LEMNISCATE

# Security Whitepaper

How Lemniscate protects chats, calls, group calls, webinars and backups, what our servers can and cannot see, and where the limits are.


| | |
|---|---|
| Version | **1.1** |
| Date | **11 November 2026** |
| App version described | **4.567 (web, Android, iOS and desktop share the same client code)** |
| Publisher | **Lemniscate Labs LLC** |
| Contact | **security@lemniscate.network** |


**About this document.** This whitepaper is written by Lemniscate Labs. It is not an independent audit, and we do not call it one. Every statement describes the code that ships today. The client code is delivered to every browser and can be inspected by anyone. An external review is planned (section 11).

# Contents


1 Summary
2 Accounts and keys
3 Cryptographic building blocks
4 Messages
5 Calls, group calls and webinars
6 Linking a new device
7 Backups
8 Notifications
9 Protection on the device
10 What our servers see
11 Known limits and external review
12 Reporting a vulnerability


## 1 Summary

**Everything private is end-to-end encrypted by default.** Chats, voice messages, images, files, one-to-one calls, group calls and webinars are encrypted on the sending device and decrypted only on the receiving devices. There is no setting to switch this off and no unencrypted fallback.

**Post-quantum protection is live.** Every message key is agreed with two methods at once: classical elliptic curve Diffie-Hellman (P-256) and ML-KEM-1024, the NIST post-quantum standard (FIPS 203) at its highest security category. An attacker has to break both. Traffic recorded today cannot be decrypted later by a future quantum computer that only breaks the classical part.

**Nobody can write in someone else's name.** One-to-one messages and call setups carry a sender proof that only the real sender can create, and the receiving app checks it. Not even our servers can forge it.

**Keys never leave your devices unencrypted.** Your keys are created on your device from a 12-word recovery phrase. We never receive them. Our servers store and forward sealed envelopes and cannot open them.

**We are honest about limits.** Like every centrally operated messenger, our servers can see some metadata (for example that an envelope was delivered to an account at a certain time). A device that is taken over by spyware can read what its owner reads. Section 11 lists every limit we know of.

## 2 Accounts and keys

### One recovery phrase

A new account starts with a 12-word recovery phrase (BIP-39), generated on the device. From this phrase the app derives, with clear domain separation:

- **Chat key (ECDH P-256):** derived with HKDF-SHA256 and the label `lemniscate-chat-p256-v1`. Its public half is your account identity.
- **Post-quantum key (ML-KEM-1024):** generated deterministically from SHA-512 over the label `lemniscate-mlkem-v1` and the chat private key. SHA-512 is one-way, so the post-quantum key reveals nothing about the chat key.
- **Wallet key:** the standard BIP-44 path `m/44'/60'/0'/0/0`. It is a separate key on a separate path and is never used for message encryption.

In addition, each account has a **signing key (ECDSA P-256)** that is generated on the device and travels only inside encrypted backups or an encrypted device link.

The recovery phrase is kept in memory only while it is needed. It is not sent to our servers.

### Signed requests

Requests that change something important (account changes, linked devices, payments) carry an ECDSA signature over the exact request contents, a timestamp and a random 96-bit value. The server rejects requests whose signature does not match and requests it has already seen (replay protection). Wallet withdrawals need a second signature made with a key that is only unlocked by your wallet PIN (section 9).

## 3 Cryptographic building blocks

We use standard, well-studied algorithms. The classical primitives run in the operating system's or browser's own cryptography engine (Web Crypto). ML-KEM, which browsers do not offer yet, is implemented by us following FIPS 203.

| Building block | Used for | Standard |
|---|---|---|
| ML-KEM-1024 | Post-quantum part of every message key | NIST FIPS 203, category 5 |
| ECDH P-256 | Classical part of every message key, call key agreement | NIST SP 800-56A |
| ECDSA P-256 | Signatures on requests and messages | NIST FIPS 186 |
| AES-256-GCM | Encryption of all content, call frames, backups | NIST SP 800-38D |
| HMAC-SHA256 | Sender proof inside envelopes and call setups | FIPS 198-1, RFC 2104 |
| HKDF-SHA256 | Combining secrets, deriving keys, call key chains | RFC 5869 |
| PBKDF2-SHA256 | Keys from passwords and PINs (600,000 iterations) | NIST SP 800-132, OWASP |
| BIP-39, BIP-44 | Recovery phrase and wallet path | Bitcoin Improvement Proposals |

Not used anywhere: MD5, SHA-1, RC4, DES or 3DES.

# 4 Messages

### The envelope

For every single message the sending device creates a fresh, one-time ECDH key pair. It then:

- computes an ECDH secret with the recipient's chat key,
- encapsulates a second secret to the recipient's ML-KEM-1024 key,
- combines both secrets with HKDF-SHA256 (label `lemniscate-hybrid-v1`) into an AES-256-GCM key,
- encrypts the content with a random 96-bit nonce. The GCM tag detects any change.

```
0x02 | one-time public key (65 bytes) | ML-KEM ciphertext (1568 bytes) | nonce (12 bytes) |
ciphertext and tag
```

If a recipient still uses an old app version that has not published a post-quantum key, the envelope falls back to ECDH P-256 with AES-256-GCM. It never falls back to plaintext.

