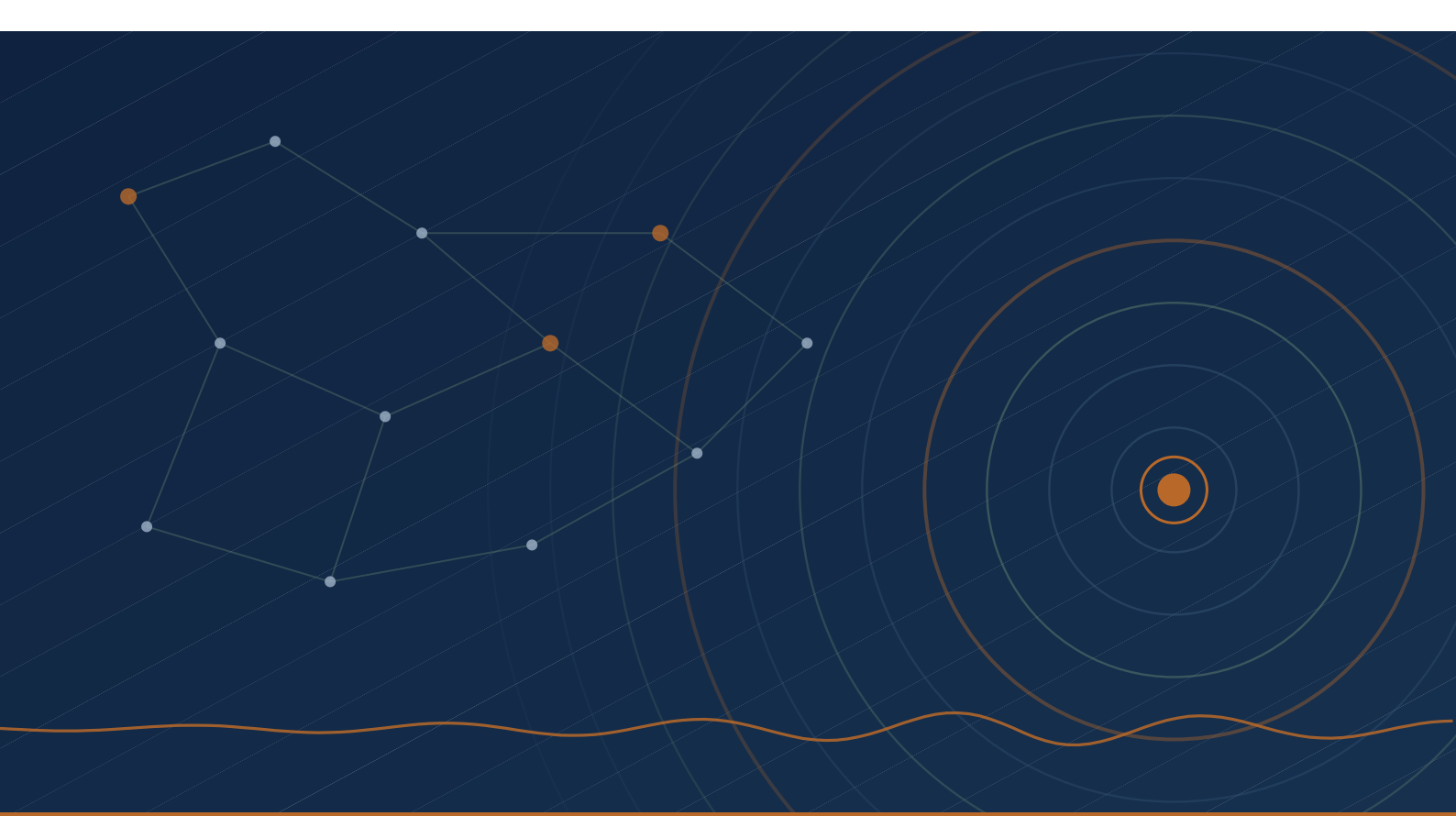




SANKET by Tosh Defence संकेत

Sovereign Secure Information Exchange Platform

Technical Security & Architecture Whitepaper

Cryptography | Zero Trust | Air-Gapped Deployment | Sovereign Infrastructure | High-Assurance Information Exchange

“The signal only you can read.”

Tosh Defence Private Limited

India

Document ID TD-SNK-WP-PUB-001 Version 2.0 Published 7 October 2026

Classification **PUBLIC**

Contact tosh@toshdefence.com

Document Control

Field	Value
Document title	SANKET by Tosh Defence - Technical Security & Architecture Whitepaper
Edition	Public edition
Document ID	TD-SNK-WP-PUB-001
Version	2.0
Publication date	7 October 2026
Prepared by	Tosh Defence Private Limited, India
Product	SANKET - Sovereign Authenticated Network Key-based Encrypted Transmission
Review status	Released for public distribution. Technical claims reviewed against the platform source code and engineering records of the release described in Chapter 2.
Intended audience	CISOs, CIOs, security architects, network security teams, technical evaluation committees, security assurance, risk and compliance teams, and procurement teams of organisations evaluating sovereign secure communications
Classification	PUBLIC
Business contact	tosh@toshdefence.com

PUBLIC EDITION

This is the public edition of the SANKET Technical Security & Architecture Whitepaper. It describes the security architecture, the cryptography, the threat model and the limits of the platform in full at the level of design. It deliberately omits component versions, detection and abuse thresholds, internal interface and configuration names, and the operating detail of protective features such as duress and remote wipe, because that detail helps an attacker more than an evaluator. A controlled edition containing it is provided to evaluating organisations under a non-disclosure agreement.

LICENCE

Copyright 2026 Tosh Defence Private Limited. This edition may be copied and redistributed in any medium or format, unmodified and with attribution to Tosh Defence Private Limited, under the Creative Commons Attribution-NoDerivatives 4.0 International (CC BY-ND 4.0) licence. Product and company names mentioned may be trademarks of their respective owners.

REPORTING A SECURITY ISSUE

Researchers who believe they have found a vulnerability in SANKET are asked to report it privately to security@toshdefence.com and to allow time for a fix to reach customer installations before any public disclosure. Reports are acknowledged and handled in confidence.

REVISION HISTORY

Version	Date	Author	Description
2.0	7 October 2026	Tosh Defence Private Limited	First public edition. Describes the release with end-to-end encrypted reactions, broadcasts and classification labels, administrator security keys, federated sign-in, signed device lists, device mutual TLS, hardware-backed key storage, calls over TCP 443, hybrid post-quantum transport, release freeze and rollback protection, release safety and the high-availability profile.

Legal and Security Disclaimer

This whitepaper describes the security architecture of SANKET as engineered by Tosh Defence Private Limited. It is provided for information and technical evaluation. It is not a warranty, a contractual specification or a certification of fitness for any particular purpose. Contractual commitments are made only in a signed agreement.

- **Architecture varies by deployment.** SANKET is self-hosted. The exact topology, network zones, identity integration, push-notification path and hardening baseline are chosen per deployment. Diagrams in this document are reference architectures, not a description of any particular installation.
- **Cryptographic configuration depends on the deployment baseline and release.** Algorithms, key sizes and protocol versions are stated for the release described in Chapter 2. Customers should confirm the configuration of the release they actually deploy.
- **Certification status must be verified against the current release.** At the date of this document SANKET holds no third-party security certification. Certification targets are identified as targets, not achievements (Chapter 40).
- **Security depends on correct configuration and operational discipline.** In a self-hosted deployment the customer operates the servers, networks, identity processes and endpoints. Weak operating practice can undermine any cryptographic design.
- **Capability status is labelled.** Every capability in this document is described as implemented, configurable, by design, deployment-dependent, planned, roadmap, recommended hardening or certification target. Planned and roadmap capabilities are not operational until delivered in a release.
- **Endpoints remain inside the security boundary.** End-to-end encryption protects data in transit and on the server. It does not protect plaintext on an authorised endpoint that an attacker fully controls (Chapter 29).
- **No absolute claims.** No responsible vendor can guarantee immunity from compromise. This document sets out controls, assumptions and residual risks so that the reader can form an independent judgement.

Product and company names mentioned in this document may be trademarks of their respective owners. Third-party specifications are cited for technical reference only. Their citation does not imply endorsement of SANKET by their authors.

How to Read This Document

The document is written for technical evaluators. Chapters 1 to 7 set out the problem, the threat model and the design principles. Chapters 8 to 20 describe the architecture, cryptography, identity and lifecycle of information in depth. Chapters 21 to 40 deal with deployment, operations, assurance and limits. The appendices collect reference tables, checklists, a glossary and the standards cited.

Capability status is shown with the following labels wherever precision matters:

Label	Meaning
IMPLEMENTED	Present in the platform release described in this document and covered by automated tests.
CONFIGURABLE	Implemented, and turned on, off or tuned by the customer administrator through tenant policy.
DEPLOYMENT-DEPENDENT	The property depends on how the customer builds and operates the deployment (network, hardware, PKI, processes).
BY DESIGN	A deliberate architectural choice. The reason is always stated alongside it.
PLANNED	Engineering work defined and scheduled, but not in the current release.
ROADMAP	Direction of travel without a committed release.
RECOMMENDED	Hardening guidance for the customer. Not a product feature.
CERTIFICATION TARGET	An assurance outcome Tosh Defence intends to pursue. Not achieved.

Footnotes cite primary sources: IETF RFCs, NIST publications, W3C and FIDO Alliance specifications, Signal's published protocol specifications and peer-reviewed research. The full list is in Appendix J.

Contents

Document Control.....2

Legal and Security Disclaimer.....3

How to Read This Document.....4

1 **Executive Summary.....10**

 1.1 The Sovereign Communications Problem.....10

 1.2 The SANKET Proposition.....10

 1.3 What This Document Establishes.....11

2 **Release Baseline and Capability Status.....13**

 2.1 Baseline Described.....13

 2.2 Capability Status Summary.....13

 2.3 Design Boundaries and Their Rationale.....19

3 **Who SANKET Is For.....20**

4 **Sovereignty as a Security Control.....22**

 4.1 Jurisdictional Sovereignty.....22

 4.2 Data Sovereignty.....22

 4.3 Cryptographic Sovereignty.....22

 4.4 Infrastructure Sovereignty.....23

 4.5 Identity Sovereignty.....23

 4.6 Operational Sovereignty.....23

 4.7 Update Sovereignty.....23

 4.8 Audit Sovereignty.....24

 4.9 Administrative Sovereignty.....24

 4.10 How the Security Posture Changes.....24

5 **Threat Model.....26**

 5.1 Assets.....26

 5.2 Trust Assumptions.....26

 5.3 Methodology.....27

 5.4 Threat-Control Matrix.....28

 5.5 Threat Model Conclusions.....31

6 **Security Architecture Principles.....32**

 6.1 Using Precise Terminology.....33

7 **System Architecture.....35**

 7.1 Client Applications and Supported Platforms.....35

 7.2 Server Components.....37

 7.3 Components Not Present.....38

 7.4 Control Plane and Data Plane.....38

8 **Service Architecture.....40**

 8.1 Why Separation Matters.....40

9 **Data Architecture.....43**

 9.1 Data Stores.....43

 9.2 Classification of Server-Held Data.....43

 9.3 Lifecycle of Stored Ciphertext.....44

10 **Cryptographic Architecture.....45**

 10.1 Transport Protection.....46

 10.2 End-to-End Message Encryption.....48

 10.3 Group Encryption.....51

 10.4 File Encryption.....51

 10.5 Endpoint Data at Rest.....52

 10.6 Server Data at Rest.....52

 10.7 Backup Encryption.....52

11	Key Management Architecture.....	54
11.1	Key Generation and Entropy.....	54
11.2	Hardware-Backed Storage.....	55
11.3	Key Inventory.....	56
11.4	Rotation, Expiry and Revocation.....	59
11.5	Compromised Key Response.....	59
11.6	Disaster Recovery of Keys.....	60
11.7	Hardware Security Modules.....	60
12	Forward Secrecy and Post-Compromise Security.....	61
12.1	Definitions.....	61
12.2	Worked Timeline.....	61
12.3	Behaviour in SANKET Specifically.....	62
12.4	Scope of the Properties.....	62
13	Identity and Authentication.....	64
13.1	Provisioning.....	64
13.2	Sign-In.....	65
13.3	Federated Sign-In.....	66
13.4	Sessions and Tokens.....	67
13.5	Brute-Force and Abuse Resistance.....	68
13.6	Local Unlock.....	68
13.7	Device Restore and Recovery.....	68
13.8	Signed Device Lists.....	69
13.9	Identity Boundaries.....	69
14	Device Trust Model.....	70
14.1	Device States.....	70
14.2	Registration.....	71
14.3	Companion Desktop Linking.....	71
14.4	Device Inventory and Remote Actions.....	71
14.5	Anti-Theft and Reset.....	72
14.6	Hardware-Backed Keys.....	72
14.7	Posture and Integrity.....	72
14.8	Forced and Recommended Updates.....	73
14.9	Managed Devices and BYOD.....	73
14.10	Device-List Integrity.....	73
15	Role and Policy Architecture.....	74
15.1	Administrator Roles (RBAC).....	74
15.2	Group Roles.....	75
15.3	Broadcast-Sender Accounts.....	75
15.4	End-User Policy Controls.....	75
15.5	Attribute-Based and Hierarchical Policy.....	76
16	Zero Trust Architecture.....	77
16.1	Evaluation Inputs.....	77
16.2	Cryptography as the Last Control.....	78
16.3	Device Mutual TLS.....	78
16.4	Integration with the Customer's Zero Trust Architecture.....	78
17	Message Security Lifecycle.....	79
17.1	Delivery Reliability.....	79
17.2	Content Envelope.....	82
17.3	Message Adjuncts.....	82
17.4	Reactions.....	82
17.5	Encrypted Broadcasts.....	82
17.6	Editing and Deleting.....	83
18	File Security Lifecycle.....	84
18.1	Broadcast Attachments.....	85
18.2	Viewing Files on Mobile.....	85
18.3	Revocation.....	85
19	Secure Voice and Video.....	86

