Research report

Sub-Question 1: How do established platforms (Signal, WhatsApp, Matrix) currently solve these design challenges, and what best practices can be adapted for a new secure messaging platform?

Introduction

Designing an end-to-end encrypted (E2EE) messaging platform presents several complex technical and ethical challenges. These include ensuring message confidentiality and integrity, maintaining forward secrecy and post-compromise security, supporting asynchronous communication, and protecting user metadata while balancing usability and scalability. Established platforms such as Signal, WhatsApp, and Matrix have spent years iterating on these challenges, producing well-documented and audited solutions. Understanding their approaches provides valuable insights and best practices that can guide the development of a new secure messaging platform, such as the one being designed in this project.

Signal

Signal serves as the foundation for most modern E2EE systems. It combines two major cryptographic protocols: the **Extended Triple Diffie-Hellman (X3DH)** for asynchronous session establishment and the **Double Ratchet** algorithm for per-message key rotation and forward secrecy (Signal, 2016a; 2016b). This combination allows users to initiate secure sessions even when one party is offline and guarantees that each message uses a unique key.

Signal addresses authenticity and trust through its **safety number** verification mechanism, allowing users to verify each other's identity keys out-of-band (Signal Support, n.d.). Furthermore, the platform minimizes metadata exposure using the **Sealed Sender** system, which conceals sender information from the Signal server (Signal, 2018). Recent developments, such as **usernames** and multi-device linking, demonstrate Signal's ability to enhance usability without compromising security (Signal, 2024). Overall, Signal prioritizes cryptographic rigor,

transparency, and minimal data collection, serving as the reference implementation for secure messaging.

WhatsApp

WhatsApp, which integrates the Signal Protocol, expands upon it to support large-scale deployment and multi-device synchronization (WhatsApp, 2024). For individual and group chats, WhatsApp uses the **Signal Protocol** and extends it with a **Sender Keys** mechanism for group encryption. Sender Keys allow one device to generate a symmetric group key shared with members, enabling efficient message distribution without requiring per-recipient encryption (WhatsApp, 2024).

To maintain trust in its large ecosystem, WhatsApp employs **Key Transparency** through an **Auditable Key Directory (AKD)**, an append-only, verifiable structure that ensures servers cannot silently replace user keys (Meta Engineering, 2023). Additionally, WhatsApp's **multi-device architecture** enables up to four linked devices, each with an independent identity key but synchronized session management (Facebook Engineering, 2021). This model preserves E2EE while ensuring consistency and usability across devices. These solutions demonstrate how E2EE can scale to billions of users without sacrificing verification or trust.

Matrix

Matrix adopts a federated approach to secure communication, prioritizing decentralization and interoperability. It uses two cryptographic protocols: **Olm** (inspired by the Double Ratchet) for one-to-one encryption and **Megolm** for group messaging (Matrix.org, 2023). Megolm introduces a group ratchet that balances forward secrecy with scalability by allowing servers to efficiently fan out encrypted messages to large rooms.

Matrix improves multi-device usability through **cross-signing**, where users verify one device to trust all others belonging to the same account. This reduces the overhead of verifying every new device individually (Element, 2020). Furthermore, Matrix employs **Secure Secret Storage and Sharing (SSSS)**, an encrypted key backup mechanism that allows users to recover encrypted data without the server ever seeing plaintext keys (Matrix.org, 2023). Through these mechanisms, Matrix achieves strong encryption guarantees within a federated, multi-device ecosystem.

Best Practices for a New Secure Messaging Platform

The analysis of these platforms reveals a set of best practices for designing a secure and scalable E2EE messaging service:

1. **Adopt a proven protocol foundation**: Combine X3DH for session setup and the Double Ratchet for ongoing communication. This approach ensures forward secrecy and post-compromise security.
2. **Implement sender-optimized group encryption**: For small groups, pairwise sessions may suffice; for larger groups, adopt a Sender Keys or Megolm-like scheme with automatic rekeying upon membership changes.
3. **Provide robust user verification**: Implement verification UX similar to safety numbers or QR codes to detect identity key changes and prevent impersonation.
4. **Design for multi-device use from inception**: Each device should maintain its own identity key, with secure linking and session synchronization.
5. **Incorporate key transparency mechanisms**: Maintain an auditable, append-only key directory to detect key substitution by malicious servers.
6. **Minimize metadata exposure**: Apply Sealed Sender-style techniques to hide sender identity and minimize server-stored data.
7. **Enable encrypted key backup and recovery**: Support encrypted key storage, using client-side passphrases, to improve usability without compromising security.
8. **Automate group key rotation**: Ensure that when a member leaves a group, new sender keys are distributed to prevent data access by former participants.
9. **Maintain open documentation and auditing**: Publish clear technical documentation, similar to WhatsApp's Security Whitepaper, to increase transparency and user trust.