### Sealed sender

One-to-one texts, stickers, images, voice messages, video notes and files are sent as **sealed envelopes** by default. On the outside the server only sees a random sender value. The real sender is inside the encrypted envelope, so the server cannot log who wrote to whom for these messages.

### Sender proof

Inside the envelope the sending app adds a sender proof: an HMAC-SHA256 code over the direction, the message ID, the type and a hash of the content. Its key is derived from an ECDH exchange between the sender's and the recipient's identity keys, so only these two can compute

it. The receiving app checks the proof and discards anything that fails. This way nobody, not even our servers, can send a message in someone else's name. All devices of an account share the identity key, so the proof works on every device.

Envelopes from app versions before 4.567 carry no proof. They are still accepted until 24 November 2026, unless that sender has already written with a proof. After that date, envelopes without a proof are discarded.

### Groups

Group messages are encrypted separately for every member with that member's keys (pairwise fan-out). There is no group key on the server and no group plaintext anywhere outside the devices. Mute settings for groups work with a per-recipient hash, so the server cannot see group IDs or member lists from it.

### Disappearing and view-once

Chats can be set to delete messages automatically after a chosen time. Images and voice messages can be sent as view-once. The server deletes an envelope once the recipient's active devices have fetched it, and undelivered envelopes expire after a limited time.

## 5 Calls, group calls and webinars

### One-to-one calls

Audio and video run over WebRTC and are encrypted with DTLS-SRTP between the two devices. When a direct connection is not possible, our relay servers (TURN) forward the encrypted packets without being able to read them. All call setup messages, including the DTLS fingerprints, travel inside the end-to-end encrypted envelopes from section 4 and carry the sender proof. Our servers can neither read nor forge them, so they cannot place themselves in the middle of the key exchange.

### Group calls

Group calls run through our own media server (SFU). On top of the transport encryption, every single audio and video frame is encrypted with AES-256-GCM on the sending device before it leaves (WebRTC encoded transform). The media server only forwards ciphertext.

- Every participant has their own key chain. The chain moves forward with HKDF every 300 frames, so an old key cannot decrypt newer frames.
- Keys are sent to each participant inside end-to-end encrypted envelopes, never through the media server.
- Someone who joins later only receives keys from that moment on. When someone leaves or is removed, everyone switches to a fresh key chain that the removed person never receives.
- **Fail closed:** without a key, no frame leaves the device. Frames that are unencrypted or fail authentication are discarded. A device that cannot do this encryption cannot join, so there is no silent downgrade.

This is on by default for every group call.

### Webinars

For the audience, the host's device seals every piece of the live stream with AES-256-GCM before uploading it. Our relay, edge servers and content delivery network only distribute ciphertext. Each viewer receives the content key inside an end-to-end encrypted envelope. When a viewer is removed, the host switches to a new key. This is on by default.

Speakers on the stage, and viewers who receive the stream directly from the host as a fallback, connect with WebRTC and DTLS-SRTP between the devices. Their connection setup is end-to-end encrypted and carries the sender proof, exactly like one-to-one calls.

### Call records on the Lemniscate blockchain

For every finished one-to-one call, the server writes a SHA-256 fingerprint of the call to the Lemniscate Mainchain, bundled with other calls (up to 64 per transaction, every two minutes). The chain only receives these hashes: no names, no accounts, no phone numbers and no readable durations. The bundling also hides the exact time of a single call.

## 6 Linking a new device

To add a device, the existing device creates a random 256-bit key and shows it only inside a QR code on its screen. The account data for the new device is encrypted with this key (AES-256-GCM) before it is uploaded. The server stores the ciphertext under a random short-lived token, but never sees the key, because the key only travels from screen to camera.

## 7 Backups

Backups are zero-knowledge: our servers store only encrypted data and cannot open it.

- **Backup with the recovery phrase:** encryption key, write key and storage name are derived from the phrase with HKDF-SHA256. The storage name is a hash and is not linked to your account key or username. Contents are encrypted with AES-256-GCM. Changes must be signed with a separate key derived from the phrase.
- **Backup with e-mail and password:** the key is derived on the device with PBKDF2-SHA256 (600,000 iterations) from your password, salted with your e-mail address, and then split with HKDF-SHA256. The storage name is a hash of the e-mail address. Without the password nobody, including us, can decrypt the backup.
- **Archive export:** a backup file you save yourself is encrypted with AES-256-GCM using a key from your chosen password (PBKDF2-SHA256, 250,000 iterations).

> **Important**
>
> If you lose both your recovery phrase and your backup password, your data cannot be recovered. This is the direct consequence of us not holding any keys.

## 8 Notifications

- **Android push (Firebase):** the push only says that something new is waiting (a message, a call or an event). It contains no text, no sender and no chat. The app then fetches and decrypts the envelope itself.
- **Web push (browsers):** the content is encrypted for your browser as defined in RFC 8291. The push services of Google, Mozilla or Apple only see ciphertext.
- **iOS:** system notifications use generic texts (for example a missed call) and never contain message content.

