SumaFlow Minutes — Privacy & Security Whitepaper

Version: 1.0 **Audience:** technically literate readers who will verify our claims — the r/privacy and Hacker News audience, security-minded professionals, and the regulated practitioners we are built for. **Scope:** SumaFlow Minutes for Android, v1.0.0. Describes the as-built architecture. **Last updated:** 2026-06-02

---

0. Summary

SumaFlow Minutes records a meeting, transcribes it, and generates structured minutes — **entirely on your phone**. There is no account, no cloud, and no server that holds your data, because there is no server at all. Your audio, transcripts, and minutes never leave the device unless *you* explicitly export a single meeting, and the app shows you exactly what will leave and where before it does.

We make seven privacy promises (§1). The rest of this document explains the architecture that backs them, names the one place the app is allowed to touch the network and why, and is honest about what this design does **not** protect against. Where a claim is machine-checkable, we point you at the open-source code and the executable test that checks it (§8) — the design intent is that you should not have to take our word for any of this.

We describe what regulations *require* and how an on-device, no-cloud design helps you meet your own confidentiality obligations. We do **not** claim any certification. SumaFlow Minutes does not claim compliance with HIPAA, FINRA, SEC rules, or any security-audit or information-security certification — and you should distrust any meeting tool that advertises such a claim without showing you the audit behind it. We earn trust through architecture, not labels.

---

1. The privacy contract (the seven promises)

These are the product. In v1 they hold without exception; we treat a violation of any of them as a critical (P0) defect.

1. **Audio never leaves the device.** Recording, transcription, and minutes generation all run locally.
2. **Transcripts and minutes are stored encrypted at rest** on the device.
3. **Zero outbound network calls** from the core app, ever, except user-initiated export intents. No telemetry. No analytics. No “anonymous” launch counters. (The one narrow, opt-in exception — model download — is described plainly in §6; it carries no user content.)
4. **Exports are user-initiated, one meeting at a time, with a confirmation screen** showing exactly what will be sent and where.
5. **No account is required.** No cloud sync. No login.
6. **An open architecture page** explains exactly what runs where.
7. **Privacy-critical code is open source**, so the claims above are verifiable rather than asserted.

The rest of this document is the evidence for these promises.

---

2. Threat model

Security claims are meaningful only against a stated adversary. Here is who SumaFlow Minutes is designed to protect you from — and who it is not.

Who we protect against

- **A lost or stolen device.** Your recordings and minutes are encrypted at rest (§3, §5) with a key held in hardware-backed storage (§4). An attacker with the physical device but not your unlocked screen does not get your meeting content.
- **A curious cloud.** There is none. We cannot read, mine, subpoena-disclose, or accidentally leak data we never receive. “We don’t look at your data” is a policy; “we never receive your data” is an architecture. We chose the architecture.

- **A subpoena to a server that does not exist.** Because no audio, transcript, or minutes content is ever transmitted to us, there is no server-side copy for a third party to compel from us. (Your own device, and your own exports, remain yours to manage — see §10.)
- **Passive network observers.** The core app makes no outbound calls carrying user content, so there is nothing on the wire to observe. The single allowlisted network use (§6) transfers a *public AI model file*, never your data.

Who we do not protect against (and we are explicit about this)

- **A compromised operating system or a rooted device with the screen unlocked.** If the OS itself is malicious or the device is unlocked in an attacker's hands, on-device encryption cannot save data that is, by definition, available to the running system. No on-device app can.
- **Shoulder-surfing and physical observation.** If someone can see your screen, they can read your minutes.
- **Your own export choices.** Once *you* export a meeting to a PDF, an email, or another app, that copy lives by the rules of wherever you sent it — not ours. The app's job is to make that boundary explicit (§4 promise, §7); it cannot follow the data after you hand it elsewhere.
- **The integrity of the underlying platform crypto and keystore.** We rely on the Android Keystore and platform AES-GCM (§4, §5). A break in those is below our layer.

