ChainLynx Bikepacking App – Messaging System

Pre-Release Technical Documentation — November 2025

1. Overview

The messaging system in the ChainLynx Bikepacking App provides secure, lightweight, and privacy-focused communication for users in both connected and offline contexts. It is designed to support one-to-one and small-group conversations among riders, ensuring end-to-end encryption and minimal data exposure.

2. Concept and Objectives

Concept: A hybrid peer-to-peer messaging layer where the central backend acts only as a temporary relay. Messages are encrypted on the sender's device and decrypted on the recipient's device, ensuring complete confidentiality.

Objectives: • Provide end-to-end encrypted (E2EE) communication using a proven cryptographic protocol.
• Ensure messages remain accessible offline until delivered.
• Minimize metadata stored or visible to the backend.
• Enable seamless user experience with low power consumption and reliable delivery.

3. System Architecture

Implementation: The messaging service consists of three layers:
1. Local Device Layer — Handles encryption, key management, and message caching.
2. Relay Layer — Cloud-based message queue that stores encrypted payloads temporarily.
3. Delivery Layer — Performs message delivery when connectivity is restored.

Protocol: The app implements the Signal Protocol (Double Ratchet + X3DH) for encryption, providing forward secrecy and deniability. No message content or encryption keys are ever accessible to the server.

4. Encryption and Security

Encryption Model:
• Each conversation generates unique session keys using the Signal Protocol.
• Messages are encrypted with AES-256 and authenticated with HMAC-SHA256.
• Key exchanges use Curve25519 elliptic-curve cryptography.
• The backend stores only encrypted message blobs, timestamps, and recipient hashes.

Device Keys: Private keys never leave the device. Public identity keys are exchanged only once via the authentication service. Session rekeying occurs automatically on each new message exchange to maintain forward secrecy.

5. Message Flow

Workflow:
1. Sender composes a message and encrypts it locally using session keys.
2. The encrypted message is sent to the backend relay, tagged with the recipient hash.
3. The relay temporarily stores the encrypted payload until delivery.

4. Recipient retrieves and decrypts the message using local keys.
5. The relay deletes the message immediately after confirmed receipt.

This approach guarantees minimal server-side exposure while ensuring delivery reliability even under intermittent connectivity.

6. Data Persistence and Privacy

Data Retention: The backend never permanently stores messages. Each message exists only until delivery confirmation, after which it is automatically purged.
Offline Behavior: If a user is offline, messages are cached locally and synchronized when connectivity resumes.
Logs and Metadata: The server logs contain no plaintext content, only delivery timestamps and anonymized hashes.

7. Integration with React Native

Implementation: The messaging module is implemented using TypeScript within the React Native framework, leveraging native bindings for cryptographic operations.
Libraries: The Signal protocol is integrated through the libsignal-client library.
Storage: Local message caching is handled via SQLite or Realm, ensuring fast and reliable offline operation.

Push notifications (when online) are delivered via Firebase Cloud Messaging (Android) and Apple Push Notification Service (iOS), without exposing any message content.

8. Recommendation

The hybrid E2EE architecture is both practical and secure. It preserves ChainLynx’s privacy-first philosophy while ensuring a dependable user experience, even in challenging network conditions. The approach scales easily, integrates well with existing React Native modules, and meets global data protection standards.