A Note on Key Agreements and Block Cyphers in Cryptographic Protocols of Secure Messengers

Namdak Tonpa

February 2026

Abstract

Secure messaging applications rely on end-to-end encryption (E2EE) to protect user communications from interception and tampering. This article examines the cryptographic protocols employed by five messaging systems — Threema, Signal, Session, WhatsApp, and Chat X.509 — focusing on their ciphers, algorithms, and underlying design decisions.

Contents

1 Introduction

End-to-end encrypted messaging has become a cornerstone of digital privacy. The strength of such systems depends critically on the cryptographic primitives and protocol constructions chosen during development. This work compares five

representative secure messaging protocols with respect to their key agreement mechanisms, symmetric encryption schemes, authentication methods, secrecy properties, and supporting cryptographic libraries.

The protocols under consideration are:

- Threema
- Signal Protocol (basis for Signal and WhatsApp)
- Session (Oxen/Loki network)
- WhatsApp (modified Signal Protocol implementation)
- Chat X.509 v1/v2

The comparison is structured around a set of core cryptographic parameters, followed by an analysis of the design trade-offs that led to the observed differences.

2 Cryptographic Properties

2.1 Key Agreement and Identity

Most modern messaging protocols use elliptic curve Diffie–Hellman (ECDH) for key agreement. Curve25519 (and its X25519 Diffie–Hellman function) is widely adopted due to its high security margin, constant-time implementation, and resistance to several classes of implementation attacks. NIST P-256 remains in use in environments requiring standards compliance or PKI integration.

Long-term identity keys are typically employed for authentication, with Ed25519 frequently chosen for signing due to its speed and collision resistance.

2.2 Symmetric Encryption and Authenticated Encryption

Authenticated encryption with associated data (AEAD) constructions have become standard. AES-256-GCM is the most widely deployed AEAD mode, offering high performance on modern hardware thanks to AES-NI instructions. The XSalsa20-Poly1305 construction, derived from NaCl/libsodium, provides an alternative stream-cipher-based AEAD scheme that avoids many pitfalls associated with nonce reuse in counter-based modes.

2.3 Forward Secrecy and Post-Compromise Security

Forward secrecy (FS) ensures that compromise of long-term keys does not reveal past session keys. This is typically achieved through ephemeral key exchanges.

Post-compromise security (PCS), also known as future secrecy or self-healing, allows a session to recover confidentiality after a temporary device compromise, provided no further messages are sent by the attacker. The Double Ratchet algorithm is the most widely deployed mechanism providing both properties simultaneously.

2.4 Key Agreements

NIST P-256 (secp256r1)

The NIST P-256 curve, also known as secp256r1, is a 256-bit elliptic curve defined over the prime field $\mathbb{F}_p$ with $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$. It follows the short Weierstrass equation

$$y^2 \equiv x^3 - 3x + b \pmod{p},$$

Standardized in NIST FIPS 186-4 and widely adopted in protocols such as TLS and ECDSA/ECDH, P-256 offers approximately 128 bits of security. The curve parameters were chosen for efficiency on 256-bit processors, with a cofactor $h = 1$ and a prime-order subgroup generated by a fixed base point G. Despite its widespread use and rigorous NIST validation, concerns persist regarding the origin of the constants (selected via unexplained SHA-1 hashing of seeds) and potential for hidden weaknesses, though no practical attacks are known as of 2026.

Curve25519

Curve25519 is a Montgomery-form elliptic curve designed for high-speed Diffie-Hellman key exchange (X25519). It is defined over the prime field $\mathbb{F}_p$ with $p = 2^{255} - 19$, using the equation

$$By^2 = x^3 + Ax^2 + x,$$

where $A = 486662$ and $B = 1$ (often normalized to the equivalent form $y^2 = x^3 + 486662x^2 + x$). The curve provides approximately 128 bits of security with cofactor $h = 8$. Its design emphasizes security against known attacks (twist security, ladder efficiency, and resistance to timing/side-channel leaks) and constant-time scalar multiplication via the Montgomery ladder. Curve25519 is immune to the concerns surrounding NIST curve seeds due to its transparent "nothing-up-my-sleeve" parameters and is the mandatory-to-implement key exchange in TLS 1.3.