Conclusion

Signal, WhatsApp, and Matrix represent three mature, yet distinct, approaches to end-to-end encryption, each balancing security, scalability, and usability differently. Signal provides the cryptographic gold standard, WhatsApp scales

those principles globally while introducing transparency mechanisms, and Matrix demonstrates how E2EE can work in a decentralized environment. The new secure messaging platform should adopt X3DH and the Double Ratchet as its cryptographic core, integrate sender-key or Megolm-style group encryption, support transparent key management, and prioritize metadata minimization and usability. By synthesizing these best practices, the project can achieve both robust security and practical usability in its E2EE design.

References

Element. (2020, May 6). *E2E encryption by default: cross-signing is here*. https://element.io/blog/e2e-encryption-by-default-cross-signing-is-here/

Facebook Engineering. (2021, July 14). *How WhatsApp enables multi-device capability*. https://engineering.fb.com/2021/07/14/security/whatsapp-multi-device/

Matrix.org. (2023, February 8). *End-to-end encryption implementation guide*. https://matrix.org/docs/matrix-concepts/end-to-end-encryption/

Meta Engineering. (2023, April 13). *Deploying key transparency at WhatsApp*. https://engineering.fb.com/2023/04/13/security/whatsapp-key-transparency/

Signal. (2016a, November 4). *The X3DH key agreement protocol*. https://signal.org/docs/specifications/x3dh/

Signal. (2016b, November 17). *The Double Ratchet algorithm*. https://signal.org/docs/specifications/doubleratchet/

Signal. (2018, October 29). *Technology preview: Sealed Sender for Signal*. https://signal.org/blog/sealed-sender/

Signal Support. (n.d.). *What is a safety number and why do I see that it changed?* https://support.signal.org/hc/en-us/articles/360007060632

WhatsApp. (2024, August 19). *WhatsApp Security Whitepaper* [PDF]. https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf

---

Sub-Question 2: How can cryptographic protocols be selected or designed to ensure forward secrecy and reliability for individual users in one-to-one chats?

Introduction

Ensuring forward secrecy and reliability in one-to-one (1:1) encrypted messaging is essential to protecting user communication against key compromise and data loss. Forward secrecy guarantees that the compromise of a device or key does not expose previously transmitted messages, while reliability ensures message delivery and correct decryption under conditions such as network latency, device restarts, or out-of-order message arrival. This section reviews established approaches adopted by major platforms—Signal, WhatsApp, and iMessage—and synthesizes best practices and cryptographic design principles applicable to the project's one-to-one communication layer.

Cryptographic Foundations

Modern E2EE messaging systems converge on a combination of **asynchronous session establishment** and **continuous key evolution**. The Signal Protocol, used by both Signal and WhatsApp, employs the **Extended Triple Diffie-Hellman (X3DH)** handshake (Marlinspike & Perrin, 2016a) and the **Double Ratchet Algorithm** (Marlinspike & Perrin, 2016b). This pairing allows participants to initiate conversations even when peers are offline and ensures that each message is encrypted with a unique key. Apple's iMessage uses a variant of elliptic-curve Diffie-Hellman for key exchange and per-message AES-GCM encryption, achieving partial forward secrecy but lacking post-compromise security (Green, 2016).

Forward Secrecy and Post-Compromise Security

The Double Ratchet algorithm provides **forward secrecy (FS)** by deriving a new message key for each outgoing message and deleting old keys immediately after use. It also introduces **post-compromise security (PCS)** by mixing new Diffie-Hellman shared secrets into the session state whenever a party's ratchet key changes. In Signal and WhatsApp, each participant's state includes a **Root Key**, **sending chain**, and **receiving chain**; every ratchet step regenerates these keys, ensuring that even if one session key is compromised, future messages remain protected (Signal, 2016b; WhatsApp, 2024).