## 9 Protection on the device

- **App lock and a separate wallet PIN.** The wallet PIN is stretched with PBKDF2-SHA256 (600,000 iterations), and repeated wrong attempts lead to growing waiting times. Access to the unlocked phone alone is not enough to move funds: withdrawals need the second signature from section 2.
- **Screenshot and screen recording protection** is on by default in the Android app (the system's secure window flag), so other apps only see a black screen. Web browsers do not allow this kind of protection.
- **Secure keyboard** is on by default. It keeps your typing away from third-party keyboards, one of the most common tools of everyday stalkerware. You can switch to the system keyboard at any time.
- **Disappearing messages and view-once media** (section 4).

> **The honest limit of every messenger**
>
> End-to-end encryption protects everything between devices. A phone that has been taken over by state-level spyware (for example Pegasus) reads the screen and keyboard directly. This has been documented against users of every major messenger. No app can undo that. Keeping the operating system updated, avoiding unknown apps and profiles, and for high-risk people using Apple's Lockdown Mode or a separate device are the real protections.

## 10 What our servers see

The data model for private messages on our servers has exactly these fields: sender value (random for sealed envelopes), recipient key, message type, sealed envelope, timestamps, expiry, signature, and two optional technical markers (a hashed mute tag and a flag for mentions). There is no field for readable content.

- Server logs are kept in memory only, never written to disk. IP addresses are shortened and request parameters are not logged. Swap and crash dumps are disabled.
- Data partitions on our servers are encrypted at rest.
- All connections from apps and browsers use TLS. Traffic between our own servers runs through encrypted tunnels.
- Our servers run on our own infrastructure in the EU and the US.

### Who can read or hear what

| Who | Content | Why |
|---|---|---|
| Lemniscate itself | No | We never hold your keys. |
| An attacker with full access to our servers | No | Only ciphertext is stored. Some metadata would be visible (section 11). |
| Network operators, Wi-Fi owners, man in the middle | No | TLS outside, end-to-end encryption inside. |
| Authorities with an order addressed to us | No | We can only provide minimal metadata. We do not have content. |
| A future quantum computer with recordings of today | No | ML-KEM-1024 protects the key agreement already today. |
| Push services (Google, Apple, Mozilla) | No | Pushes carry no content or are encrypted for the browser. |
| Another user who wants to write in your contact's name | No | The sender proof fails (section 4). |
| Spyware that has taken over the device | Yes | As with every messenger: a compromised device sees what its owner sees. |

## 11 Known limits and external review

We prefer a complete list over marketing. These are the limits we know of today:

- **Metadata.** Our servers see which account receives envelopes, when and how large they are. Sealed sender hides the sender of one-to-one messages, but not every message type is sealed yet (for example group and control messages).
- **Forward secrecy for messages.** Every message uses a fresh one-time key on the sender side, but the recipient's long-term key stays the same. A continuous key ratchet for one-to-one chats (as in the Signal protocol) is on our roadmap. Group calls already use a ratcheting key chain (section 5).
- **Key directory.** Public keys are handed out by our servers. A manipulated server could give you a wrong key for a contact you have never verified. You can rule this out with "Verify encryption" in the chat: compare the 60-digit safety number or scan your contact's QR code.
- **Groups and other control messages.** Group messages and some control messages are signed by the sender and checked by our servers, but not yet by the receiving app. A manipulated server could therefore inject such messages in someone else's name (it still could not read genuine ones). We plan to extend the in-envelope sender proof to groups together with the new key ratchet.
- **Old app versions.** Recipients who have not updated may receive messages with classical encryption only (section 4), never plaintext. Envelopes without a sender proof are accepted only until 24 November 2026 (section 4).
- **Web version.** In a browser the app code is loaded from our servers each time, as with every web messenger. The installed apps carry the code inside the app package.

- **Call fingerprints.** A fingerprint on the blockchain reveals nothing by itself. Someone who already knows every detail of a call could confirm that this call happened.
- **Public content.** Public posts, public profiles and public channels are public by design.
- **No external audit yet.** This whitepaper is our own description. We plan an independent review of the encryption core (key creation, envelope format, hybrid key agreement, group fan-out, call key chains) by a recognised security firm, and we will publish its report.

## 12 Reporting a vulnerability

If you find a security problem, please write to **security@lemniscate.network** with the details. We answer quickly and fix confirmed issues with priority. Please do not exploit a problem or publish it before it is fixed. We are happy to credit researchers who want to be named.

---

Lemniscate Labs LLC, 30 N Gould St Ste R, Sheridan, WY 82801, USA · lemniscate.network/security
Security Whitepaper, version 1.1, 11 November 2026, describing app version 4.567. Changes from 1.0: sender proof for sealed messages, call setups and webinar setups; webinar stage setup is now end-to-end encrypted. This document describes the design as implemented at the time of writing and will be updated when the design changes.