We state these limits plainly because the audience that scrutinizes a privacy whitepaper is exactly the audience that will notice if we don't.

---

3. Architecture — what runs where

Everything that touches your meeting content runs **on your phone**:

Stage	Where it runs	How
Recording	On-device	Microphone → foreground service → app-private storage
Encryption at rest	On-device	AES-256-GCM (§5) before anything is persisted
Transcription	On-device	whisper.cpp via Dart FFI — no network
Minutes generation	On-device	LiteRT-LM + Gemma 4 E2B (if the model is present and RAM allows), or a deterministic Dart extractor otherwise
Storage	On-device	SQLCipher (encrypted SQLite) + encrypted audio files in app-private scoped storage
Export	On-device → where *you* send it	User-initiated intent, one meeting at a time, with a confirmation screen (§7)

There is **no server**. There is **no account**. There is **no cloud sync**. The minutes-generation step has two on-device paths — a downloaded open-weight model (LiteRT-LM + Gemma 4 E2B) for most hardware, and a pure-Dart deterministic template extractor as a graceful fallback for RAM-constrained devices or users who decline the model download. Both run locally; neither transmits your content.

---

4. Key management

- **One master key per install.** On first launch the app generates a single 256-bit master key drawn from the platform cryptographically-secure RNG (never `dart:math`). It is stored via `flutter_secure_storage`, which is backed by the **Android Keystore** and uses **StrongBox** (the dedicated secure element) where the device provides one.

- **The master key is non-exportable and never leaves the device.** It is not transmitted, backed up to a cloud, or written anywhere outside Keystore-backed storage.
- **All working keys are derived, not stored.** The audio-file encryption key and the SQLCipher database key are derived from the master key with **HKDF-SHA256**, each under a distinct purpose label (`audio-aead-v1`, `sqlcipher-v1`). Separating purposes means a single derived key is scoped to a single job; the master key itself is never used directly to encrypt content.
- **No password, no passphrase, no key escrow.** There is no account to attach a key to and no recovery server. This is deliberate: a recovery path is also an exfiltration path. The trade-off is stated honestly in §10 — a factory reset or loss of the device's keystore is unrecoverable by design.

---

5. Encryption details

At-rest file format

Audio is encrypted with **AES-256 in GCM mode** (an authenticated cipher) using the platform-native AES-GCM implementation. Each encrypted file on disk is:

```
[ 12-byte nonce ][ 16-byte GCM authentication tag ][ ciphertext ]
```

- A **fresh 96-bit (12-byte) random nonce** is generated for every encryption from the platform CSPRNG. Nonces are never reused under a given key.
- The **16-byte GCM tag** authenticates the ciphertext. On decryption, a tag mismatch — whether from tampering or the wrong key — fails closed: the app refuses to return plaintext rather than returning corrupted or attacker-influenced data. (Tampering and a wrong key are deliberately indistinguishable from the AEAD's perspective; both simply fail authentication.)

Why GCM

GCM is an AEAD (authenticated encryption with associated data): it provides both confidentiality and integrity in one pass, so we don't bolt a separate MAC onto a confidentiality-only cipher and risk getting the composition wrong. Tampered ciphertext is rejected, not silently decrypted.

Database encryption

Structured data — transcripts, generated minutes, metadata — is stored in **SQLCipher**, an encrypted build of SQLite. The database key is the HKDF-derived `sqlcipher-v1` subkey (§4); it is supplied at open time and is never written to disk in plaintext.

What is plaintext, and when

Meeting content exists in plaintext only transiently in app-private memory and temporary files during the live record → encrypt window, and again briefly when you open a meeting to read or export it. Temporary working files created during recording are removed once the encrypted artifact is written. There is no plaintext copy left behind in shared storage.

---

6. Network posture — the honest part

This is the section this audience reads first, so we put the whole truth here.