Reliability Mechanisms in Practice

Reliable decryption under asynchronous delivery is achieved through mechanisms such as **skipped-message key caching** and **chain key advancement**. Signal clients store a limited number of message keys in memory to handle out-of-order messages (Signal, 2016b). WhatsApp inherits this mechanism while adding server-side message queues that hold undelivered ciphertexts until the recipient reconnects (Facebook Engineering, 2021). Both systems treat each message as idempotent, meaning retransmission does not cause duplicates or data corruption. iMessage similarly queues encrypted messages on Apple's servers until devices retrieve them, but the system does not maintain Double Ratchet-style state updates, providing reliability without full PCS (Green, 2016).

Key Management and Rotation

Signal and WhatsApp use long-term **identity keys**, medium-term **signed prekeys**, and short-lived **one-time prekeys**. This multi-tier key structure supports asynchronous session initiation while minimizing exposure if a key is leaked. Devices periodically rotate signed prekeys, reducing the window of potential compromise (Signal, 2016a). WhatsApp enforces the same rotation policy and uses device-specific identity keys to maintain secure multi-device synchronization (Facebook Engineering, 2021).

Multi-Device Synchronization

Forward secrecy becomes more complex when users operate multiple devices. WhatsApp and iMessage assign each device a distinct identity key and establish independent encrypted sessions for each peer device (Meta Engineering, 2023; Green, 2016). Signal achieves this by linking secondary devices through a verified session, replicating message history using end-to-end encrypted transfers. Each device maintains its own ratchet state, ensuring that compromise of one device does not automatically expose the sessions of others.

Protocol Design Recommendations

Based on the practices of these platforms, the following design principles are recommended for ensuring forward secrecy and reliability in 1:1 messaging:

1. **Adopt X3DH for asynchronous session setup** to enable secure initiation without simultaneous connectivity.

2. **Use the Double Ratchet algorithm** for ongoing communication to provide per-message forward secrecy and post-compromise security.
3. **Implement reliable delivery logic** through message key caching, replay detection, and persistent session storage.
4. **Enforce hierarchical key lifetimes** (identity, signed prekey, one-time prekey) and periodic rotation.
5. **Support independent device sessions** per user and provide verification mechanisms such as safety numbers or fingerprints.
6. **Persist ratchet state securely** after each encryption or decryption operation to prevent loss of synchronization.
7. **Utilize authenticated encryption** (e.g., ChaCha20-Poly1305 or AES-GCM) to ensure message integrity and authenticity.

Conclusion

The combination of X3DH and the Double Ratchet algorithm, as implemented in Signal and WhatsApp, currently represents the industry standard for secure and reliable one-to-one encrypted communication. These protocols ensure that even under device compromise, previous messages remain confidential and future communication can self-heal. Incorporating these mechanisms, along with robust session persistence, key rotation, and multi-device session management, will enable the project to achieve both forward secrecy and high reliability in its 1:1 messaging component.

References

Facebook Engineering. (2021, July 14). *How WhatsApp enables multi-device capability*. https://engineering.fb.com/2021/07/14/security/whatsapp-multi-device/

Green, M. (2016, June 15). *A look at Apple's iMessage encryption protocol*. *A Few Thoughts on Cryptographic Engineering.* https://blog.cryptographyengineering.com/2016/06/15/a-look-at-apples-imessage-encryption/

Marlinspike, M., & Perrin, T. (2016a). *The X3DH key agreement protocol*. Signal. https://signal.org/docs/specifications/x3dh/

Marlinspike, M., & Perrin, T. (2016b). *The Double Ratchet algorithm*. Signal. https://signal.org/docs/specifications/doubleratchet/

Meta Engineering. (2023, April 13). *Deploying key transparency at WhatsApp*. https://engineering.fb.com/2023/04/13/security/whatsapp-key-transparency/

WhatsApp. (2024, August 19). *WhatsApp Security Whitepaper* [PDF]. https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf

Sub-Question 3: How can group key management mechanisms be structured to enable multi-device users in group messaging contexts to exchange messages efficiently with scalability up to thousands of participants?

Introduction

Group messaging introduces additional complexity compared to one-to-one communication, especially when users participate from multiple devices. A secure system must ensure that every authorised device can decrypt group messages, while maintaining end-to-end encryption (E2EE), forward secrecy, and post-compromise security. At the same time, the protocol must remain efficient as groups grow to thousands of participants. These challenges are well-documented in modern secure messaging literature (Marlinspike, 2013; Cohn-Gordon et al., 2020; Beurdouche et al., 2023).

