Cryptographic Properties 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 six messaging systems — Chat X.509, Session, Signal, Threema, Viber and WhatsApp — focusing on their ciphers, algorithms, and underlying design decisions.

A comparative analysis of key parameters reveals trade-offs in key agreement, symmetric encryption algorithms, forward secrecy, post-compromise security, performance, and implementation complexity. Protocols implementing the Double Ratchet algorithm achieve superior long-term security properties, while others prioritize simplicity and efficiency at the expense of resilience after key compromise. The implications of these choices for future protocol evolution are discussed.

Contents

1 Cryptographic Properties

This section provides detailed technical descriptions of the cryptographic primitives and protocol mechanisms appearing in the comparison tables across the examined messaging systems. The parameters are organized into subsections corresponding to the table rows, with explanations of each variant used in the protocols.

1.1 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:

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

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.

1.2 Key Agreement

Key agreement establishes shared secrets between parties. All examined protocols use elliptic-curve-based Diffie–Hellman variants.

1.2.1 NIST P-256 ECDH

The NIST P-256 curve (secp256r1) is a 256-bit elliptic curve over the prime field $\mathbb{F}_p$ with $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$, following the short Weierstrass equation

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

It provides approximately 128 bits of security with cofactor $h = 1$. Standardized in NIST FIPS 186-4, it is widely supported but has faced criticism for its parameter origins. NIST P-256 ECDH is used in Chat X.509 v1.

1.2.2 Curve25519 / X25519

Curve25519 is a Montgomery curve over $\mathbb{F}_p$ with $p = 2^{255} - 19$, equation

$$By^2 = x^3 + Ax^2 + x, \quad A = 486662,\ B = 1.$$

The X25519 function provides constant-time scalar multiplication with cofactor $h = 8$ and approximately 128-bit security. Its design prioritizes side-channel resistance and transparent parameters. Curve25519 / X25519 is used in Threema, Signal, WhatsApp, Session, Chat X.509 v2, Viber.

1.2.3 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, Session, and TLS.

1.2.4 Ephemeral-Static ECDH

A single ephemeral ECDH exchange using one party's ephemeral key and the other's static public key establishes a session key. Provides forward secrecy for that session but no per-message key evolution. Ephemeral-Static ECDH keys are used in Threema, Session, Chat X.509.

1.2.5 X3DH + Double Ratchet

Extended Triple Diffie–Hellman (X3DH) combines multiple ECDH exchanges (including signed prekeys and one-time prekeys) for asynchronous authenticated key agreement. The Double Ratchet algorithm then applies symmetric KDF ratcheting and periodic DH ratcheting to derive per-message keys, achieving both forward secrecy and post-compromise security. X3DH is used in Signal, WhatsApp, Viber.

1.3 Identity / Long-term Keys

Long-term keys authenticate parties and bind identities.

1.3.1 Curve25519

Pure static Curve25519 keys registered with the server; no separate signing keys. Curve25519 DH is used in Threema.

1.3.2 Ed25519 + X25519

Long-term X25519 for key agreement, separate Ed25519 key for signing prekeys and identity authentication. Ed25519 is a twisted Edwards curve over the same field as Curve25519:

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

with high security and constant-time implementation (RFC 8032). Ed25519 is used in Signal, WhatsApp, Session, Viber.

1.3.3 X.509 certificate-bound EC keys

Traditional X.509 certificates (ITU-T X.509 / RFC 5280) containing either NIST P-256 (v1) or X25519 (v2) public keys, encoded in ASN.1 DER (X.690). Certificates chain to trusted roots, enabling enterprise PKI integration. X.509 certificate envelopes for keys are used in Chat X.509 v1/v2.

1.4 Key Derivation

1.4.1 HKDF-SHA256/512

HMAC-based Extract-and-Expand KDF (RFC 5869) using SHA-256 or SHA-512. Provides domain separation and strong extraction from shared secrets. HKDF is used in Signal, WhatsApp, Chat X.509, Viber.

1.4.2 Argon2id

Memory-hard password-based KDF (RFC 9106), hybrid of data-independent (Argon2i) and data-dependent (Argon2d) memory access. Designed to resist GPU/ASIC cracking. Argon2id is used in Session.

1.4.3 HSalsa20

Core Salsa20 function applied to a 256-bit key and 128-bit nonce to derive a 256-bit subkey. Used in NaCl/libsodium for XSalsa20 nonce extension. HSalsa20 is used in Threema.

