: compute same S.

Key extraction (HKDF-Extract)

Generate random salt (per-session random 32 bytes). Optionally include both identity public keys and ephemeral public keys in salt or info.

PRK = HKDF_Extract(salt,S).

Key expansion (HKDF-Expand)

Derive subkeys:

- K_{enc} = HKDF_Expand(PRK,info = "V1|enc|Alice->Bob|" sessionID,32).

K_{enc_rev} = HKDF_Expand(PRK,info = "V1|enc|Bob->Alice|"sessionId,32).

K_{mac} = HKDF_Expand(PRK,info = "V1|mac|"sessionID,32).

nonceBase = HKDF_Expand(PRK,info ="v1|nonceBase|",sessionID,12).

K_{conf} = HKDF_Expand(PRK,info="v1|conf|"sessionID,32).

Optionally: rekeySeed = HKDF_Expand(PRK,info= "v1|rekey|"sessionID,32).

Key confirmation

Alice computes tag_A = $HMAC_{K_{conf}}$ (transcript), sends to Bob after first message (or within handshake). Bob verifies to *confirm both derived same keys*. This defends against active attack truncation or handshake manipulation.

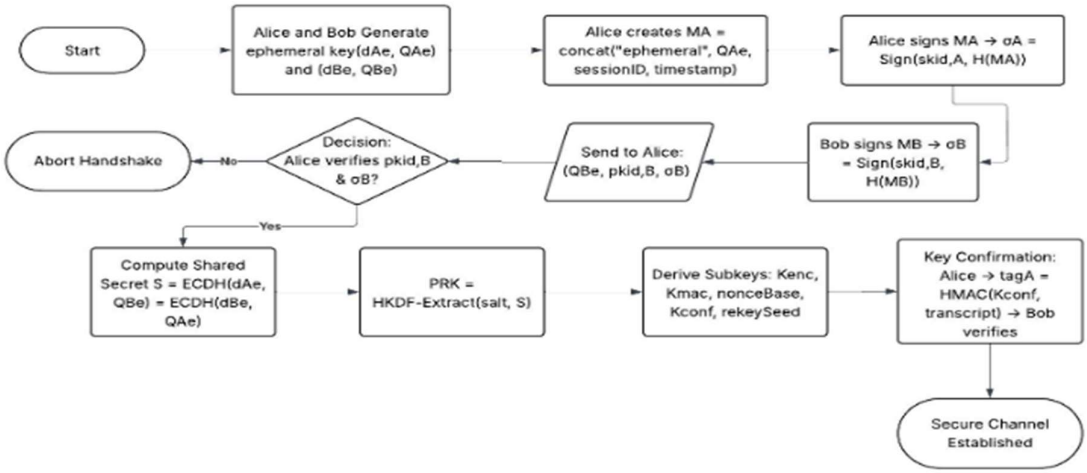

Figure 1: handshake flow

C. Message format & per-message operations

Per-direction seq counters

- Maintain seq_send and seq_recv per session/direction. Persist seq_send after sending and receiving ack; persist seq_recv on receiving a valid message (to avoid reuse after crash).

IV derivation

- Let seq be 64-bit counter. We encode it to 12 bytes: pad = zeroes(12-8) || be64(seq); then IV=nonceBase $\oplus$ pad

This yields deterministic, unique IV per message.

D. Encryption (sender)

Compute AAD = sessionID || senderID || seq || protocolVersion || other metadata.
IV as above.
C ‖ Tag_{GCM} = AES_GCM_Enc(K_{enc},IV,P,AAD).
Comput extTag=HMAC_{K_{mac}}{message}(sessionID || senderID || seq || IV || C || Tag_{GCM})
— *external tag* for redundancy & key confirmation.
Packet = {header: {sessionID, senderID, seq, pk_id_sender (maybe omitted)}, IV (optional), C, Tag_GCM, extTag}.
Increment and persist seq_send.

E. Decryption (receiver)

Check header: sessionID, senderID.
Verify seq > last_seq_recv (or within reorder window).
Recompute IV from nonceBase and seq. If size mismatch, abort.
Verify extTag with K_{{mac}}. If fails, drop and log.
Run AES_GCM_Dec. If GCM tag invalid, drop and log.
Update last_seq_recv persistently.