19.1	Media Server.....	86
19.2	Frame Encryption (End to End).....	86
19.3	Key Rotation Rate.....	87
19.4	Transport Encryption (Client to SFU).....	87
19.5	Operation on 3G, 4G and Constrained Networks.....	88
19.6	Server Enforcement.....	88
19.7	Calls in Isolated and Restrictive Networks.....	88
19.8	No Recording.....	89
20	Metadata Security.....	90
20.1	Metadata Inventory.....	90
20.2	What SANKET Does to Minimise Metadata.....	91
20.3	Content Confidentiality Versus Traffic-Analysis Resistance.....	92
21	Audit Architecture.....	93
21.1	Events Recorded.....	93
21.2	Tamper Evidence.....	93
21.3	Access, Retention and Export.....	94
21.4	Administrators Cannot Audit Plaintext.....	94
22	Monitoring, Threat Detection and Observability.....	95
22.1	Always-Active Protections.....	95
22.2	Optional Stream Processor.....	95
22.3	What Is Not Claimed.....	95
22.4	Usage Analytics on the Installation.....	96
22.5	Observability Without Content Exposure.....	96
23	Air-Gapped Deployment.....	98
23.1	Definition.....	98
23.2	Replacing External Dependencies.....	98
23.3	Client Devices in an Enclave.....	99
23.4	Operational Considerations.....	100
24	Offline-First Operation.....	101
24.1	Store and Forward.....	101
25	Network Architecture and Segmentation.....	102
25.1	Exposed Surface of the Released Profile.....	102
25.2	Reference Architectures.....	103
25.3	Network and Protocol Categories.....	104
26	PKI Architecture.....	105
26.1	Customer Transport PKI.....	105
26.2	Client Trust in the Edge.....	106
26.3	Messaging Identity Is Not X.509.....	106
26.4	Vendor Signing Keys.....	106
27	Administrative Security.....	108
27.1	Central Management Console.....	108
27.2	Platform Controls.....	108
27.3	Lost Security Keys.....	110
27.4	Separation of Duties.....	110
27.5	Recommended Practices.....	110
28	Security Impact of Server Compromise.....	112
29	Endpoint Compromise Analysis.....	113
29.1	Mitigations in Combination.....	113
29.2	Desktop and Mobile.....	114
30	Software Supply Chain and Secure Updates.....	115
30.1	Supply-Chain Controls.....	115
30.2	Release Contents.....	116
30.3	Reference Update Process.....	117
30.4	Independent Verification by an Air-Gapped Customer.....	118
30.5	Rollback, Pinning and Emergency Patches.....	118

31	Offline Licensing.....	119
31.1	Licence Format.....	119
31.2	Validation.....	119
31.3	Expiry Behaviour.....	120
31.4	Design Notes.....	120
32	Availability, High Availability and Disaster Recovery.....	121
32.1	Denial of Service and Resource Exhaustion.....	121
32.2	Resilience of the Single-Host Profile.....	121
32.3	High-Availability Reference Topology.....	122
32.4	Behaviour When an Identity Provider Is Unavailable.....	122
32.5	Backup.....	122
32.6	Disaster Recovery Methodology.....	123
33	Hardware Security.....	125
34	Security Control Matrix.....	126
35	Defensive Attack Scenarios.....	128
35.1	Scenario 1: Untrusted public network interception.....	128
35.2	Scenario 2: Compromised SANKET application server.....	128
35.3	Scenario 3: Stolen employee device.....	128
35.4	Scenario 4: Malicious system administrator.....	129
35.5	Scenario 5: Replay of captured encrypted traffic.....	129
35.6	Scenario 6: Compromised user password.....	129
35.7	Scenario 7: Unauthorised device access attempt.....	129
35.8	Scenario 8: Database theft.....	130
35.9	Scenario 9: Backup media theft.....	130
35.10	Scenario 10: Supply-chain attack on an update package.....	130
35.11	Scenario 11: Lost device with cached messages.....	130
35.12	Scenario 12: User removed from a confidential group.....	131
36	Deployment Mode Comparison.....	132
37	Feature Catalogue.....	134
38	Information Classification Support.....	138
38.1	Classification Labels.....	138
38.2	Separation Between Security Domains.....	139
38.3	Policy Support Is Not Accreditation.....	139
39	Crypto-Agility and Post-Quantum Readiness.....	140
39.1	Crypto-Agility in the Current Architecture.....	140
39.2	National Cryptographic Modules.....	140
39.3	Post-Quantum Readiness.....	141
40	Cryptographic Certification Roadmap.....	143
40.1	Types of Requirement.....	143
40.2	SANKET's Approach.....	143
40.3	India-Specific Pathway.....	143
41	Compliance, Assurance and Security Testing.....	144
41.1	Assurance Status.....	144
41.2	Internal Security Reviews.....	144
41.3	Testing Performed.....	145
41.4	Recommended Testing Programme.....	145
41.5	Cryptographic Testing.....	145
42	Secure Deployment Baseline.....	147
42.1	Provided by the Platform.....	147
42.2	Operator Hardening Points.....	147
42.3	Customer Hardening Baseline.....	148
43	External Dependency Analysis.....	149
43.1	Security-Relevant Libraries and Runtimes.....	150
44	Privacy Architecture.....	152

44.1 Principles Applied.....152

44.2 Retention Schedule.....152

44.3 Personal Data Inventory..... 153

45 Security and Usability Trade-offs.....154

46 What SANKET Does Not Claim.....156

47 Why Air-Gapped SANKET Is Structurally Different.....157

47.1 Trust Removed.....157

47.2 Control Retained.....157

48 Comparative Architecture.....158

49 SANKET within the Tosh Defence Ecosystem.....160

50 Technical FAQ.....161

Appendix A - Cryptographic Algorithm Reference.....165

Appendix B - Key Hierarchy.....167

Appendix C - Security Control Matrix.....168

Appendix D - Threat Model Summary.....169

Appendix E - Deployment Reference Architectures.....170

Appendix F - Air-Gap Deployment Checklist.....171

Appendix G - Security Hardening Checklist.....172

Appendix H - Network and Protocol Categories.....173

Appendix I - Glossary.....174

Appendix J - Standards and References.....176

After editing, refresh this contents list in Microsoft Word: right-click it, choose Update Field, then Update entire table. Caption numbers refresh the same way (select all, then press F9).

1 Executive Summary

SANKET is a sovereign secure information exchange platform. It gives an organisation end-to-end encrypted messaging, file exchange, voice and video, and priority broadcast with acknowledgement, on infrastructure that the organisation owns, operates and can disconnect from the public internet.

1.1 The Sovereign Communications Problem

High-trust organisations increasingly depend on instant communications for operational decisions. Most of that traffic runs on services that the organisation neither owns nor controls. Encryption is now common in those services, but encryption on its own answers only one question: whether a third party on the network can read a message. It does not answer the questions that matter most to a defence ministry, an intelligence service or a critical infrastructure operator:

- **Jurisdiction.** Which government can compel the service provider to hand over data, change the software or stop the service?
- **Operational dependency.** Does the service keep working if the provider is unreachable, sanctioned, compromised or simply decides to change its terms?
- **Telemetry and analytics.** What does the software report back to its vendor, and to which third-party analytics, crash-reporting or advertising services?
- **Metadata.** Even when content is encrypted, who can see who talked to whom, when, from where and how often?
- **Cloud dependency.** Are content, keys, logs or backups held in infrastructure the organisation cannot inspect?
- **Disconnected operation.** Can the system run on a closed network, in an enclave or at a site without internet access?
- **Administrator compromise.** If an administrator account, or the vendor's own staff, is compromised, what can the attacker read or change?
- **Endpoint compromise.** What protection remains when a phone is lost, seized or infected?
- **Software supply chain.** How does the organisation know that an update is genuine, and who decides when it is installed?
- **Ownership.** Who owns the infrastructure, the keys, the identities and the audit trail?

A platform that encrypts content but leaves these questions to a foreign provider leaves the most valuable information about an organisation, its structure, its tempo and its relationships, outside its control.

1.2 The SANKET Proposition

SANKET is designed so that every one of those questions has an answer that sits inside the customer's own security boundary. Its security rests on six reinforcing properties:

Table 1: The six properties of the SANKET security proposition

Property	What it means in SANKET
Sovereignty	One dedicated installation per customer, run with Docker Compose on customer-chosen infrastructure. No vendor-operated service in the message path. No third-party analytics or crash-reporting SDK in any product application.
Cryptography	Messages, reactions, encrypted broadcasts and classification labels are encrypted with Signal's official, audited libsodium library (the current release on iOS, Android, desktop and server), using PQXDH key agreement with a mandatory post-quantum Kyber prekey and the Double Ratchet with its post-quantum companion ratchet. Files are encrypted on the device with AES-256-GCM. Transport is TLS with AES-256-GCM cipher suites only, and the edge offers hybrid post-quantum key exchange to clients that support it.
Identity	Customer-controlled identities identified by Sanket ID. Argon2id password hashing, TOTP with SHA-256 as the users' second factor, FIDO2 security keys for administrators, device-bound session tokens with single-use refresh rotation, and device lists signed by each account's own devices so that the server cannot silently add a device. Sign-in can be federated to the customer's own identity provider, never to a public one.
Isolation	A deployment needs no cloud service. It can run on a closed or air-gapped network with an internal certificate authority, self-hosted push for Android, offline licence validation and offline release bundles.
Auditability	A SHA-256 hash-chained audit log of security and administrative events, recorded by Sanket ID, verifiable on demand and exportable to the customer's SIEM over TLS-protected syslog. Audit records never contain message content.
Operational control	The customer decides when to install a release, what features are enabled, how long data is retained, which devices are trusted, and when a device is revoked, placed in lost mode or remotely wiped.

*Sovereignty + Cryptography + Identity + Isolation +
Auditability + Operational Control*

No single property is sufficient. Strong cryptography on a vendor-operated service still leaves jurisdiction and metadata exposure. A self-hosted service with weak identity controls still lets an attacker sign in. SANKET's position is that the combination is what produces a defensible security posture, and the rest of this document examines each part on its merits.

1.3 What This Document Establishes

- A threat model covering twenty-three threat actors, from passive network observers to a malicious system administrator and a compromised update package (Chapter 5).
- An end-to-end account of how messages, files and calls are encrypted, which keys exist, where each is generated and held, and what each protects (Chapters 10 to 12 and 19).
- A precise inventory of the metadata the server necessarily handles, and why (Chapter 20).
- An analysis of the effect of a server or endpoint compromise, and why compromise of the server yields no message content (Chapters 28 and 29).
- Reference architectures for single-site, segmented, air-gapped and resilient deployments, with the parts that are implemented separated from the parts that are recommended or planned (Chapters 23 to 26 and 32).
- An explicit statement of what SANKET does not claim (Chapter 46).

KEY POINT - HOW CLAIMS ARE MADE IN THIS DOCUMENT

Every technical statement in this whitepaper was checked against the SANKET platform source code at the release described in Chapter 2. Where a capability is planned, optional or dependent on deployment choices, the text says so, and every design boundary is stated with its rationale. Cryptographic statements were checked against the primary specifications cited in the footnotes.

2 Release Baseline and Capability Status

This chapter fixes the release that the whitepaper describes and summarises, in one place, the status of every major security capability and the design boundaries that an evaluator should understand.

2.1 Baseline Described

Table 2: Software baseline described in this whitepaper

Component	Baseline
Platform release	SANKET platform as built at the date of this edition (October 2026), including the versioned end-to-end encrypted content envelope, encrypted broadcasts, classification labels, administrator security keys, signed device lists, device mutual TLS and the high-availability profile
Mobile application	SANKET for iOS and Android (React Native, Expo)
Desktop application	SANKET for desktop (Electron, on a supported release line), companion device linked from a mobile device
Messaging cryptography	The current Signal libsignal release on iOS (LibSignalClient), Android (libsignal-client / libsignal-android), desktop and server (@signalapp/libsignal-client)
Media server	SANKET build of the open-source LiveKit server, pinned at a fixed source commit, patched to accept only the AEAD_AES_256_GCM SRTP profile and to restrict TURN relay peers to the node's own media address
Server runtime	Containerised services on Docker Compose: nginx edge, SANKET API and WebSocket, notifier, scheduler and analytics processor on a supported LTS runtime, PostgreSQL, Redis, MinIO, OpenBao, ntfy, Postfix, LiveKit
Deployment profile	Single-host ("Edge") profile, and a multi-host high-availability profile with an asynchronous second site, both on Docker Compose (Chapter 32). Kubernetes is on the roadmap.

This public edition does not state exact component versions. They are given to evaluators in the controlled edition.

Capabilities are verified by automated unit, route and integration tests, an end-to-end suite run against a running installation, release freeze and rollback tests, and media-server proof tests that perform real encrypted-media handshakes.