The core app makes zero outbound network calls carrying user content — ever. No telemetry, no analytics, no crash reporting that phones home, no "anonymous" usage counters. `usesCleartextTraffic` is disabled.

The app's manifest declares exactly these permissions:

- `RECORD_AUDIO`, `FOREGROUND_SERVICE`, `FOREGROUND_SERVICE_MICROPHONE`, `POST_NOTIFICATIONS` — the recording flow.
- `INTERNET`, `ACCESS_NETWORK_STATE` — gated to the opt-in model download described below; this is the app's only outbound network use.
- `com.android.vending.BILLING` — the Google Play Billing channel for the optional Pro purchase described below.

Exactly **two user-initiated uses** touch the network, and neither carries your audio, transcript, or minutes: an opt-in download of an on-device AI model from `huggingface.co`, and the optional Pro purchase through Google Play Billing. We describe each below.

The model download: `huggingface.co`

The `INTERNET` permission exists for exactly one thing the app does itself: **optional, opt-in, Wi-Fi-gated downloads of the on-device AI models** from `huggingface.co`:

- the **Gemma 4 E2B** minutes-generation model, and
- the optional **Whisper** `small.en` transcription model.

Both downloads are:

- **Opt-in** — the app works without them (the deterministic fallback handles minutes generation; a smaller default Whisper model handles transcription). Nothing downloads unless you choose it.
- **Wi-Fi-gated** — never on cellular.
- **Content-free** — these transfers pull a *public model file down* to your device. They never upload anything. None of your audio, transcript, or minutes is involved.
- **Integrity-checked** — each download is **SHA256-verified against a pinned HuggingFace commit hash** before first use; a mismatch deletes the file and re-downloads rather than trusting it.

After download, **inference runs entirely on-device** with no further network calls. We disclose this download rather than hide it: the alternative — shipping a multi-gigabyte model inside the app — would be worse for users, and pretending the permission isn't there would be dishonest. The permission is present; here is exactly what it is for.

Pro purchases — Google Play Billing

SumaFlow Minutes Pro is an **optional in-app purchase**, handled entirely by **Google Play Billing**. Buying it unlocks app features; it changes nothing about where your meeting content lives — Pro and free run the same on-device pipeline.

- **User-initiated only.** Nothing is purchased unless you choose to buy it. There is no background billing call and no recurring network chatter.
- **No payment data reaches us.** Google Play processes the payment. Your card details, billing address, and Google account are handled by Google and are **never seen by SumaFlow** — we run no payment server to receive them.
- **No account, no SumaFlow server.** Buying Pro creates no SumaFlow login and contacts no SumaFlow backend, because there is none. Your Pro entitlement is **derived locally on the device** from what Google Play Billing reports.
- **Content-free.** The billing channel carries no audio, no transcript, and no minutes — only what Google needs to record a purchase, none of which is your meeting content.

Google Play Billing is **Google's native channel**, running below the app layer: the app calls the Play Billing API, and Google Play services — a separate component of the device — carries out the purchase and any network it needs. The app does not make the payment call itself.

Export intents

The only other way data leaves the app is an **export you initiate** (§7). That uses Android's share/intent system to hand a single meeting to an app you choose; it is not a background network call and it is never automatic.

How this is enforced, not just promised

A privacy regression test (§8) boots the real app under an HTTP layer that **fails the test if any outbound request goes to a host other than** `huggingface.co`. The allowlist is enforced in an executable check that runs in CI and on real devices — not left to code review or good intentions.

This test covers the **app layer** — the outbound calls the app makes itself. Google Play Billing runs **below** that layer, inside Google Play services' own process, so the purchase is Google's native billing channel rather than an app-layer call the test would intercept. The guarantee the test enforces — that *the app* phones no host but `huggingface.co` — is unchanged by adding Pro.