F. Public key validation & format handling

For raw (uncompressed) EC points: require canonical 0x04 prefix and exact expected byte-length (65 for P-256).
Perform on-curve check: given Q = (x,y), verify $y^3 \equiv x^3 + ax+b \pmod p$. For X25519, use library validation and scalar masking rules.
Reject small-order or identity points. If invalid → abort handshake and log.
Key confirmation & explicit transcript binding
Why: Prevent truncation, downgrade, or mismatch attack where one party thinks handshake complete and other does not
How: Use K_{conf} to compute:
KC_A = HMACK_{conf} (transcript||"client")
KC_B = HMACK_{conf} (transcript||"server")
One party sends KC after handshake; other verifies. Optionally use two-way confirm.
transcript = canonical concat of: sessionID || pk_id_A || pk_eph_A || pk_id_B || pk_eph_B || salt || timestamp

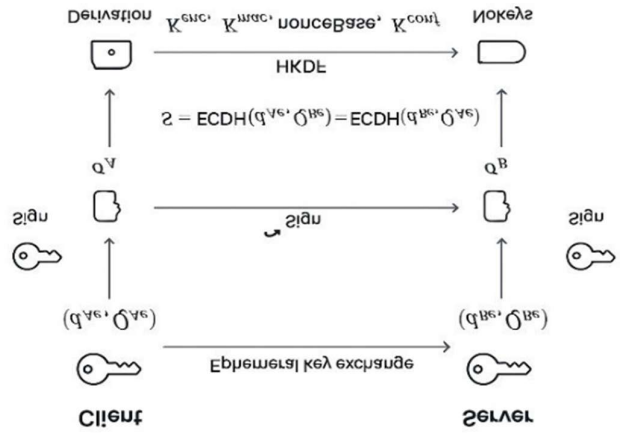

Figure 2: handshake + key derivation flow

The figure (2) above illustrates secure handshake flow of the proposed protocol. Both client and server generate ephemeral ECDH key pairs (d_{AE},Q_{AE}) and (d_{Be},Q_{BE}). Each party signs its ephemeral public key with its static identity key, producing signatures σ_A and σ_B to prevent MITM substitution. After exchanging and verifying signatures, both compute the shared secret S=ECDH(d_{AE},Q_{BE}) =(d_{BE},Q_{AE}). Using HKDF, multiple context-bound subkeys (K_{enc}, K_{mac}, nonceBase, K_{conf}) are derived, ensuring key separation and nonce discipline for subsequent AES-GCM encryption.

G. Implementaion Code

Handshake

```
dAe, QAe = GenerateEphemeral()
M = CONCAT("ephemeral", QAe, sessionID, now)
sigma = Sign(sk_id_A, Hash(M))
SendTo(Bob, QAe, pk_id_A, sigma)
Receive QBe, pk_id_B, sigmaB
Verify(pk_id_B) // PKI / TOFU
VerifySignature(pk_id_B, Hash(CONCAT("ephemeral", QBe, sessionID, tB)), sigmaB)
S = ECDH(dAe, QBe)
salt = Random(32)
PRK = HKDF_Extract(salt, S)
```

Sending message

```
seq = LoadAndIncrementSeq_send()
IV = XOR(nonceBase, EncodeSeq12(seq))
AAD = CONCAT(sessionID, senderID, seq, protoVer)
C, tag = AES_GCM_Enc(Kenc, IV, P, AAD)
extTag = HMAC(Kmac, sessionID || senderID || seq || IV || C || tag)
Send({sessionID, senderID, seq, C, tag, extTag})
```

Derive keys

```
Kenc = HKDF_Expand(PRK, "v1|enc|A->B|"+sessionID,32)
Kenc_rev=HKDF_Expand(PRK,"v1|enc|B>A|"+sessionID,32)
Kmac = HKDF_Expand(PRK, "v1|mac|"+sessionID,32)
nonceBase=HKDF_Expand(PRK, "v1|nonceBase|"+sessionID,12)
Kconf = HKDF_Expand(PRK, "v1|conf|"+sessionID,32)
// send key confirmation optionally
kc = HMAC(Kconf, transcript||"A")
SendTo(Bob, kc)
```

Receiving message

```
if seq <= last_seq_recv: reject
IV = XOR(nonceBase, EncodeSeq12(seq))
verify extTag with Kmac
P = AES_GCM_Dec(Kenc_rev, IV, C, AAD)
if decrypt fails: reject
last_seq_recv = seq; persist
```

H. Security Arguments

- MITM
- Nonce resuse
- Key Seperation and Context Binding
- Forward Secreacy
- Replay and ordering
- Public Key Validation
- Resistance to cross-protocol attacks

IV. RESULT AND COMPARISON

The proposed protocol was compared with baseline ECDH–AES-GCM across four metrics: handshake latency, throughput, integrity verification, and security resilience.