2.2 Capability Status Summary

Table 3: Status of principal security capabilities in the described baseline

Area	Capability	Status	Notes
Messaging	libsignal PQXDH + Double Ratchet, one ciphertext per recipient device, 1:1 and groups	IMPLEMENTED	One-time Kyber (ML-KEM family) prekeys served first, each once, with the last-resort Kyber prekey only when none remain; server verifies prekey signatures with libsignal
Messaging	Post-quantum ratchet (SPQR, "Triple Ratchet")	IMPLEMENTED	Provided by the current libsignal release, in which SPQR runs alongside the Double Ratchet for all sessions (Chapter 39)
Messaging	Versioned content envelope inside every libsignal message (messages, reactions, encrypted broadcasts)	IMPLEMENTED	Carries a sender-generated message id identical in every per-device copy and the optional classification label; mobile and desktop produce byte-identical envelopes (Chapter 17)
Messaging	Reactions end-to-end encrypted	IMPLEMENTED	Each reaction is a small libsignal message; the server stores only opaque per-device rows and each device computes the counts. A tenant option lets the server keep an emoji-free count for very large groups (Chapters 17 and 20)
Messaging	Sender identity visible to the server	BY DESIGN	An organisational platform authorises, rate-limits and audits every sender; the operator is the customer itself. Sealed sender is a roadmap item (Chapter 20)
Messaging	Identity pinning (trust on first use, fail closed) and safety-number verification	IMPLEMENTED	When a contact's identity key changes, every conversation shared with them shows a notice, also on devices that were offline at the time, and the contact stays unverified until the safety number is compared again (mobile and desktop)
Messaging	Signed device lists (self-hosted key transparency)	IMPLEMENTED	Each account's device list is signed with an account list key held on the primary device; the installation keeps an append-only log and refuses a list that does not extend the previous one; senders verify the signed list before encrypting on every fan-out path, so a device added without the account's signature receives nothing (Chapters 14 and 28)
Messaging	Server strips non-essential message metadata	IMPLEMENTED	Only the idempotency key and a link-preview dismissal are kept beside the ciphertext; replies, locations, contact cards, link previews and forwards travel inside it (Chapter 20)
Files	Per-file AES-256-GCM on device, key delivered inside the libsignal message	IMPLEMENTED	Chapter 18
Calls	1:1 and group voice and video through a self-hosted SFU	IMPLEMENTED	Group calls off by default

Area	Capability	Status	Notes
Calls	AES-256-GCM frame encryption with per-sender keys delivered over libsignal; SRTP AEAD_AES_256_GCM only	IMPLEMENTED	Server refuses any device or offer below the AES-256 profile; clients end any call whose media is not AES-256-GCM; every call, one-to-one included, has its own server-issued call id that names its media room and binds its frame keys (Chapter 19)
Calls	Key renewal: five new sender keys per second for every participant, plus a fresh key on every join and leave	IMPLEMENTED	Chapter 19
Calls	Operation on 3G, 4G and constrained links: Opus with DTX at 24 or 12 kbps, VP8 simulcast, automatic adaptation	IMPLEMENTED	Chapter 19
Calls	Media over TCP 443 and TURN over TLS	IMPLEMENTED	One public TCP 443 carries HTTPS and TURN, routed by TLS server name without decryption; TURN over TLS is terminated with TLS 1.3 and TLS_AES_256_GCM_SHA384 only, and relays only to the installation's own media address. On by default for internet-facing installations, off for air-gapped ones; UDP media remains the primary path (Chapter 25)
Clients	Native apps for Android, iOS, Windows, macOS and Linux; no end-user web client by design	IMPLEMENTED	Chapter 7
Transport	TLS 1.3 (TLS_AES_256_GCM_SHA384 only) and TLS 1.2 (ECDHE AES-256-GCM only); HSTS	IMPLEMENTED	TLS 1.2 retained only for client platform stacks that do not yet offer TLS 1.3, with the same AES-256-GCM-only policy; outbound syslog and webhooks use the same policy
Transport	Hybrid post-quantum key exchange at the edge (X25519MLKEM768 first, then X25519 and secp384r1)	IMPLEMENTED	Negotiated wherever the client platform's TLS stack supports it, with platform coverage widening on the roadmap. Cipher suites unchanged; the server's outbound TLS offers the hybrid group too (Chapter 39)
Transport	Certificate pinning	IMPLEMENTED	Desktop: configurable certificate pinning. Mobile: native pinning of the edge public key on Android and iOS (REST and WebSocket), configured in the customer build
Transport	Device mutual TLS: a client certificate for every approved mobile and desktop device	IMPLEMENTED	P-256 key generated on the device (hardware-backed on phones), issued under the customer CA with an opaque name, short-lived, revoked through a CRL at the edge; presented on REST, the chat WebSocket and media signalling. Configurable per tenant (Chapter 16)
Identity	Argon2id passwords, TOTP (SHA-256), device-bound tokens, refresh rotation with reuse detection	IMPLEMENTED	TOTP as the users' second factor; each code accepted once; TOTP secrets AES-256-GCM bound to the account; device restore requires a libsignal signature over a single-use server challenge

Area	Capability	Status	Notes
Identity	Administrator FIDO2 security keys (WebAuthn)	IMPLEMENTED	Verified by the installation itself with no external service; ES256 and EdDSA; user verification required; synced passkeys refused by default; suspected clones refused. Policy can require a security key for every administrator (Chapter 13)
Identity	Administrator sign-in through the customer's own OIDC or SAML 2.0 identity provider; LDAP check for end users at enrolment, restore and new-device sign-in	OPTIONAL	The identity provider is a first factor only, the local second factor is still required; public identity providers refused; no just-in-time accounts (Chapter 13)
Identity	SCIM 2.0 user provisioning; scoped Integration API	IMPLEMENTED	Users resource; OpenAPI documented
Devices	Device approval, caps, revocation, server-confirmed remote wipe, lost mode, locate, duress PIN, app passcode, wipe after failures, MDM managed configuration	IMPLEMENTED	Chapter 14
Devices	Device posture (root / jailbreak) enforcement	IMPLEMENTED	Local root and jailbreak detection on the phone, reported and audited, with configurable enforcement; MDM managed configuration remains the authoritative posture source
Devices	Hardware-backed device keys (Secure Enclave on iOS; StrongBox, or the TEE where absent, on Android) and Android Key Attestation	IMPLEMENTED	The libsignal store, device secrets, the device certificate key and the account list key are protected by a hardware key; protection level recorded per device and policy can require one; attestation verified offline against vendor roots shipped in the release, with no vendor service contacted (Chapters 11, 14 and 33)
Local data	SQLCipher database and AES-256-GCM file cache under a wrapped data-encryption key, mobile and desktop; decrypted viewer copies on mobile deleted when the viewer closes, when the app backgrounds and at start-up	IMPLEMENTED	Chapters 10 and 18
Groups	Group creation by users or, under the administrators-only policy, by administrators from the console, audited	IMPLEMENTED	Chapters 15 and 27
Messaging	Disappearing messages: one shared duration table, 30 seconds to 90 days	IMPLEMENTED	A forced tenant duration is a ceiling: the shorter of it and the user's own choice applies (Chapter 17)
Content handling	Clipboard copy and file forwarding and sharing policy, mobile and desktop	IMPLEMENTED	Configurable per tenant; enforced in the mobile and desktop apps, with documents still opening inside the app (Chapters 15 and 18)

Area	Capability	Status	Notes
Classification	Customer-defined ranked classification labels on messages and attachments, carried inside the end-to-end encrypted envelope	IMPLEMENTED	The server can neither read nor alter a label. Content above the lowest rank cannot be forwarded, exported or downloaded on mobile or desktop; an unknown label is treated as the highest rank; a forward never lowers a label. Conversation-level labels on the roadmap; separation between accredited levels by installation remains recommended (Chapter 38)
Broadcast	End-to-end encrypted broadcasts with priority, acknowledgement tracking and full-screen flash alerts	IMPLEMENTED	Sent from a dedicated broadcast-sender account's single approved desktop, encrypted for every recipient device with libsinal; priority, deadline and expiry are also inside the envelope, so the server cannot downgrade, move or replay them; lock-screen text is generic by priority; unacknowledged urgent and critical broadcasts re-notified at the deadline with an administrator alert (Chapter 20)
Broadcast	Notice broadcasts for non-sensitive announcements	BY DESIGN	Server-readable; sent and scheduled from the administration console and labelled "Notice (not end-to-end encrypted)" to sender and recipients (Chapter 20)
Audit	SHA-256 hash-chained audit log with periodic Ed25519 anchors signed in OpenBao, verifier, syslog (RFC 5424 over TLS), pull API, signed webhooks	IMPLEMENTED	Chapter 21
Detection	Account lockout, rate limiting, refresh-token reuse detection and real-time alerting to the administration console and the customer SIEM	IMPLEMENTED	Chapter 22
Analytics	First-party usage analytics with an organisation switch and an installation lock	IMPLEMENTED	Counts only, pseudonymous, never content, no third party; on by default. The organisation can switch it off (step-up authentication, high-severity audit) and a lock set at provisioning forces it off; when off, clients collect nothing and collected analytics can be deleted. Audit logs are not analytics and stay on
Server data	Non-exportable signing and wrapping keys in OpenBao; field encryption of secrets	IMPLEMENTED	Chapter 10
Supply chain	Ed25519-signed release manifest with per-image SHA-256, a sequence number, an expiry and per-platform minimum app versions; signed timestamp for connected upgrades; offline release package; automatic rollback	IMPLEMENTED	Install and upgrade refuse an unsigned, expired or lower-sequence manifest and any downgrade; two releases claiming one sequence are always refused; a running installation never stops because its manifest expires; CycloneDX SBOM verified on offline and connected upgrades (Chapter 30)
Release safety	Per-platform minimum app versions, encrypted restore point before every upgrade, health-gated cutover, encrypted outbox for text, attachments and voice notes	IMPLEMENTED	Below-minimum apps can be refused per platform (configurable); every upgrade failure rolls back automatically; calls survive an API-only upgrade (Chapter 30)

Area	Capability	Status	Notes
Containers	SANKET containers non-root with all capabilities dropped and a read-only root filesystem in the Edge profile; every image pinned by digest	IMPLEMENTED	Hardening exceptions are recorded and justified in the controlled edition (Chapter 42)
Licensing	Ed25519-signed licence verified offline; trusted-time rollback detection	IMPLEMENTED	A licence may state a minimum app version, enforced on every platform (Chapter 31)
Availability	Single-host profile with health checks; per-artefact AES-256-GCM backups (data keys wrapped by OpenBao, the recovery artefact's key by a passphrase) with Ed25519-signed manifests, verified before restore; scheduled database backups to NAS, S3-compatible storage or SFTP	IMPLEMENTED	Chapter 32
Availability	High-availability multi-host profile on Docker Compose with an asynchronous second site	IMPLEMENTED	Two edge hosts, N+1 API and WebSocket nodes, PostgreSQL streaming replication with fenced failover, Redis with Sentinel, distributed MinIO with site replication, three-node OpenBao, multiple media nodes. Reference topology designed for RPO 5 minutes and RTO 1 hour; each deployment's disaster-recovery drill records its measured figures. Kubernetes on the roadmap (Chapter 32)
Watermarking	Visible watermark in the desktop and mobile file viewers	IMPLEMENTED	Configurable; forensic (invisible) watermarking on the roadmap

2.3 Design Boundaries and Their Rationale

Every secure platform makes deliberate choices about what the server must process in order to deliver the service. The boundaries below are stated so that evaluators can scope risk precisely; each is accompanied by the reason for it and the controls available to the customer.

Table 4: Design boundaries, rationale and customer controls

Boundary	Rationale	Customer control	Chapter
Routing and account metadata (sender account and device, recipient devices, time, size band, membership) is visible to the server	Required to authorise, deliver, rate-limit and audit communications inside an accountable organisation; in a self-hosted deployment the operator is the customer. Sealed sender is on the roadmap	Retention windows; separation of database and audit access	20
Optional large-group reaction counts reveal who reacted to which message	For very large groups, a tenant option lets the server keep an emoji-free count per message so that such groups do not depend on every device holding every reaction	Leave the option off; disable reactions by policy	17, 20
Notice broadcasts are server-readable by design	An administrator announcement channel for non-sensitive content that can be scheduled from the console; it is labelled "Notice (not end-to-end encrypted)" to sender and recipients	Use encrypted broadcasts or conversations for sensitive content; notice retention is configurable (90 days by default)	20, 37
The optional identity vault stores an encrypted copy of the user's identity key	Lets users recover their account after device loss; protected by a key derived on the device (PBKDF2-HMAC-SHA256, 600,000 iterations, AES-256-GCM)	Enable or disable the vault by policy and licence	11, 13
Out-of-band assurance of a contact's identity comes from safety-number comparison	Device additions are covered by signed device lists; identity keys are pinned and any change is flagged until compared again, the model used by mainstream Signal-protocol deployments	Safety-number comparison; identity-change notice in every shared conversation; device inventories; administrator device approval	14, 28
Connected installations may use Apple push for iOS wake-ups	iOS background wake-up is available only through Apple; it carries no user data. The media server never uses public STUN: its address is set at installation	Run iOS without Apple push; air-gapped installations do not use it	19, 43
High availability across sites is asynchronous	Synchronous replication between distant sites would tie every write to the inter-site link; the reference topology is designed for RPO 5 minutes and RTO 1 hour	Each disaster-recovery drill records the measured figures; zero data loss across sites is not part of the built profile	32

3 Who SANKET Is For

SANKET is built for organisations whose communications would cause serious harm if they were read, altered, withheld or mapped by an outside party, and that cannot accept an outside party holding the infrastructure, the keys or the logs.