This section analyses existing group key management approaches, focusing on Sender Keys-style schemes (Signal, WhatsApp), Megolm (Matrix), and tree-based Messaging Layer Security (MLS). It then derives design principles and concrete recommendations for structuring group key management in the project's E2EE messaging platform.

Design Space for Group Key Management

Group key management mechanisms must balance several competing requirements:

- **Scalability:** Cost of sending messages and updating keys must remain practical as group sizes scale (Beurdouche et al., 2023).
- **Security properties:** Confidentiality, authentication, forward secrecy, and post-compromise security must hold (Cohn-Gordon et al., 2020).
- **Multi-device support:** Each authorised device must independently participate in group cryptography.
- **Asynchronicity:** Devices may be offline, which requires protocols tolerant of message delays.
- **Membership dynamics:** Systems must efficiently handle joins and removals.

Existing systems implement these requirements via sender-optimised symmetric keys (Sender Keys), symmetric room ratchets (Megolm), and tree-based group key agreement (MLS).

Sender Keys-Style Group Encryption

Basic idea

Sender Keys, introduced as part of WhatsApp's deployment of E2EE (WhatsApp, 2016), extend the Signal Protocol (Marlinspike, 2013) to group messaging. Each sending device maintains:

- A symmetric **sender chain key** for deriving per-message keys.
- A **signing key pair** for message authentication.

A Sender Key Distribution Message (SKDM) is delivered over the pairwise X3DH + Double Ratchet channels to all devices in the group.

Scalability characteristics

Sender Keys separates expensive operations (key distribution) from inexpensive ones (message sending). SKDM distribution is rare, whereas message sending is constant-cost. This design achieves strong scalability (Marlinspike, 2013; WhatsApp, 2016).

Multi-device implications

Each device participates independently, improving security isolation but increasing the number of SKDMs required. This model is consistent with formal analyses of Signal-style protocols (Cohn-Gordon et al., 2020).

Membership changes and rekeying

Selective rekeying ensures that removed devices cannot decrypt future messages. This strategy aligns with industry deployments (Marlinspike, 2013; WhatsApp, 2016).

Megolm: Room-Level Symmetric Ratchet

Matrix's Megolm protocol is optimised for large rooms (Matrix.org, 2019). Megolm uses a single symmetric session per sender device; message sending is efficient, but distributing the Megolm session to new devices incurs overhead. Megolm provides weaker post-compromise security than the Signal family (Matrix.org, 2019).

Tree-Based Group Key Agreement (MLS)

MLS introduces TreeKEM, a scalable group key agreement algorithm offering logarithmic update cost and strong post-compromise guarantees (Beurdouche et al., 2023). MLS explicitly models devices as group members, making it suited for dynamic and multi-device environments.

Design Considerations for Multi-Device Users

Supporting multi-device users requires:

1. **Device identities**, allowing participants to authenticate each device (Marlinspike, 2013).
2. **Efficient onboarding** via X3DH and Double Ratchet (Cohn-Gordon et al., 2020).
3. **Targeted rekeying** when devices leave.
4. **Server-assisted ciphertext fan-out** without exposing keys.

Recommended Architecture for the Project

Short- to medium-term: Sender Keys-style scheme

- Per-device identity keys and pairwise channels (Cohn-Gordon et al., 2020).
- Per-device sender keys following Signal/WhatsApp practice (Marlinspike, 2013; WhatsApp, 2016).
- Out-of-band history sharing for new devices.

Long-term: Migration path to MLS

MLS can be introduced gradually for large public channels or groups requiring efficient key updates (Beurdouche et al., 2023).

Conclusion

Sender Keys-style mechanisms provide excellent short-term scalability and align with widely deployed secure messengers. MLS presents a promising long-term direction for supporting extremely large or dynamic groups.

References

Beurdouche, B., Bhargavan, K., Tomlinson, L., & Zinzindohoue, J. K. (2023). *Messaging Layer Security (MLS) Protocol Specification*. IETF.

Cohn-Gordon, K., Cremers, C., Dowling, B., Garratt, L., & Stebila, D. (2020). A Formal Security Analysis of the Signal Messaging Protocol. *IEEE European Symposium on Security and Privacy*.

Marlinspike, M. (2013). *The Signal Protocol*. Open Whisper Systems.

Matrix.org. (2019). *Megolm and Olm Cryptographic Overview*.

WhatsApp. (2016). *End-to-End Encryption Design Whitepaper*. https://www.whatsapp.com/security/