A Note on Comparative Analysis of Cryptographic Protocols in Secure Messaging Applications: Ciphers, Algorithms, and Design Decisions

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 Cryptographical 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.

3 Protocol Comparison

3.1 Threema and Session

The table below summarizes the principal cryptographic parameters of the five protocols.

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
Curve Family	Curve25519	Curve25519
Quantum Resistance	Low	Low

3.2 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)
Curve Family	Curve25519	Curve25519
Quantum Resistance	Low (PQXDH variant exists)	Low (PQXDH variant exists)

3.3 Viber and Telegram

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

3.4 Chat X.509

Parameter	**Chat X.509 v1**	**Chat X.509 v2**
Key Agreement	NIST P-256 ECDH (ephemeral)	Curve25519 ECDH
Identity / Long-term	X.509 cert + P-256 key pair	X.509 cert + Curve25519 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
Curve Family	NIST P-256	Curve25519
Quantum Resistance	Low	PQXDH

4 Design Analysis

4.1 Signal Protocol and WhatsApp

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.

4.2 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).

4.3 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.

4.4 Chat X.509 v1

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.

5 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.