The common factor is not a sector. It is a security requirement: confidentiality and integrity must hold even if the communications provider, the hosting provider or the network is not trusted. That rules out designs in which a third party operates the service, holds the content or decides when it changes. The environments below share that requirement.

Table 5: Representative high-trust deployment environments

Environment	Typical driver	Most relevant SANKET properties
Defence and armed forces organisations	Operational security, command communications, dispersed units, contested networks	Self-hosted deployment, end-to-end encryption, end-to-end encrypted priority broadcast with acknowledgement, classification labels, device revocation and remote wipe
Intelligence and national security organisations	Compartmentalised communications, insider risk, counter-intelligence	Customer-held infrastructure and keys, audit metadata without content, duress and lost-device controls, usage analytics that can be locked off for the whole installation
Ministries and government departments	Policy deliberation, inter-departmental coordination, jurisdictional control	Sovereign deployment, administrator-provisioned identities, sign-in through the department's own identity provider, retention policy
Law-enforcement agencies	Case confidentiality, field coordination, integrity of instructions	End-to-end encryption, device trust, audit trail of administrative actions
Critical infrastructure operators (energy, water, transport)	Incident coordination during outages, segmented operational networks	Operation on closed networks, offline delivery queues, self-hosted push options
Telecom operators	Network operations and security operations coordination	Self-hosted deployment, integration API, syslog export to internal SIEM
Financial institutions and regulators	Market-sensitive information, regulatory confidentiality	Customer-controlled logs and retention, role-based administration
Research, aerospace and strategic manufacturing	Intellectual property, export-controlled programmes	Encrypted file exchange, classification labels that block forwarding, export and download, controlled user population
Executive leadership and board communications	Targeted surveillance of senior staff	End-to-end encrypted voice, video and messaging; identity-key change warnings
Incident-response and security operations teams	Out-of-band communications when primary systems may be compromised	Independent infrastructure, separate identity domain, offline-capable deployment
Allied and friendly governments	Need for a platform they can host and inspect themselves	No mandatory runtime dependency on the vendor; customer-operated update and licensing

KEY POINT - NOT A CONSUMER PRODUCT

SANKET has no public service that anyone can join, no global directory and no vendor-operated server that end users connect to. Users exist only inside an installation the customer controls, and the customer decides how they are admitted: by administrator provisioning, by directory provisioning, or by single-use invitation (the default). That removes several attack paths that consumer services have by design, including account creation by an arbitrary adversary, contact discovery across organisations, and vendor-side access to the user graph.

4 Sovereignty as a Security Control

Sovereignty is often treated as a hosting statement: the servers sit in a particular country. In security terms that is the least important part. What matters is who can read, change, switch off or observe the system, and under whose legal authority they can be compelled to do so.

A communications platform has many points of control besides the servers: the people who administer it, the keys that protect it, the identity system that admits users, the logs that record its use, the software updates that change its behaviour, and the third-party services it calls while running. An organisation is sovereign over its communications only when each of these points of control sits inside its own security boundary, or can be moved there.

SANKET treats each of these points of control as a separate design requirement. The sections below define nine dimensions of sovereignty, explain the threat each one addresses, and state how SANKET meets it.

4.1 Jurisdictional Sovereignty

Threat addressed: lawful or extra-legal compulsion of a foreign service provider to disclose data, insert access, or suspend service. Cloud and SaaS providers are subject to the laws of the countries in which they are incorporated and operate, which can reach data held anywhere in the world.

How SANKET addresses it: SANKET is installed and operated by the customer on infrastructure the customer chooses. Tosh Defence does not operate the customer's communications service, does not hold its data, and has no standing access to it. There is therefore no provider in another jurisdiction that can be served with a demand for the customer's content, metadata or logs.

4.2 Data Sovereignty

Threat addressed: data residing on, or transiting, infrastructure the customer does not control.

How SANKET addresses it: the relational database, cache, object storage, secrets store, media server and push relay all run inside the customer's installation. Message content and file content are end-to-end encrypted before they leave the sending device, so even the customer's own servers hold ciphertext. Metadata that the server must process (for example routing information and audit records) stays on the customer's servers. Chapter 20 lists exactly what that metadata is.

4.3 Cryptographic Sovereignty

Threat addressed: a third party holding, generating or escrowing keys; dependence on a provider's key management service; inability to inspect or replace the cryptography.

How SANKET addresses it: end-user identity and session keys are generated on the user's device and never leave it in usable form. Server-side signing and wrapping keys are generated inside the customer's own OpenBao secrets service and are marked non-exportable. TLS certificates can be issued by the customer's own certificate authority. No key is held by Tosh Defence. The message protocol is the published, open-source Signal Protocol implementation (libsignal), which the customer or its evaluators can inspect. The signed device-list log that lets senders check every account's devices is kept by the installation itself, never by an external key transparency

service, and Android hardware attestation is verified offline against vendor roots shipped in the release, with no runtime call to the platform vendor.

4.4 Infrastructure Sovereignty

Threat addressed: reliance on infrastructure that can be withdrawn, throttled, inspected or modified by an outside party.

How SANKET addresses it: a SANKET installation is a set of containers that run on the customer's servers with Docker Compose. It needs no cloud provider, managed database, managed queue, content delivery network or hosted media service. The video and voice media server is a self-hosted build of the open-source LiveKit server, built and pinned by Tosh Defence and run by the customer.

4.5 Identity Sovereignty

Threat addressed: an external identity provider deciding who may sign in, or being compromised so that an attacker can mint identities.

How SANKET addresses it: the customer decides how identities come into existence. Administrators can create accounts directly, the customer's own provisioning system can create them through the SCIM 2.0 interface, or users can register with a single-use invitation issued by the organisation. Registration mode is a tenant policy and the shipped default is invitation-only. Administrator sign-in can be federated to the customer's own OIDC or SAML 2.0 identity provider, and enrolment can be checked against its own LDAP directory; public identity providers are refused and the local second factor is still required. Administrator security keys are verified by the installation itself with no external service. There is no dependency on a public identity provider, and identities exist only inside the customer's installation. Each user is identified by a Sanket ID.

4.6 Operational Sovereignty

Threat addressed: the vendor or a provider being able to stop, degrade or reconfigure the service; dependence on the vendor's availability.

How SANKET addresses it: the customer operates every component. Licence validation is performed locally against a signed licence file (Chapter 31), so the service does not stop because a vendor licensing server is unreachable. Configuration, retention, security policy and feature enablement are set by the customer's administrators.

4.7 Update Sovereignty

Threat addressed: a vendor pushing a change into production without the customer's approval, whether through error, coercion or compromise of the vendor.

How SANKET addresses it: server releases are pinned container image versions plus a configuration bundle that the customer installs. A release changes the installation only when the customer's operator runs the upgrade. Each release manifest is signed and carries a sequence number and an expiry, so an old or replayed release is refused, and connected installations also check a signed statement of the newest release, so they cannot be held on an old one without noticing. Air-gapped installations receive releases as offline bundles that the customer can inspect, scan and stage before deployment (Chapter 30).

4.8 Audit Sovereignty

Threat addressed: audit records held by a third party, or invisible to the customer; inability to prove what administrators did.

How SANKET addresses it: the audit log is stored in the customer's database, hash-chained so that alteration or deletion can be detected, and exported only to destinations the customer configures, such as its own SIEM over syslog with TLS. No audit or telemetry data is sent to Tosh Defence.

4.9 Administrative Sovereignty

Threat addressed: vendor staff with privileged access to customer systems; administrators with more power than their role needs.

How SANKET addresses it: the tenant administration console is part of the customer's installation and is used by the customer's staff. Tosh Defence's own operator console manages licences and releases. It is a separate system with no access path to the customer's message store. Within the customer's installation, administrative power is divided into permission-scoped roles (Chapter 15) and every administrative action is audited.

4.10How the Security Posture Changes

The effect of these properties is easiest to see by comparing the questions an evaluator must ask about a conventional cloud-hosted communications service with the questions that remain for a sovereign SANKET deployment.

Table 6: Sovereignty dimensions compared

Sovereignty dimension	Conventional cloud architecture	SANKET sovereign deployment
Jurisdiction	Provider is subject to the laws of its home and operating countries; data can be compelled from it	No provider holds the customer's data; compulsion would have to be served on the customer itself
Data location	Provider-selected regions; replicas and backups may sit elsewhere	Customer-selected servers; backups are made by the customer's operator
Keys	Often held in a provider key management service	End-user keys on devices; server keys in the customer's OpenBao, non-exportable
Infrastructure	Shared multi-tenant provider infrastructure	Dedicated installation on customer hardware or a customer-chosen sovereign data centre
Identity	Provider or federated public identity service	Customer-controlled identities: administrator-created, invitation-based or SCIM-provisioned from the customer's own directory; optional federation only to the customer's own identity provider
Operations	Provider controls availability and configuration	Customer operates every component; licence checked locally
Updates	Provider deploys changes continuously	Customer decides when a pinned release is installed
Audit and logs	Provider-held logs, partial customer visibility	Hash-chained audit log in the customer's database; export only to customer destinations
Administration	Provider staff hold privileged access	No vendor access; customer administrators in permission-scoped roles

Sovereignty dimension	Conventional cloud architecture	SANKET sovereign deployment
External analytics and telemetry	Commonly present	None sent outside the installation; first-party usage counts stay on the customer's servers and can be switched off, or locked off for the whole installation
Foreign CDN and hosted fonts	Commonly present	Not used by product applications; desktop builds are checked for foreign hosts and blocked from them at run time
Internet connectivity	Required	Not required for the server installation; an air-gapped mode is supported (Chapter 23)

NOTE - SCOPE OF THE SOVEREIGNTY CLAIMS

These properties describe the SANKET server installation and the SANKET client applications. Two practical boundaries apply.

- Mobile operating systems: Apple iOS devices receive wake-up notifications through Apple's push service when the customer chooses to use it. SANKET sends only generic, content-free payloads on that path, and air-gapped deployments can operate without it (Chapter 23).
- Endpoint and app distribution: smartphones and their operating systems are made by third parties. Customers with the highest assurance requirements should use managed devices and enterprise distribution channels (Chapter 14).

5 Threat Model

A security architecture is only as meaningful as the threats it is evaluated against. This chapter states the assets SANKET protects, the trust assumptions it makes, and the threat actors it is designed to resist, with the residual risk that remains for each.

5.1 Assets

Table 7: Protected assets

Asset	Description	Primary property
Message content	Text and structured content of direct and group messages, reactions, encrypted broadcasts and classification labels	Confidentiality, integrity
File content	Documents, images, audio and video exchanged as attachments	Confidentiality, integrity
Call media	Voice and video frames	Confidentiality
End-user keys	libsignal identity, prekeys and session state; file and call keys; the account list key that signs the device list; device client-certificate keys	Confidentiality, integrity
Credentials	Passwords, TOTP secrets, session and refresh tokens, passcodes	Confidentiality
Communication metadata	Who communicates with whom, when, how often, from which device and network	Confidentiality
Organisation data	User directory, group membership, roles, policy	Confidentiality, integrity
Audit trail	Record of security and administrative events	Integrity, availability
Service availability	The ability to communicate when needed	Availability
Server secrets	Token signing secrets, field keys, OpenBao keys, TLS private keys	Confidentiality
Software integrity	The code that runs on servers and endpoints	Integrity

5.2 Trust Assumptions

- **Endpoints running genuine SANKET code are trusted to protect plaintext while in use.** A fully compromised endpoint is out of scope for content confidentiality (Chapter 29).
- **The server is not trusted with content.** SANKET is designed so that a hostile server learns no message, file, call, reaction, encrypted-broadcast or classification-label content, cannot forge messages and cannot silently add a device to an account. The server is trusted for availability, routing and enforcement of access policy, and it sees metadata (Chapter 20).
- **The network is untrusted.** All client traffic is TLS-protected and content is additionally end-to-end encrypted.
- **Cryptographic primitives behave as specified.** SANKET relies on libsignal, AES-256-GCM, X25519, Ed25519, HKDF, SHA-256, Argon2id and ML-KEM as published and analysed.
- **The customer operates the infrastructure competently.** Host hardening, physical security, backups, firewalling and key custody are customer responsibilities in a self-hosted deployment, supported by the baseline in Chapter 42.

- **Random number generation on endpoints and servers is sound.** Keys are drawn from operating-system cryptographic random number generators.

5.3 Methodology

Threats are organised by actor and mapped to the STRIDE categories of Spoofing, Tampering, Repudiation, Information disclosure, Denial of service and Elevation of privilege.¹ MITRE ATT&CK provides a common vocabulary for the adversary techniques involved, and is referenced where it adds precision.² For each actor the matrix records the objective, the attack surface, the principal controls and the residual exposure.