Ed25519

Ed25519 is a twisted Edwards-form elliptic curve digital signature scheme derived from Curve25519, operating over the same prime field $\mathbb{F}_p$ with $p = 2^{255} - 19$. The curve equation is

$$-x^2 + y^2 = 1 + dx^2y^2,$$

It provides roughly 128 bits of security with cofactor $h = 8$. Ed25519 uses the Edwards birational equivalence to Curve25519 for efficient, constant-time implementations, offering strong resistance to side-channel attacks and hash-function weaknesses via its EdDSA construction (double-hash of message and

private key). It achieves the highest security level among widely deployed signature schemes (against both classical and quantum side-channel threats) and is standardized in RFC 8032 with widespread adoption in protocols like SSH, Signal, and TLS.

2.5 Block Ciphers

AES-GCM

The Advanced Encryption Standard (AES) is a 128-bit block cipher supporting key sizes of 128, 192, and 256 bits, standardized in FIPS 197. In authenticated encryption mode, AES-GCM (Galois/Counter Mode) combines counter-mode encryption with a Galois-field MAC, providing both confidentiality and integrity. For a 256-bit key, AES-256-GCM offers approximately 128 bits of security against classical attacks (limited by nonce-reuse catastrophic failure and birthday-bound authentication security). The mode is parallelizable, highly efficient on modern hardware with AES-NI instructions, and is the most widely deployed AEAD scheme in TLS 1.3, HTTPS, and disk encryption (e.g., BitLocker). Extensive cryptanalysis over two decades has revealed no practical weaknesses in properly implemented AES-GCM with unique nonces.

ChaCha20-Poly1305

ChaCha20-Poly1305 is an authenticated encryption scheme combining the stream cipher (a 20-round variant of Salsa20) with the Poly1305 one-time authenticator, standardized in RFC 8439. ChaCha20 operates on 512-bit blocks with a 256-bit key, 96- or 128-bit nonce, and 32-bit counter, producing a keystream XORed with plaintext. Poly1305 provides a 128-bit authentication tag using a one-time key derived from ChaCha20. The construction offers approximately 128 bits of security, excellent performance on software platforms without hardware acceleration, and strong resistance to timing attacks due to its addition-rotation-XOR design. It is mandatory-to-implement in TLS 1.3 alongside AES-GCM, widely used in mobile and embedded systems (e.g., WireGuard, Noise Protocol), and considered highly robust with no known practical weaknesses when nonces are unique.

3 Protocol Comparison

3.1 Session (Oxen/Loki)

Session uses libsodium sealed boxes and Argon2id for memory-hard key derivation, reflecting a focus on resistance to offline attacks and decentralized routing. Like Threema, it omits ratcheting to reduce complexity and state requirements, accepting the absence of post-compromise security.

3.2 Threema

Threema adopts the NaCl/libsodium cryptographic API, using XSalsa20-Poly1305 for encryption and Curve25519 for key agreement without ratcheting. The design prioritizes implementation simplicity, constant-time execution, and low latency over post-compromise recovery. This choice is reasonable for a system emphasizing minimal server-side state and moderate group sizes (up to 256 members).

Parameter	**Threema**	**Session**
Key Agreement	Curve25519 ECDH (eph.-static)	Ed25519-X25519
Identity / Long-term	Curve25519 key pair	Ed25519-X25519
Key Derivation	HSalsa20	Argon2id
Symmetric Encryption	XSalsa20	XSalsa20-Poly1305
Authentication / MAC	Poly1305	Poly1305
Authenticated Encryption	XSalsa20 + Poly1305	XSalsa20-Poly1305
Forward Secrecy	Ephemeral (no ratchet)	Ephemeral (no ratchet)
Post-Compromise Security	No	No
Standards / Format	Custom binary (NaCl-inspired)	libsodium sealed boxes