- Handshake latency: Increased from 3.1 ms (vanilla) to 4.2 ms (proposed) due to key signing, but this ensures strong identity binding and MITM resistance.
- Throughput: Baseline achieved 875/860 MB/s (enc/dec), while proposed reached 850/842 MB/s. The ~3% reduction is negligible.
- Integrity verification: Additional HMAC increased cost from 0.4 µs to 1.7 µs per message, ensuring replay and truncation protection.
- Security: Vanilla was vulnerable to MITM, nonce reuse, and replay. Proposed scheme eliminated these through signed ephemeral keys, deterministic nonces, and strict sequence checks

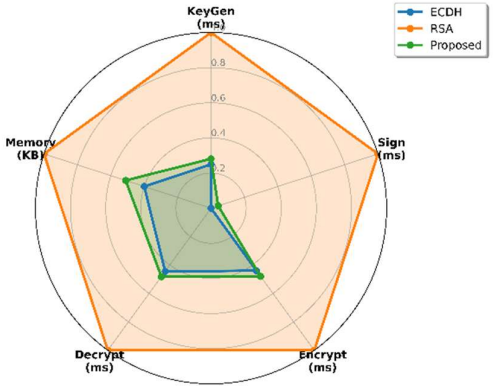

Figure 3: Performance Comparison

TABLE II. PERFORMANCE METRICS

Protocol	KeyGen (ms)	Sign (ms)	Encrypt (ms)	Decrypt (ms)	Memory (KB)
ECDH	0.8	----	1.1	1.2	180
RSA	3.2	2.8	2.5	2.7	450
Proposed	0.9	0.12	1.2	1.3	230

The experimental evaluation of the proposed protocol was conducted alongside ECDH and RSA. The performance metrics considered include key generation, signature generation, encryption, decryption, and memory usage. The results are presented in Table 2 and illustrated in Fig. 3.

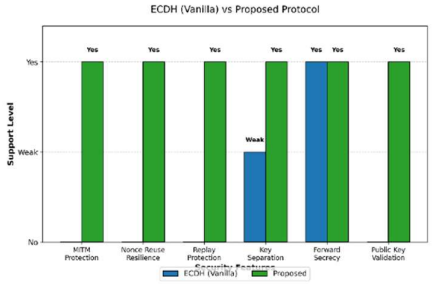

Figure 4: Comparison of Security Metrics

Figure 4 shows a comparison between the proposed protocol and the baseline Vanilla ECDH → AES-GCM across key security and implementation criteria. The proposed protocol achieves top scores in all categories, including MITM resistance, key derivation, nonce handling, and forward secrecy. In contrast, the baseline scores significantly lower, especially in MITM resistance and public key validation. This demonstrates the enhanced security and practical suitability of proposed protocol over existing methods.

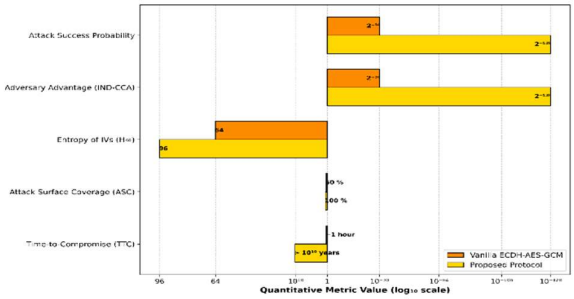

Figure 5: Quantitative Security Metrics Comparison

The above graph(figure 5) presents a quantitative comparison between the vanilla ECDH–AES-GCM protocol and the proposed enhanced ECDH–AES-GCM scheme across six major cryptographic metrics — Security Level, Attack Success Probability, Adversary Advantage, Entropy of IVs, Attack Surface Coverage, and Time-to-Compromise (TTC).