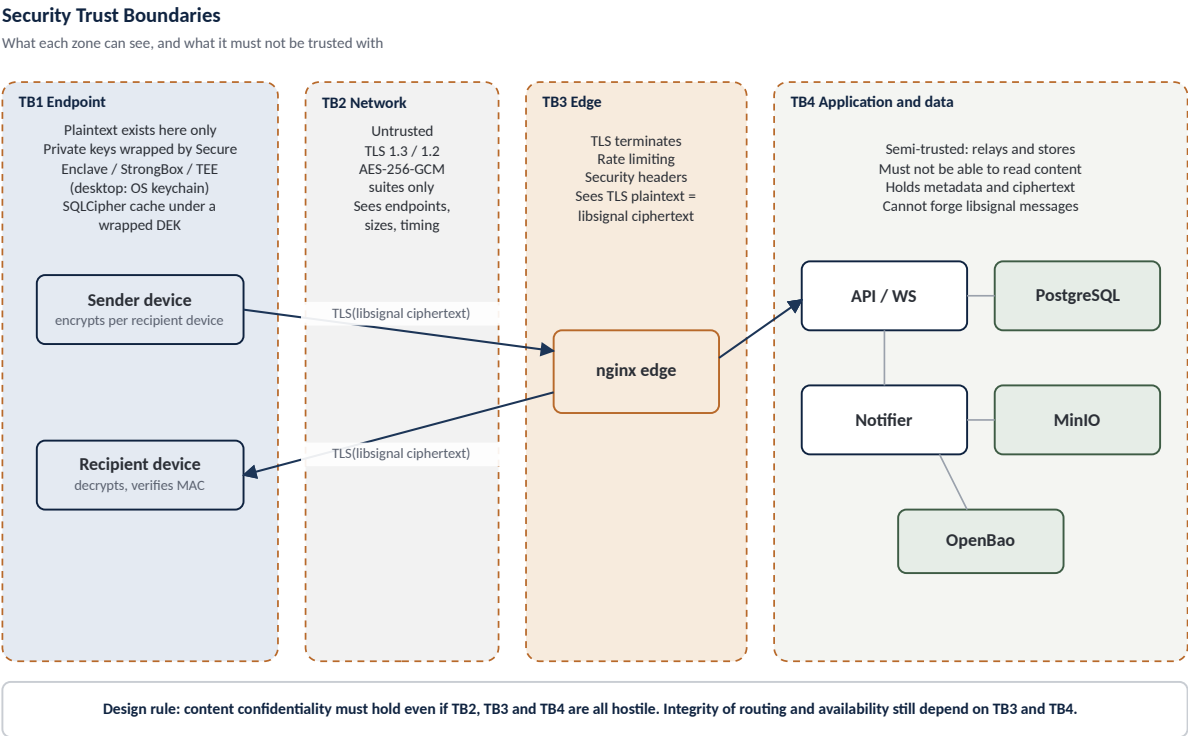


Figure 1: Security trust boundaries

The figure above summarises the four trust boundaries. The design rule is that content confidentiality must survive the loss of every boundary except the endpoint. Integrity of routing, enforcement of policy and availability do depend on the edge and the server, which is why those components are hardened and audited even though they never see content.

¹A. Shostack, "Threat Modeling: Designing for Security", Wiley, 2014 (STRIDE methodology).

²The MITRE Corporation, "MITRE ATT&CK Enterprise and Mobile Matrices", attack.mitre.org.

5.4 Threat-Control Matrix

Table 8: Threat-control matrix

Threat actor	Objective	Attack surface	Principal controls	Residual exposure	STRIDE
Passive network eavesdropper	Read content; map who talks to whom	Internet, Wi-Fi, mobile carrier, any link between device and edge	TLS with AES-256-GCM suites only; libsignal end-to-end encryption inside TLS; HSTS	Traffic analysis: timing, volume and the fact that a device talks to the installation	I
Active man-in-the-middle	Intercept or alter traffic; impersonate the server	TLS handshake, DNS, captive portals, rogue access points	Certificate validation against the customer CA; certificate pinning on desktop and mobile; end-to-end encryption means a TLS break exposes only ciphertext and metadata	Content stays end-to-end encrypted even if transport security is subverted	S, T, I
Compromised network device	Capture, redirect or drop traffic	Routers, firewalls, proxies inside customer or carrier networks	As above; end-to-end encryption; integrity of every libsignal message (MAC)	Denial of service; metadata	T, I, D
Malicious internet intermediary	Same as above at internet scale	Transit providers, DNS resolvers	Air-gapped or private-network deployment removes the path; TLS; E2EE	Availability and metadata for connected deployments	T, I, D
External attacker (remote)	Gain access to the service or its data	HTTPS edge, media port, exposed APIs	One public TCP 443 (HTTPS and TURN, routed by server name without decryption) plus the media UDP port; strict input validation; authentication on every API; device mutual TLS; edge and application rate limits; security headers	Undiscovered vulnerabilities in exposed code	S, T, E
Credential thief	Sign in as a user	Password phishing, reuse, keylogging	Second factor always required: FIDO2 security keys (WebAuthn) for administrators, bound to the customer's admin origin; TOTP for end users; per-account and per-IP rate limits; lockout; new devices subject to caps and optional admin approval	Bounded by the strength of the second factor; administrators use phishing-resistant security keys	S
Session token thief	Use stolen access or refresh tokens from another machine	Tokens copied from a device, a log or an intercepted session	Device mutual TLS: a client certificate whose key is generated on the device (hardware-backed store on phones) and never leaves it, verified by the edge and bound to the device record; single-use refresh tokens with reuse detection; short token lifetimes; device binding	Bounded by short token lifetimes, single-use refresh and device binding	S, E
Stolen endpoint attacker	Read messages and keys on a seized device	Physical device	OS full-disk encryption; app passcode and biometric lock; SQLCipher cache; protocol state and device secrets encrypted under a hardware key (Secure Enclave on iOS, StrongBox or the TEE on Android); wipe after failed attempts; lost mode; remote wipe	Forensic attacks on an unlocked or compromised OS	I
Malicious insider (end user)	Exfiltrate content they can legitimately read	Their own authorised device	Download and export policy; screenshot blocking and detection; forwarding controls; audit of file events	An authorised reader can always photograph or retype content	I, R
Privileged administrator	Read content; impersonate users; cover tracks	Admin console, database, hosts	No content keys on the server; RBAC with least privilege; mandatory administrator second factor; federated sign-in only from the customer's own identity provider and only as a first factor; senders verify signed device lists, so an administrator cannot silently add a device to a user's account; hash-chained audit log; separation of duties by role	Access to server-visible metadata (Chapter 20); mitigated by separation of duties	I, T, R, E

Threat actor	Objective	Attack surface	Principal controls	Residual exposure	STRIDE
Compromised application server	Read or alter messages; add a device; impersonate	API and WebSocket processes	libsinal E2EE: no plaintext or private keys on the server; message MACs; reactions, encrypted broadcasts and classification labels travel inside the end-to-end encrypted envelope; broadcast priority, deadline, sent time and expiry are also inside it, so the server cannot downgrade or replay a broadcast; senders verify each account's signed, append-only device list before encrypting, so a device the server adds receives no copy; identity keys pinned on first contact	Server-visible metadata (Chapter 20); withholding or delaying messages; denial of service	S, T, I, D
Compromised database server	Bulk theft of stored data	PostgreSQL storage and backups	Ciphertext only for message bodies, files, reactions and encrypted broadcasts (broadcast copies deleted per device once stored); Argon2id password hashes; encrypted secret fields	Server-visible metadata only (Chapter 20)	I
Compromised storage system	Read stored files	MinIO object store and its disks	Files encrypted on the device with per-file AES-256-GCM keys; objects are opaque	Opaque ciphertext only; availability	I, D
Rogue device	Join an account or conversation	Device registration, device linking	Device caps without silent eviction; optional admin approval; desktop linking only from a trusted phone with TOTP step-up and libsinal signature; every device addition signed into the account's device list by an approving device, and senders encrypt only for devices on that signed list; device inventory	Social engineering of an approver, countered by device inventories, notifications and audit	S, E
Replay attacker	Re-submit captured traffic	API requests, messages, tokens, webhooks	TLS replay protection; libsinal per-message keys and duplicate rejection; single-use refresh tokens and TOTP challenge tokens; idempotency keys; webhook timestamps	Minimal	S, T
Supply-chain attacker	Insert malicious code into a release	Dependencies, build system, release artefacts, update channel	Frozen lockfile builds; pinned upstream sources; Ed25519-signed release manifest with per-image SHA-256; per-image SBOM verified on offline and connected upgrades; offline verification; customer-controlled installation; pre-commit cryptographic policy checks	Compromise upstream of the build	T, E
Metadata analyst	Infer organisation structure and activity	Server database, network observation	Self-hosting keeps metadata inside the customer boundary; retention windows; masking of identifiers in edge logs; content-free push payloads	Inside the boundary, metadata is visible to operators (Chapter 20)	I
Brute-force attacker	Guess passwords, codes or passcodes	Login, TOTP, activation codes, device passcode, vault password	Argon2id; per-account and per-IP limits; logout; TOTP failure caps; passcode PBKDF2 with 600,000 iterations and wipe after N failures; vault PBKDF2 600,000 iterations	Bounded by password strength	S
Malicious software update	Run attacker code in the installation	Upgrade process	Customer decides when to upgrade; signed manifest and hash verification; staging; encrypted restore point and automatic rollback on any failed step	A signed release that is faulty rather than malicious; mitigated by staging	T, E
Freeze or rollback attacker	Hold an installation on an old release, or roll it back to a vulnerable one	Update channel, offline media, vendor registry responses	Signed manifest carries a monotonic sequence number and an expiry; install and upgrade refuse an expired manifest or one below the recorded sequence floor, and two releases claiming one sequence with different content; connected upgrades also require a short-lived signed timestamp statement under a separate key, refused if missing, expired, naming another manifest or older than the newest seen	An air-gapped installation learns of newer releases only through its own media	T, D
Lateral-movement attacker	Move from one compromised service to others	Container network, shared hosts	Only the edge published; OpenBao on an internal-only network; least-privilege service credentials; container hardening	Host-level compromise is equivalent to installation compromise (Chapter 42)	E

Threat actor	Objective	Attack surface	Principal controls	Residual exposure	STRIDE
Denial-of-service attacker	Make the service unavailable	Edge, media ports, expensive APIs	Layered edge and application rate limiting with adaptive protection and per-feature limits	Volumetric attacks beyond the customer's network capacity; the multi-host high-availability profile removes single points of failure	D
Restrictive network	Prevent calls by allowing only outbound TCP 443	Hotel, ministry, carrier or enterprise firewalls	TURN over TLS on the same public TCP 443 as HTTPS, terminated by the edge with TLS 1.3 and an AES-256-GCM suite only; relay limited to the node's own media address; media stays SRTP AES-256-GCM with end-to-end encrypted frames; UDP media remains the primary path where allowed	Higher latency over TCP; a network that blocks or intercepts TLS to the installation still denies service	D
Harvest-now, decrypt-later adversary	Record traffic today to decrypt it with a future quantum computer	Any recorded link between device and edge	Message content: libsignal with post-quantum key agreement (PQXDH) and ratchet (SPQR); transport: the edge offers hybrid X25519MLKEM768 key exchange first, negotiated wherever the client platform's TLS stack supports it; the server's outbound TLS offers it too	Message content is post-quantum protected by libsignal independently of transport; hybrid signatures for releases and licences are on the roadmap	I
Physical server theft	Take servers and their data	Data centre, edge sites	End-to-end encryption; OpenBao sealed at rest; recommended full-disk encryption with TPM	Depends on customer full-disk encryption (Chapter 42)	I
Backup theft	Recover data from backup media	Backup archives and media	Every component encrypted with its own AES-256-GCM key; database and object keys wrapped by the installation's OpenBao, the recovery key under a script passphrase-derived key; Ed25519-signed manifest verified before restore; content already end-to-end encrypted	Bounded by passphrase strength and custody	I
Endpoint malware	Read content as it is displayed or typed	Compromised phone or computer	Encrypted local storage; screenshot blocking where configured; managed devices; remote wipe; minimal local plaintext	High: E2EE does not protect a compromised authorised endpoint (Chapter 29)	I, T, E

5.5 Threat Model Conclusions

Three conclusions follow from the matrix and shape the rest of the architecture.

- 1. Content confidentiality is robust against network and server compromise.** Every network and server-side actor in the matrix obtains at most ciphertext of message bodies, files, reactions, encrypted broadcasts and classification labels, because the keys exist only on endpoints, and senders encrypt only for devices on each account's signed device list. This is the core property that end-to-end encryption delivers.
- 2. Metadata and availability are where server-side actors retain power.** A compromised server, database or privileged administrator can learn communication patterns and can deny service. Self-hosting confines that exposure to the customer's own staff and infrastructure, rather than eliminating it.
- 3. Endpoints and insiders remain the hardest problem.** No cryptographic design protects plaintext from a fully compromised authorised device or from an authorised reader. SANKET reduces the exposure with local encryption, device governance and policy, and is explicit that the residual risk is significant.

6 Security Architecture Principles

Twenty-four principles govern SANKET's design. Each is stated with the specific mechanism through which the current release applies it, so that the principle can be tested rather than taken on trust.