1.5 Symmetric Encryption

1.5.1 AES-256

256-bit key Advanced Encryption Standard (FIPS 197) block cipher in GCM or IGE mode. AES-256-GCM is used in Signal, WhatsApp, Chat X.509 v1.

1.5.2 ChaCha20

20-round variant of Salsa20 stream cipher (RFC 8439) with 256-bit key and 96/128-bit nonce. Addition-rotation-XOR design offers excellent software performance and timing-attack resistance. ChaCha20 is used in Chat X.509 v2.

1.5.3 XSalsa20

Salsa20/20 core with 192-bit nonce extension. First 128 nonce bits and key run through HSalsa20 to produce subkey; remaining 64 bits used as standard Salsa20 nonce. Allows safe random nonce selection. XSalsa20 is used in Threema, Session.

1.6 Authentication / MAC

1.6.1 Poly1305

One-time Wegman–Carter authenticator over $\mathbb{F}_{2^{130}-5}$, 128-bit security with unique keys/nonces. Poly1305 is used in Threema, Session, Chat X.509 v2.

1.6.2 GCM tag

Galois-field authentication (GHASH) providing 128-bit security (birthday bound). GCM tag is used in Signal, WhatsApp, Chat X.509 v1, Viber.

1.6.3 HMAC-SHA256

Standard HMAC construction for additional authentication in ratchet chains. HMAC is used in Signal additional layers.

1.7 Authenticated Encryption

1.7.1 AES-256-GCM

Counter-mode encryption with Galois/Counter Mode authentication. Parallelizable, hardware-accelerated via AES-NI, 128-bit security. AES-256-GCM AE is used in Signal, WhatsApp, Chat X.509 v1, Viber.

1.7.2 ChaCha20-Poly1305

RFC 8439 AEAD: ChaCha20 keystream XOR encryption + Poly1305 authentication. High software speed, mandatory in TLS 1.3. ChaCha20-Poly1305 is used in Chat X.509 v2

1.7.3 XSalsa20-Poly1305

NaCl-style separate encryption and authentication (not strictly AEAD but equivalent security when composed correctly). libsodium sealed boxes: XSalsa20 encryption + Poly1305 authentication using derived one-time keys. XSalsa20-Poly1305 is used in Session and Threema.

1.8 Forward Secrecy

1.8.1 Double Ratchet

Per-message key deletion and DH ratcheting ensure past messages remain confidential even if long-term keys are later compromised. Double Ratchet is used in Signal, WhatsApp, Viber.

1.8.2 Ephemeral

Session keys derived from ephemeral ECDH; provides FS for the session duration but not per-message. Ephemeral without ratchet is used in Threema, Session, Chat X.509.

1.8.3 Periodic Rotation

Keys rotated after 100 messages or one week, limiting exposure window. Periodic rotation is used in Chat X.509 v1.

1.9 Post-Compromise Security

1.9.1 Double Ratchet

Asymmetric DH ratchet introduces fresh entropy, allowing recovery of confidentiality after temporary state compromise. Double Ratchet is used in Signal, WhatsApp, Viber.

1.9.2 Ratchet healing

Custom mechanism providing limited post-compromise recovery (implementation details not fully standardized). Ratchet healing is used in Chat X.509 v2.

No PCS

No mechanism to introduce fresh entropy after compromise; requires new session initiation for recovery. Applicable to Threema, Session, Chat X.509 v1.

2 Protocol Comparison

2.1 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.

2.2 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

2.3 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.

2.4 Viber

Parameter	**Telegram Secret Chats**	**Viber**
Key Agreement	2048-bit DH	X25519 (X3DH + Double Ratchet)
Identity / Long-term	Server-mediated (long-term)	X25519 + Ed25519 signing
Key Derivation	Custom (SHA-256 based)	HKDF (SHA-256/512)
Symmetric Encryption	AES-256-IGE	AES-256-GCM
Authentication / MAC	Custom SHA-256 msg_key	GCM tag
Authenticated Encryption	Custom (IGE + MAC)	AES-256-GCM
Forward Secrecy	Yes (100msgs/1week)	Yes (Double Ratchet)
Post-Compromise Security	No	Yes
Standards / Format	MTProto 2.0 custom	Proprietary (2-Ratchet)
Curve Family	None (finite-field DH)	Curve25519
Quantum Resistance	Low	Low

2.5 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.

2.6 Conclusion

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.

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`.

[7] Rakuten Viber. Viber Encryption Overview. Technical whitepaper. `https://www.viber.com/app/uploads/viber-encryption-overview.pdf`.