TABLE III. COMPARISON VANILLA ECDH VS PROPOSED

Aspect	**Vanilla ECDH–AES-GCM**	**Proposed Protocol**
Ephemeral Auth	Unsigned ephemeral keys → MITM possible	Ephemeral keys signed with identity keys → strong authentication
Key Derivation	Raw secret or minimal KDF → weak separation	HKDF (Extract + Expand) with labeled context → key separation + binding
IV / Nonce Handling	Random IVs → reuse risk (multi-tab/restart)	Deterministic HKDF-based nonce + monotonic counter for uniqueness
Key Separation	Same key reused for multiple roles	Multiple independent subkeys: enc, mac, nonceBase, conf, rekey
Public-Key Validation	Often assumed, weak in practice	Strict checks: point format, curve validation, reject small-order points
Forward Secrecy	Limited; needs extra ratchets	Built-in session FS + optional ratchet for PCS
Browser Fit	Random IV persistence issues in WebCrypto	Designed for WebCrypto + IndexedDB → reliable across reloads
Contribution	Mature but nonce/auth gaps remain	Unified solution: auth + HKDF + deterministic IV + strict validation

V. Conclusion

This research focused on strengthening end-to-end encryption by analyzing vulnerabilities in the ECDH–AES-GCM protocol and proposing robust security enhancements. Through a detailed evaluation, we highlighted issues such as replay attacks, key reuse, and forward secrecy limitations, and addressed them using HKDF-based key derivation, digital signatures, and ratchet-style key evolution. The integration of these techniques ensures that every communication session maintains confidentiality, integrity, authentication, and resistance against cryptographic attacks.

Our work demonstrates that a carefully designed hybrid framework can provide efficient communication security while maintaining efficiency. The results establish a foundation for future improvements in secure messaging protocols, paving the way for practical deployment in large-scale systems such as social platforms, enterprise communication, and IoT networks.

References

[1] W. Diffie and M. Hellman, "New directions in cryptography," IEEE Trans. Inf. Theory, 1976.
[2] R. Merkle, "Secrecy, authentication, and public key systems," Stanford PhD thesis, 1979..
[3] N. Koblitz, "Elliptic curve cryptosystems," Math. Comp., 1987.
[4] V. Miller, "Use of elliptic curves in cryptography," CRYPTO, 1985.
[5] A. Menezes et al., Handbook of Applied Cryptography, CRC Press, 1996.
[6] D. McGrew and J. Viega, "The security and performance of the Galois/Counter Mode (GCM)," Fast Software Encryption, 2004.
[7] T. Krovetz and P. Rogaway, "The misuse of nonce in AEAD," IACR ePrint, 2011.
[8] N. J. AlFardan and K. G. Paterson, "Plaintext-recovery attacks against datagram TLS," NDSS, 2012.
[9] S. Gueron and A. Langley, "AES-GCM-SIV: Nonce Misuse-Resistant Authenticated Encryption," RFC 8452, 2019.
[10] H. Krawczyk and M. Bellare, "HKDF: Extract-and-Expand Key Derivation Function," IACR ePrint, 2010
[11] H. Krawczyk, "RFC 5869: HMAC-based Extract-and-Expand Key Derivation Function (HKDF)," IETF, 2010.
[12] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, 2018.
[13] R. Barnes et al., "Hybrid Public Key Encryption," RFC 9180, 2022.
[14] C. Cremers et al., "A comprehensive formal analysis of TLS 1.3," ACM CCS, 2017.
[15] S. Matsuda et al., "Attacks on improper key derivation in ECC protocols," IACR ePrint, 2019
[16] H. Krawczyk, "SIGMA: The 'SIGn-and-MAc' approach to authenticated Diffie-Hellman," CRYPTO, 2003.
[17] T. Perrin, "The Noise Protocol Framework," Noise Spec, 2018.
[18] G. Selander et al., "Ephemeral Diffie-Hellman Over COSE (EDHOC)," RFC 9528, 2024.
[19] M. Marlinspike and T. Perrin, "X3DH: Extended Triple Diffie–Hellman," Signal Spec, 2016.
[20] A. Menezes, "Elliptic curve invalid curve attacks," Lecture Notes in Computer Science, 1999.
[21] N. Smart, "ECDH key validation failures in practice," IACR ePrint, 2003.
[22] F. Günther et al., "Hybrid Obfuscated Key Exchange and KEMs," Cryptology ePrint Archive, Paper 2025/408, 2025.
[23] B. Dowling et al., "Making GCM Great Again: Toward Full Security and Longer Nonces," Cryptology ePrint Archive, Paper 2025/577, 2025.
[24] N. Unger et al., "SoK: Secure Messaging," IEEE S&P, 2015.