Table 9: Security architecture principles and how SANKET applies them

#	Principle	How SANKET applies it
1	Zero trust	No request is trusted because of where it comes from. Every API call carries a device-bound token and a device client certificate verified at the edge; the device's trust state is re-checked on every request, so revocation, lost mode or suspension takes effect almost immediately (Chapter 16).
2	Server-blind content, not "zero knowledge"	The server cannot decrypt message bodies, reactions, encrypted broadcasts, classification labels, file contents or call media because it never holds the keys. That is a precise and verifiable property. SANKET does not describe itself as a zero-knowledge system in the cryptographic sense, because an accountable organisational platform must know accounts, devices, membership and delivery metadata (Chapter 20).
3	End-to-end encryption	Content is encrypted on the sending device and decrypted only on recipient devices, using libsignal for messages, reactions and encrypted broadcasts and per-file AES-256-GCM keys for attachments. The server relays ciphertext.
4	Least privilege	Administrative capabilities are divided into permission categories with separate actions. Built-in roles grant the minimum for a job, and custom roles cannot exceed the permissions of their creator. Service credentials are scoped to the operations they need.
5	Defence in depth	Content is protected by libsignal inside TLS; local data by SQLCipher inside the operating system's storage encryption; calls by frame encryption inside SRTP. Access is checked at the edge, in middleware and in each route.
6	Minimise plaintext exposure	Plaintext exists only on endpoints. Notification payloads carry no content by default, and broadcast lock-screen text is generic by priority. The desktop holds keys in the main process, away from the renderer.
7	Minimise metadata	Push payloads carry only routing identifiers; Android wake-ups are sealed, padded to a fixed size and mixed with decoy traffic; edge logs mask identifiers in request paths; delivered ciphertext is tombstoned after a short retention window. Metadata that remains is stated in Chapter 20.
8	Explicit device trust	Devices are first-class security principals with states (pending, trusted, lost, revoked), per-platform caps, optional administrator approval, a recorded hardware protection level and an inventory that users and administrators can act on. Each account's device list is signed by its own devices.
9	Strong identity before access	Argon2id-verified password plus a second factor before any token is issued: TOTP (SHA-256) for users; a FIDO2 security key for administrators.
10	Cryptographic authentication	Device-link approvals are signed with the phone's libsignal identity key and verified by the server with libsignal. Device lists are signed with a per-account libsignal key and verified by every sender before encrypting. Prekeys are signed and verified. Administrator security keys sign a fresh WebAuthn challenge. Licences, release manifests and tenant status reports are Ed25519-signed.
11	Compartmentalisation	One installation per customer, with no shared database or tenant identifier columns. Within an installation, conversations are separate cryptographic sessions per device pair.
12	Failure containment	Cryptographic failures fail closed: a client refuses to send without libsignal encryption, a call ends if its media cannot be secured, a strict helper refuses any AES key that is not 256 bits.

#	Principle	How SANKET applies it
13	Tamper evidence	The audit log is a SHA-256 hash chain whose integrity can be verified on demand and whose links are carried into SIEM exports.
14	Secure by default	Defaults include TOTP required, invitation-only registration, member approval for groups, security keys for administrators, notification previews off, file download to device off, link previews off, message forwarding off and group calls off.
15	Customer-controlled deployment	The customer installs, configures, upgrades and administers every component.
16	Offline survivability	Licence checks are local; Android wake-ups can use the self-hosted push relay; clients queue outbound messages in an encrypted outbox when offline.
17	No hidden runtime dependency	Product applications contain no third-party analytics or crash-reporting SDK. Desktop builds are scanned for foreign hosts at build time and restricted to the tenant's hosts at run time. Exceptions, such as Apple push for iOS, are documented (Chapter 43).
18	Segmentation	Only the HTTPS edge and media ports are exposed; the secrets service is on an internal-only network. Finer zoning is a deployment recommendation (Chapter 25).
19	Revocation	Device revocation, lost mode, account suspension, refresh-token revocation and device certificate revocation are enforced by the server on the next request, and remote wipe is executed only after a confirmed revocation.
20	Auditability	Security-relevant actions by users, administrators, integrations and the system are recorded by Sanket ID with outcome and severity.
21	Crypto-agility	Call crypto is negotiated by profile and enforced by the server; prekey formats and the content envelope are versioned; the edge offers hybrid post-quantum key exchange first and classical groups after it; the release and licence trust bundles support key rotation and revocation (Chapter 39).
22	Administrative separation	Tosh Defence's licensing and release console is a separate system with no path into customer data. Customer administrators are separated by role.
23	Secure software supply chain	Frozen dependency lockfile, pinned upstream sources, pre-commit cryptographic policy checks and signed release manifests (Chapter 30).
24	Controlled update architecture	Releases are pinned versions installed by the customer, verified before load (signature, sequence and expiry), preceded by an encrypted restore point and rolled back automatically if the post-upgrade health and smoke checks fail.

6.1 Using Precise Terminology

Security documents often blur distinct properties under broad labels. SANKET uses the following distinctions throughout this whitepaper.

Table 10: Distinguishing confidentiality properties

Term	Meaning in this document	Applies to SANKET?
Server inability to decrypt (E2EE)	The server never holds keys that decrypt content	Yes, for message bodies, reactions, encrypted broadcasts, classification labels, file contents and call frames
Encryption at rest	Stored data is encrypted on disk, with keys held separately	Endpoints: yes (SQLCipher, AES-256-GCM cache). Server: content is E2EE ciphertext; disk encryption is a deployment control

Term	Meaning in this document	Applies to SANKET?
Administrative metadata	Accounts, devices, roles, policy, audit records	Visible to authorised administrators by design
Routing metadata	Sender, recipient device, conversation, time, size	Visible to the server; needed to deliver messages
Server-readable content by design	Content the server can read because the customer chose a channel that is not end-to-end encrypted	Notice broadcasts only, labelled as not end-to-end encrypted. With the large-group reaction option on, the server also learns who reacted to which message, never the emoji (Chapter 20)
Zero-knowledge proof / system	A protocol in which a party proves a statement without revealing anything else, or a service that learns nothing about its users	Not claimed. SANKET's server learns metadata

NOTE - WHY SANKET AVOIDS THE PHRASE "ZERO KNOWLEDGE"

The phrase is often used loosely to mean that the server holds only ciphertext of content. SANKET uses the more precise term server-blind content, because a messaging server necessarily learns who is communicating, and lists that metadata explicitly.

7 System Architecture

A SANKET installation is a self-contained set of services that a customer runs on its own infrastructure. Each customer has its own installation: there is no shared multi-tenant service and no vendor-operated component in the communication path.

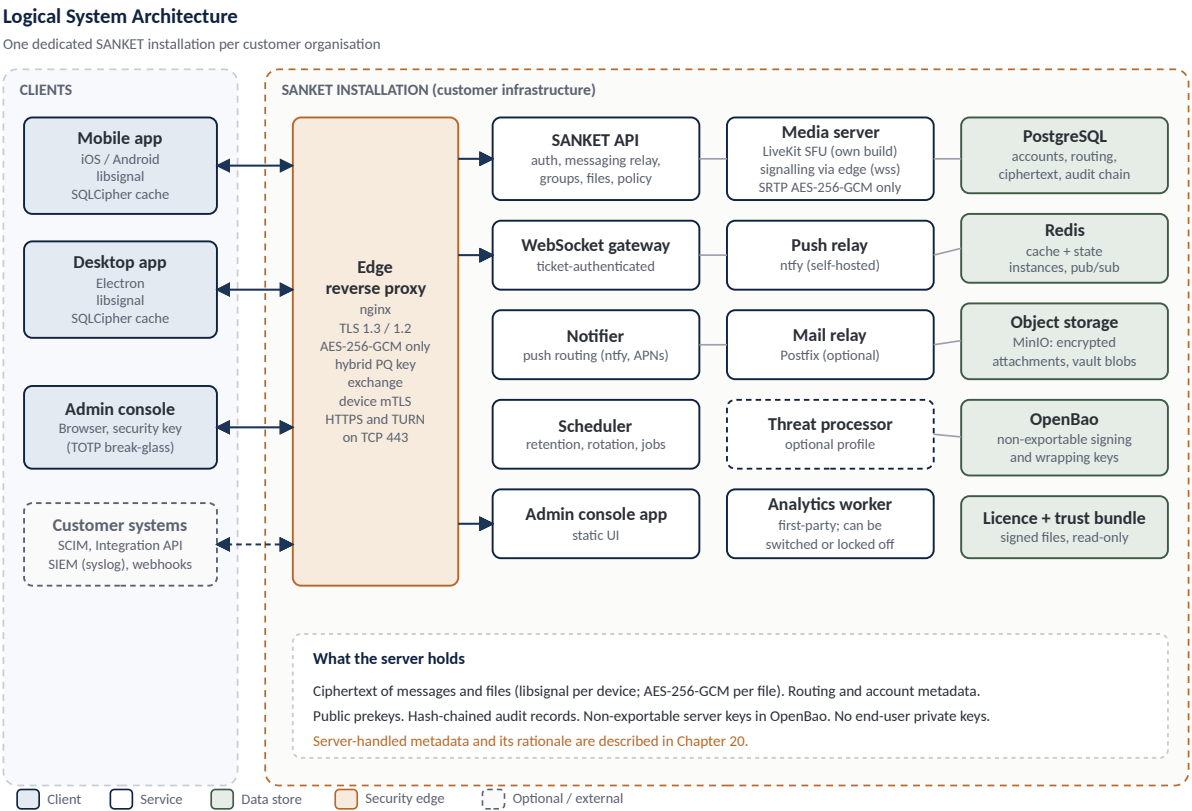


Figure 2: Logical system architecture

7.1 Client Applications and Supported Platforms

SANKET users work with native applications on phones and computers. There is no browser-based client for end users: browser messaging is deliberately excluded, because it would require running the Signal Protocol inside a web page served by the installation, which would let whoever controls the server change the cryptographic code delivered to users. The only browser application is the administrator console, used by administrators and never for communications.

Table 11: SANKET applications and supported platforms

Application	Platforms and minimum versions	Distribution	Role
SANKET for Android	Supported Android releases; built against the current Android SDK	Signed APK or app bundle for the customer's managed distribution or a private store listing	Primary device
SANKET for iOS and iPadOS	Supported iOS and iPadOS releases, including the current release	Apple App Store, Apple Business Manager custom app, or enterprise distribution	Primary device

Application	Platforms and minimum versions	Distribution	Role
SANKET for Windows	Supported 64-bit Windows releases	Signed installer (Authenticode); automatic updates from the customer's own installation, verified against the publisher	Companion device linked from a phone
SANKET for macOS	Supported macOS releases, on Apple silicon and Intel processors (separate native builds)	Signed and notarised disk image; automatic updates from the customer's own installation	Companion device linked from a phone
SANKET for Linux	Mainstream 64-bit distributions	ApplImage, .deb and .rpm packages	Companion device linked from a phone
Administrator console	Current desktop browsers	Served by the customer's installation	Administration only; no messaging

Minimum platform versions follow the supported range of the underlying platform SDKs, are reviewed with each release and are stated in the controlled edition.

Mobile (iOS and Android). The primary client, built with React Native on Expo. Signal Protocol operations run in a thin native module that calls the official libsignal Swift and Java libraries (the current audited libsignal release); no protocol logic is reimplemented in JavaScript. Protocol state and device secrets are encrypted with AES-256-GCM under a hardware key: on iOS a key derived from a Secure Enclave P-256 key, on Android a StrongBox key where the phone has one and a key in the trusted execution environment otherwise (Chapter 11). The local message and attachment cache is a SQLCipher database plus AES-256-GCM encrypted files. Each approved device also holds a client certificate for device mutual TLS, whose key is generated on the device and protected the same way (Chapter 26). Android devices receive wake-ups through the installation's own push relay; iOS devices use Apple's push service in connected deployments (Chapter 23).

Desktop (Windows, macOS and Linux). Built on a supported Electron release line, the desktop application is a companion client linked from a trusted phone. All cryptographic operations and key storage run in the application's main process using the official libsignal TypeScript binding. Its client certificate for device mutual TLS is presented from the main process; the user interface never sees the key. The desktop is also the only client that can send end-to-end encrypted broadcasts, from a dedicated broadcast-sender account, encrypting each broadcast for every recipient device in its main process (Chapter 17). The user interface is restricted by a runtime network allowlist that cancels any request to a host outside the tenant's own hosts, and desktop builds are checked for foreign hosts at build time.

Administrator console. A browser application served by the installation. Administrator sign-in always requires a second factor, a FIDO2 security key (WebAuthn) verified by the installation itself; the first factor can be a local password or, optionally, the customer's own OIDC or SAML identity provider (Chapter 13). Access tokens are held in memory; the refresh token is an HttpOnly, SameSite=Strict cookie scoped to the authentication path.

Mobile and desktop parity. Message sending and receiving, delivery and read state, attachments, calls and call signalling, session handling, retries and security prompts follow the same protocol semantics and API contracts on every platform. Platform differences are limited to user interface and native storage mechanisms.