7. Export transparency and the audit log

Every export is:

- **User-initiated** — nothing leaves automatically.
- **One meeting at a time** — there is no bulk silent export path.
- **Preceded by a confirmation screen** that shows exactly what will be sent and where, including the line *"This is the only way data leaves SumaFlow Minutes."*

Each export is also recorded in an **append-only on-device audit log**. The log stores, per export, what was exported and a **SHA-256 hash of the exported payload** (`content_sha256`) along with the destination and timestamp — a tamper-evident, on-device record of every time content left the app, visible to you in-app. The audit log never leaves the device and is not transmitted anywhere; it exists so *you* can see your own export history.

---

8. Open-source components

Promise 7 is *verifiable* privacy. The human-readable half of that promise is this whitepaper; the machine-verifiable half is the open-source **`sumaflow-minutes-privacy-core`** package (`github.com/SumaFlow-App/sumaflow-minutes-privacy-core`, MIT-licensed).

The package contains the code whose correctness backs promises 1–3:

- the **AES-256-GCM at-rest crypto layer** (the file format in §5),
- **key derivation and management** (the HKDF + Keystore design in §4), and
- the **no-network test harness** — an executable assertion (`PrivacyAssertingHttpOverrides`) that the app makes no outbound call to any host outside the `huggingface.co` allowlist.

You can run the no-network test yourself. It is the open-source artifact that turns promise 3 from a claim into something you can check; the repository README has the current module paths and the command to run it.

What is deliberately **not** in the package — and why — is documented in the package README: the SQLCipher wrapper (a thin shim over a third-party dependency), the Whisper FFI bindings, and the generated platform bridges are not privacy-critical and would add surface without adding verifiability.

9. What we explicitly do not do

- **No cloud.** No server holds your data because there is no server.
- **No account, no login, no sync.** Nothing to breach, phish, or correlate across devices.
- **No data sale.** We have no data to sell.
- **No ads, no ad SDKs.**
- **No third-party analytics or crash SDKs that phone home** — no Google Analytics, Firebase Analytics, Mixpanel, Amplitude, Crashlytics, or default-mode Sentry. Before any dependency is added, its network behavior is reviewed.
- **No background uploads, no "anonymous" telemetry, no launch counters.**

10. Threat-model boundaries — what this design does not cover

In the interest of not overclaiming:

- **A compromised or rooted device with the screen unlocked** can read what the running app can read. On-device encryption protects data at rest, not data in use on a hostile OS.
- **Shoulder-surfing / screen capture by someone with physical access** is outside the cryptographic boundary.
- **Your exports are yours to govern.** Once you export a meeting, that copy lives by the rules of its destination. The app makes the export explicit and logs it; it cannot recall it.
- **No recovery path.** There is no cloud backup and no key escrow. If you lose the device or its keystore (e.g., a factory reset), encrypted data is unrecoverable. This is the direct cost

of having no server — we think it is the right trade for this audience, and we state it rather than burying it.

- **Platform trust.** We rely on the Android Keystore, StrongBox where present, and the platform AES-GCM implementation. Weaknesses below our layer are outside what this app can remedy.

---

11. For regulated professionals — how this helps (and what it does not claim)

SumaFlow Minutes is **designed for** financial advisors, fiduciaries, and other regulated professionals who cannot route client conversations through a cloud meeting service. An on-device, no-cloud architecture can **help you meet your own confidentiality obligations** by keeping client meeting content on hardware you control, with no third-party processor in the path.

We are precise about the limits of that statement. SumaFlow Minutes is a tool; **compliance is a property of your practice, not of an app.** We do not claim the app is certified under, or compliant with, HIPAA, FINRA, SEC books-and-records rules, or any security-audit or information-security certification — we hold no such certifications, and claiming them would be exactly the kind of overstatement this whitepaper exists to avoid. What we offer is an architecture whose properties you can verify (§8) and reason about against your own obligations.

SumaFlow Minutes — Private AI minutes of meeting, on-device, no cloud.