3.3 Signal and WhatsApp

Parameter	**Signal Protocol**	**WhatsApp**
Key Agreement	X25519 (X3DH + 2-Ratchet)	X25519 (X3DH + 2-Ratchet)
Identity / Long-term	X25519 + Ed25519 signing	X25519 + Ed25519 signing
Key Derivation	HKDF (SHA-256/512)	HKDF (SHA-256/512)
Symmetric Encryption	AES-256-GCM	AES-256-GCM
Authentication / MAC	GCM tag / HMAC-SHA256	GCM tag
Authenticated Encryption	AES-256-GCM	AES-256-GCM
Forward Secrecy	Yes (Double Ratchet)	Yes (Double Ratchet)
Post-Compromise Security	Yes	Yes
Standards / Format	Custom binary	Custom binary (Signal)

The Signal Protocol combines the Extended Triple Diffie–Hellman (X3DH) key agreement with the Double Ratchet for per-message key evolution. This design provides both forward secrecy and post-compromise security while supporting asynchronous message delivery and out-of-order message processing. The choice of Curve25519 reflects a preference for modern, implementation-resistant curves over legacy NIST curves. AES-256-GCM was selected for its hardware acceleration and misuse resistance. WhatsApp inherits this design but modifies group key management (Sender Keys) to scale to very large user bases.

3.4 Chat X.509

Parameter	**Chat X.509 v1**	**Chat X.509 v2**
Key Agreement	NIST P-256 ECDH	Curve25519 ECDH
Identity / Long-term	X.509 cert + P-256 key pair	X.509 cert + x25519 key pair
Key Derivation	HKDF-SHA256	HKDF-SHA256
Symmetric Encryption	AES-256-GCM	Chacha20_poly1305
Authentication / MAC	AES-GCM tag	16-byte POLY1305 tag
Authenticated Encryption	AES-256-GCM	POLY1305
Forward Secrecy	Ephemeral (no ratchet)	Ephemeral (no Ratchet yet!)
Post-Compromise Security	No	Ratchet healing
Standards / Format	X.509 X.894 X.680 X.690	X.509 X.894 X.680 X.690

This protocol integrates traditional X.509 public-key infrastructure with NIST P-256 and CMS enveloped data formats. The design favors interoperability with enterprise PKI environments and standards-based tooling over modern secrecy properties. The lack of ratcheting is a deliberate simplification, trading advanced security for compatibility.

4 Future Directions

All five protocols currently offer low quantum resistance due to their reliance on elliptic curves vulnerable to Shor's algorithm. Hybrid post-quantum constructions (e.g., PQXDH combining X25519 with lattice-based key encapsulation mechanisms) are already being deployed experimentally in Signal-based applications. The ongoing standardization of Messaging Layer Security (MLS) promises improved group key management with forward secrecy and post-compromise security at scale.

5 Conclusion

The cryptographic design choices in secure messengers reflect different priorities: maximal secrecy properties (Signal, WhatsApp), implementation simplicity and performance (Threema, Session), or standards compliance and interoperability (Chat X.509 v1). As quantum threats mature and group messaging requirements grow, future protocols will likely combine hybrid post-quantum key exchange, ratchet-based secrecy, and MLS-style group management.

References

[1] Threema GmbH. Cryptography Whitepaper. Threema GmbH, March 2025. Version: March 13, 2025. `https://threema.com/press-files/2_documentation/cryptography_whitepaper.pdf`.

[2] Trevor Perrin, Moxie Marlinspike, and Rolfe Schmidt. The Double Ratchet Algorithm. Signal Messenger, November 2025. Revision 4, 2025-11-04. https://signal.org/docs/specifications/doubleratchet/doubleratchet.pdf.

[3] Trevor Perrin. The X3DH Key Agreement Protocol. Signal Messenger, November 2016. Revision 1, 2016-11-04. https://signal.org/docs/specifications/x3dh/x3dh.pdf.

[4] WhatsApp LLC. WhatsApp Encryption Overview: Technical White Paper. WhatsApp LLC, August 2024. Version 8, Updated August 19, 2024. https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf.

[5] Kee Jefferys, Maxim Shishmarev, and Simon Harman. Session: End-To-End Encrypted Conversations With Minimal Metadata Leakage. *arXiv preprint arXiv:2002.04609*, July 2024. Updated July 4, 2024. https://arxiv.org/pdf/2002.04609.

[6] Loki Project. Loki Network Whitepaper. July 2018. Early foundation document for Session / Oxen messaging. https://loki.network/wp-content/uploads/2020/02/Whitepaper.pdf.