7.2 Server Components

Table 12: Server components of an installation

Component	Implementation	Role
Edge	nginx with an OpenSSL release that supports ML-KEM, pinned by digest	The only public TCP listener (443). Routes each connection by its TLS server name without decrypting it: TURN to a TLS terminator in front of the media server, everything else to the HTTPS virtual hosts, with the client address carried over PROXY protocol for rate limits, the administrator network allowlist and audit. TLS termination (TLS 1.3 and 1.2, AES-256-GCM suites only; hybrid X25519MLKEM768 key exchange offered first), device client-certificate verification against the customer CA and its revocation list, HSTS and security headers, request rate limits, routing to internal services.
API	SANKET API (Node.js on a supported LTS line, Fastify), non-root container	Authentication (passwords, TOTP, WebAuthn security keys, federated sign-in against the customer's own identity provider), sessions, device management and the append-only log of signed device lists, key directory, message relay and storage, groups, files, calls signalling, broadcast metadata and per-device broadcast ciphertext, administration, integration interfaces.
WebSocket gateway	Part of the API	Real-time delivery. A connection opens with a single-use ticket that is consumed atomically and re-checked against device trust, revocation and suspension.
Notifier	SANKET notifier	Routes wake-up notifications per device through the self-hosted ntfy relay (Android) or Apple push (iOS, connected deployments).
Scheduler	SANKET scheduler	Retention and tombstoning, expired-message purge, prekey and device hygiene, idle-device revocation, licence and status tasks. In the high-availability profile each singleton job runs on one elected node.
Media server	SANKET build of a pinned LiveKit release	Selective forwarding unit for voice and video. Patched to accept only AES-256-GCM SRTP and to relay TURN traffic only to the node's own media address. Its TURN listener sits behind the edge's TLS terminator, so it holds no certificate key. Forwards end-to-end encrypted frames it cannot decrypt.
Push relay	ntfy (self-hosted) ³	UnifiedPush server for Android wake-ups. Sign-up, web interface and attachments disabled. Treated as an untrusted transport: payloads are sealed per device.
Mail relay	Postfix (optional)	Local, relay or direct delivery of administrative email such as lockout alerts. Off by default.
Relational database	PostgreSQL	Accounts, devices, public key directory, signed device-list log, per-device message, reaction and broadcast ciphertext, groups, policy, audit chain.
Cache and state	Redis, two instances, with authentication	An evictable cache instance for data that can be rebuilt and a persistent state instance for short-lived session and real-time delivery state.
Object storage	MinIO (S3-compatible) ⁴	Encrypted attachment objects and encrypted vault blobs.

³ntfy project, "ntfy - push notifications made easy" (self-hostable pub-sub notification server), docs.ntfy.sh.

⁴MinIO, Inc., "Server-Side Encryption" documentation, min.io/docs.

Component	Implementation	Role
Secrets service	OpenBao, internal network only	Non-exportable Ed25519 and AES-256-GCM keys for the installation's signing and key-wrapping operations, and issuance of short-lived device client certificates under the customer CA with their revocation list (Chapter 11).
Analytics processor	Usage analytics	Ships and runs on every installation by default; its health is monitored and alerted. It computes first-party, content-free usage rollups from a capped event stream inside the installation; no other service depends on it. The organisation can switch analytics off, and an installation-level lock set at provisioning can force it off.
Optional services	Threat processor	Optional rule-based processing of security events (Chapter 22).

7.3 Components Not Present

The architecture is also defined by what it leaves out. A SANKET installation has no dependency on a managed cloud database, managed queue, content delivery network, hosted identity provider, hosted media service, cloud key management service, third-party analytics, third-party crash reporting or vendor-operated relay. The application tier is deliberately consolidated into a small number of services on Docker Compose, which reduces the operational surface for self-hosted and disconnected customers. The same services run on one host in the single-host profile or across several hosts in the high-availability profile, also on Docker Compose (Chapter 32); Kubernetes packaging is on the roadmap.

7.4 Control Plane and Data Plane

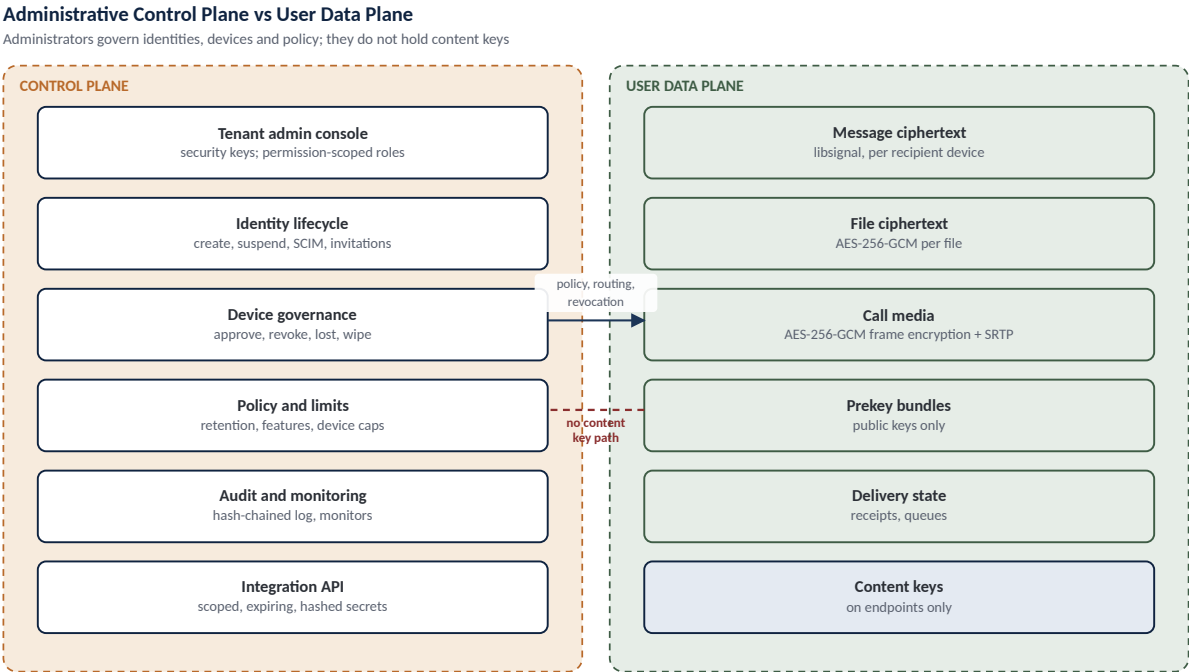


Figure 3: Administrative control plane versus user data plane

Administrators govern who exists, which devices are trusted, what features are enabled and how long data is kept. None of those actions gives an administrator a content key. The data plane carries ciphertext between endpoints, and the keys for that ciphertext are generated and held only on the endpoints. The separation matters most when an administrator account is compromised: the attacker gains the ability to change policy, revoke devices or read metadata, which is serious and is audited, but not the ability to read past or future message content directly.

NOTE - DEVICE GOVERNANCE AND CONTENT

Because the control plane decides which devices belong to an account, SANKET does not let it add a device silently. Each account's device list is signed by the account's own devices and kept in an append-only log, and senders verify that signed list before they encrypt, so a device the server or an administrator inserts receives no copy of any message, call key or broadcast. Device caps, optional administrator approval, identity pinning on first contact, device inventories visible to users, safety-number comparison and audit events complete the picture (Chapter 14).

8 Service Architecture

SANKET separates functions into containers with distinct privileges, so that the failure or compromise of one function has a bounded effect. This chapter describes each service's responsibility, the controls around it and the effect of its failure.

8.1 Why Separation Matters

- **Privilege separation.** The secrets service is reachable only from the services that need it, on an internal-only network; the edge never talks to it.
- **Blast radius.** Each service holds only the credentials it needs, so a defect in one service does not expose credentials held by another.
- **Independent scaling.** The media server, notifier and API have different resource profiles and can be sized independently. The Edge profile sets per-service CPU and memory limits.
- **Independent upgrade.** Each service is a separately pinned image in the signed release manifest.

The application tier is intentionally consolidated: authentication, messaging relay, groups, files, broadcast and administration share one API process. This keeps the number of moving parts, credentials and network paths small for customers who operate the platform themselves, at the cost of coarser internal isolation than a fully decomposed microservice design. Logical separation inside the API is enforced through route-level permission checks and distinct database access paths.

Table 13: Service responsibilities, controls and failure impact

Service	Responsibility	Security controls	Failure impact
Edge (nginx)	Server-name routing of TCP 443 to HTTPS or TURN without decryption; TLS termination for HTTPS and TURN; device client-certificate verification; rate limiting; headers	AES-256-GCM-only suites, TURN hop TLS 1.3 only; hybrid X25519MLKEM768 key exchange offered first; session tickets off; HSTS; strict CSP for API responses; client address preserved and forged forwarding headers from outside refused; customer CA and revocation list for device certificates; identifier masking in logs; connection and request limits	Service unavailable; no content exposure
Authentication (API)	Sign-in, TOTP, security keys, federated sign-in, tokens, sessions, logout	Argon2id; WebAuthn with ES256 and EdDSA only, user verification required, synced passkeys refused by default and sign-count regression refused; OIDC with PKCE and verified ID tokens, SAML with a pinned signing certificate and replay cache, always beside the local second factor; HS256-only verification of SANKET tokens; separate access and refresh secrets; single-use refresh with reuse detection; device certificate fingerprint bound to the device record; rate limits	Users cannot sign in; existing sessions continue until expiry
Messaging relay (API + WebSocket)	Accept per-device ciphertext, store, deliver, receipts	Active-membership checks; fan-out caps; idempotency keys; per-user send rate; tombstoning	Delivery paused; clients queue in encrypted outbox
Key directory and device lists (API)	Publish and serve prekey bundles; keep the append-only log of signed device lists	libsinal signature verification of signed and Kyber prekeys; trusted-device check on upload; consumption of one-time prekeys; a device list that does not extend the previous one is refused, and clients check the log themselves; rate limits	New sessions and new devices cannot be added; existing sessions continue
Files (API + MinIO)	Upload and download encrypted objects	Licence and tenant gates; size and type limits; active membership on download; lifecycle expiry	Attachments unavailable
Broadcast (API)	Encrypted broadcasts from a broadcast-sender account; clear notice broadcasts from the console; acknowledgement, deadline and escalation	Broadcast permission; named administrators for each broadcast-sender account; one approved sending desktop per account; recipient cap; generic push text by priority; audit	Broadcasts delayed
Calls signalling (API + Redis)	Room tokens, ringing, key-message relay	Membership checks; server-signed device identity in room token; AES-256 profile enforcement; one call per device	Calls cannot start
Media server (LiveKit)	Forward media between participants; TURN relay behind the edge	SRTP AES-256-GCM only (patched build with proof tests); TURN relay limited to the node's own media address (patched); no TLS certificate key; frame E2EE; token-gated rooms	Calls drop; messaging unaffected
Notifier + ntfy	Wake devices	Sealed, padded payloads; constant priority; decoys; no content	Wake-ups delayed; WebSocket delivery continues
Administration (API + console)	Users, devices, policy, monitors, audit	Mandatory administrator second factor; RBAC; role ceiling; audit of every change	Administration unavailable; users unaffected
Integration API, SCIM, syslog, webhooks	Machine interfaces for customer systems	Scoped expiring credentials stored as SHA-256; network allowlist; per-client limits; HMAC-signed webhooks; syslog over TLS with pinned CA	Integrations pause; core service unaffected
Scheduler	Retention, purge, hygiene, status	Least-privilege OpenBao token; internal network	Retention delayed; storage grows

Service	Responsibility	Security controls	Failure impact
OpenBao	Non-exportable signing and wrapping keys; device-certificate PKI	Internal-only network; Shamir unseal ceremony in production; least-privilege runtime token; device certificates carry an opaque random name and a short lifetime; three-node raft with TLS between nodes in the high-availability profile	Signing, backup wrapping and device-certificate issue unavailable
PostgreSQL / Redis	Persistent state; cache and live state	No host ports; authentication on Redis; Redis split into an evictable cache and a state instance that never evicts; streaming replication and Sentinel in the high-availability profile	Service unavailable
Analytics processor	Aggregate first-party usage events	Inside the installation, no third party; health monitored; can be switched off by the organisation or locked off at provisioning	Analytics stop; chat and calls unaffected
Threat processor (optional)	Rule-based detection on event stream	Separate database; configurable	Alerts from this source stop; built-in protections continue

9 Data Architecture

This chapter classifies every major category of data held by an installation according to what protects it. The classification is the basis for the server-compromise analysis in Chapter 28.

9.1 Data Stores

Table 14: Data stores and their contents

Store	What it holds	Protection
PostgreSQL	Accounts, devices (including app version and hardware protection level), public key directory, append-only log of signed device lists, per-device message and reaction ciphertext and routing data, groups and membership, policy, broadcast metadata and per-device encrypted broadcast copies, notice broadcasts, security-key public keys, call logs, audit chain, integration records	Content E2EE; secrets field-encrypted or hashed; no host port; disk encryption is a deployment control; in the high-availability profile, streaming replication with fenced failover and WAL archiving to the second site
Redis	Two instances: a cache for data that can be rebuilt, and a state instance for short-lived session, real-time delivery and call state. The usage-analytics event stream	Password authentication; internal network; append-only persistence on the state instance; Sentinel failover in the high-availability profile
MinIO	Encrypted attachment objects, including encrypted broadcast attachments (uploaded once); encrypted identity-vault blobs (versioned)	Client-side AES-256-GCM before upload; server-side AES-256-GCM at rest; lifecycle expiry for attachments; distributed per site with site replication in the high-availability profile
OpenBao	Non-exportable signing and wrapping keys; issuance of device client certificates	Sealed at rest; Shamir unseal; internal network only; in the high-availability profile, a three-node raft cluster with TLS between nodes and encrypted snapshots shipped to the second site
Licence and trust bundle	Signed licence and the public keys used to verify it	Ed25519 signatures; mounted read-only
Threat database (optional)	Security event stream and alerts for the optional processor	Separate PostgreSQL instance
Endpoint storage	Message history, attachments, protocol state, keys	SQLCipher, AES-256-GCM file cache, platform key stores (Chapter 11)

9.2 Classification of Server-Held Data

Table 15: Server-held data by protection class

Class	Examples	Readable by a database operator?
End-to-end encrypted content	Message bodies (libsignal ciphertext, one row per recipient device); reactions (opaque per-device rows); encrypted broadcasts, including the priority, deadline, sent time and expiry carried inside them; classification labels of messages and attachments; attachment objects; file names and types; vault blobs	No

Class	Examples	Readable by a database operator?
Server-encrypted secrets	TOTP secrets (AES-256-GCM, bound to the account); SMTP and SMS credentials and webhook secrets (AES-256-GCM); backup key wraps (OpenBao Transit)	Only with the corresponding server-held key
Secret-derived verifiers	Password hashes (Argon2id); vault proof verifier (SHA-256); invitation and activation code hashes; integration secret hashes (SHA-256)	Hash only; resistant to reversal subject to secret strength
Public key material	Identity keys, signed prekeys, Kyber prekeys, one-time prekeys, identity revocations; signed device lists and account list public keys; security-key public keys; device client certificates (opaque random names)	Yes, by design (public)
Account and organisation metadata	Sanket ID, display name, email, phone, profile fields, roles, group names and descriptions, membership, device information, app version, platform and build, hardware protection level	Yes
Communication metadata	Sender account and device, recipient device, conversation, timestamps, delivery status, ciphertext size band, call participants and durations; that a reaction was sent, by whom and to which devices, never the emoji; replies, link previews, locations, contacts and forwards are inside the ciphertext	Yes
Large-group reaction counts	Only when the tenant option for very large groups is on: an emoji-free count per message, so the server learns who reacted to which message but not with what	Yes
Broadcast metadata	Encrypted broadcasts: sender account, audience, priority, deadline, expiry and receipts	Yes
Notice broadcasts	Title, body and attachments of notice-type broadcasts from the administrator console, labelled to senders and recipients as not end-to-end encrypted	Yes
Audit metadata	Actor, action, target, IP address, user agent, device, result, hash links	Yes, to holders of the audit-read permission and database operators

9.3 Lifecycle of Stored Ciphertext

Message ciphertext is held on the server only as long as delivery requires. When every recipient device copy of a message has been delivered, the scheduler blanks the ciphertext, initialisation-vector field and any metadata of all copies after a short window, leaving a stub for ordering and receipts; stubs are later hard-deleted. Messages with a disappearing-message expiry are deleted when that time passes. Whole message partitions older than the hot-retention window are dropped. These windows are configurable within limits (Chapter 44). Reactions follow the same lifecycle as other messages. Each per-device copy of an encrypted broadcast is deleted as soon as that device confirms it has stored the broadcast, and any copy still undelivered is deleted when the broadcast expires.

SECURITY PROPERTY - EFFECT ON HISTORICAL EXPOSURE

Because delivered ciphertext is removed from the server, a later compromise of the database exposes far less history than the volume of traffic might suggest, and even that ciphertext is protected by the forward secrecy of the Double Ratchet (Chapter 12).

10 Cryptographic Architecture

SANKET layers several independent cryptographic mechanisms. Each protects a different asset at a different point, and each is described here as implemented in the baseline release, with the primary specification it follows.

Table 16: Cryptographic layers

Layer	Protects	Mechanism	Status
Transport	All client to server traffic, including the TURN media relay on TCP 443	TLS 1.3 (TLS_AES_256_GCM_SHA384) or TLS 1.2 (ECDHE with AES-256-GCM); hybrid post-quantum key exchange (X25519MLKEM768) offered first in TLS 1.3; the TURN relay hop TLS 1.3 with TLS_AES_256_GCM_SHA384 only	IMPLEMENTED
Device authentication at the edge	Which device a connection comes from	Mutual TLS with a per-device client certificate (ECDSA P-256 key generated on the device, issued by the installation's OpenBao PKI under the customer CA)	CONFIGURABLE
Message E2EE	Message, reaction and encrypted-broadcast content, including classification labels	The current audited libsignal release: PQXDH + Double Ratchet with SPQR; AES-256-CBC + HMAC-SHA256 per message; a versioned content envelope inside every message	IMPLEMENTED
File E2EE	Attachment content, names, types	AES-256-GCM with a random 256-bit key per file, key carried inside the libsignal message	IMPLEMENTED
Call E2EE	Voice and video frames	AES-256-GCM frame encryption with per-sender keys ratcheted by HKDF-SHA256 and delivered over libsignal	IMPLEMENTED
Call transport	Media between client and SFU	DTLS-SRTP with AEAD_AES_256_GCM only	IMPLEMENTED
Push payload	Android wake-up content	AES-256-GCM per device, fixed-size padding	IMPLEMENTED
Local storage	Data at rest on endpoints	SQLCipher (AES-256 page encryption, HMAC-SHA256 page authentication) and AES-256-GCM files under a wrapped 256-bit key; on phones the protocol store and device secrets encrypted with AES-256-GCM under a hardware-backed key (Secure Enclave on iOS, StrongBox or the TEE on Android)	IMPLEMENTED
Server secrets	Credentials and keys at rest on server	Argon2id hashing; AES-256-GCM field encryption; OpenBao Transit (non-exportable); identity-provider secrets in OpenBao	IMPLEMENTED
Signatures	Licences, releases, status reports, prekeys, device links, device lists	Ed25519; XEdDSA (libsignal), including the per-account device-list key	IMPLEMENTED
Administrator authentication	Administrator console sign-in	FIDO2 security keys (WebAuthn): ES256 (ECDSA P-256 with SHA-256) or EdDSA (Ed25519) assertions verified by the installation	IMPLEMENTED
Backups	Full-installation archives; scheduled database backups	Archives and scheduled backups: AES-256-GCM per artefact; the recovery key under a scrypt passphrase-derived key, data keys wrapped by OpenBao Transit; Ed25519-signed manifest	IMPLEMENTED

KEY POINT - AES-256 OR FAIL

SANKET's engineering policy is that every message, attachment, call frame, SRTP media stream and local cache is protected with AES-256, and that every TLS session the installation terminates or opens (the edge, the TURN relay hop and outbound connections) uses AES-256-GCM, with no AES-128 fallback and no compatibility mode. The policy is enforced in code: a shared helper refuses any AES-GCM key that is not exactly 32 bytes or any nonce that is not 12 bytes, a pre-commit check rejects code that constructs AES-GCM any other way, and the server refuses any call participant that does not support the AES-256 media profile.

The one upstream exception is the cipher suite of the WebRTC DTLS handshake, which protects no voice, video or message data; Chapter 19 explains why it is accepted and how every call records it.

10.1 Transport Protection

All client connections, REST and WebSocket, terminate at the edge over TLS.⁵ The edge configuration offers exactly the following:

Table 17: Edge TLS configuration

Parameter	Setting
Protocol versions	TLS 1.3 and TLS 1.2. Nothing below TLS 1.2.
TLS 1.3 cipher suites	TLS_AES_256_GCM_SHA384 only
TLS 1.2 cipher suites	ECDHE-ECDSA-AES256-GCM-SHA384 and ECDHE-RSA-AES256-GCM-SHA384 only ⁶
Excluded	AES-128 suites, ChaCha20-Poly1305, CBC-mode suites, static RSA key exchange
Key exchange groups	X25519MLKEM768 (hybrid X25519 plus ML-KEM-768, TLS 1.3 only) offered first, then X25519 and secp384r1. A client that offers the hybrid group negotiates it; a classical client is served classically and never refused
TURN relay on TCP 443	Terminated by the edge with TLS 1.3 and TLS_AES_256_GCM_SHA384 only; AES-128, ChaCha20 and TLS 1.2 refused on this hop. The edge reads only the TLS server name to route the TURN name to this terminator and everything else to the HTTPS hosts, without decrypting anything for that decision
Client certificates	Device client certificates are requested and verified on the device routes (see Mutual TLS below)
Session tickets	Disabled
HSTS	max-age of two years, includeSubDomains, preload ⁷
Other headers	Content-Security-Policy default-src 'none' for API responses; X-Frame-Options DENY; Referrer-Policy no-referrer
Outbound TLS	Syslog, webhook, SMTP relay and identity-provider connections from the installation use a minimum of TLS 1.2 with the same AES-256-GCM-only policy and refuse servers that offer only AES-128 or ChaCha20; the server runtime offers the hybrid post-quantum group on these connections too, and the SMTP relay's certificate is verified (optionally against a configured CA)
Negotiated-group log	The edge records only the negotiated key-exchange group name per connection, with no client address, URI or user, so the share of post-quantum connections can be measured
Administrator plane (vendor console)	TLS 1.3 only

⁵E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3", IETF RFC 8446, August 2018.

⁶E. Rescorla, "TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)", IETF RFC 5289, August 2008.

⁷J. Hodges, C. Jackson, A. Barth, "HTTP Strict Transport Security (HSTS)", IETF RFC 6797, November 2012.

10.1.1 Handshake, authentication and forward secrecy

In both TLS 1.3 and the permitted TLS 1.2 suites the session key is established with ephemeral elliptic-curve Diffie-Hellman, so a later compromise of the server's certificate private key does not decrypt recorded sessions. TLS 1.3 additionally encrypts the server certificate and most of the handshake. Server authentication is by X.509 certificate.⁸ TLS 1.2⁹ is retained solely for client platform network stacks that do not yet support TLS 1.3, and only with AEAD AES-256-GCM suites and forward secrecy. This conforms to current IETF guidance, which recommends TLS 1.3 and permits TLS 1.2 with AEAD suites and forward secrecy.¹⁰

In TLS 1.3 the edge offers the hybrid group X25519MLKEM768 first. Its shared secret combines an X25519 agreement with an ML-KEM-768 encapsulation,¹¹ so a recorded session stays confidential unless both are broken: an attacker who later obtains a quantum computer cannot recover it from X25519 alone, and a cryptographic weakness in ML-KEM cannot make the session weaker than X25519 alone; the residual risk is an implementation defect in the newer code. Clients whose TLS stack offers the group use it; others negotiate X25519 or secp384r1, and message content stays post-quantum protected by libsignal in either case (Chapter 39).

10.1.2 Certificates and private PKI

The edge certificate is installed in one of two modes. In automatic mode it is obtained by ACME DNS-01 validation, which needs no inbound port 80. In import mode the operator supplies a certificate and key issued by the customer's own certificate authority, which is the mode used for closed and air-gapped networks. Internal services can be given an additional trusted CA bundle. Chapter 26 describes the recommended PKI hierarchy.

10.1.3 Certificate validation and pinning

Clients validate the server certificate chain against their trust store; on managed devices that trust store carries the customer CA. The desktop client can additionally enforce certificate pinning: when pinning is required, the presented certificate's SHA-256 fingerprint must match a configured pin, and chain-validation failures are always rejected. Mobile clients pin the edge natively: the SHA-256 of the edge certificate's public key (SubjectPublicKeyInfo), or of its issuing CA, a current and at least one backup pin, compiled into the signed app for the deployment's own domain and its subdomains, the media server included. On Android the platform network security configuration enforces them for every connection; on iOS App Transport Security enforces them for the HTTPS API, and a native check in the WebSocket layer enforces the same pins for the real-time connection. A handshake that presents no pinned key is refused before any request is sent, and the app refuses to transmit at all if the native pinning layer is absent.¹²

10.1.4 Mutual TLS

Every approved mobile and desktop device holds its own client certificate. The device generates an ECDSA P-256 key pair (on phones inside the hardware-backed store, Chapter 11), builds the certificate signing request itself and sends only the request; the installation's OpenBao PKI issues a short-lived certificate under the customer CA. The certificate carries an opaque random name and no user or device identifier, because in TLS 1.2 a client certificate travels unencrypted. Devices present it on the REST API, the chat WebSocket and the media signalling connection on iOS, Android and desktop; on desktop it is used only by the Electron main process and the renderer never sees the key. The edge verifies the certificate against the PKI CA and its certificate revocation list, and the API binds the

⁸D. Cooper et al., "Internet X.509 Public Key Infrastructure Certificate and CRL Profile", IETF RFC 5280, May 2008.

⁹T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", IETF RFC 5246, August 2008.

¹⁰Y. Sheffer, P. Saint-Andre, T. Fossati, "Recommendations for Secure Use of TLS and DTLS", IETF RFC 9325 (BCP 195), November 2022.

¹¹NIST, "Module-Lattice-Based Key-Encapsulation Mechanism Standard" (ML-KEM), FIPS 203, August 2024.

¹²OWASP Foundation, "Pinning Cheat Sheet", cheatsheetseries.owasp.org.

verified fingerprint to the device record, so a token stolen from one device cannot be used with another device's certificate.

Enforcement is configurable per tenant. Device-bound tokens continue to be checked on every request (Chapter 13); the certificate is an additional, independent proof of the device. Chapter 14 describes the certificate lifecycle and Chapter 26 its place in the PKI.

10.2 End-to-End Message Encryption

SANKET uses Signal's official libsignal library¹³ for all message encryption, at the same release on every platform: the Swift package on iOS, the Java/Android packages on Android, and the Node.js package on desktop and server. On mobile a thin native module calls the library; on desktop it runs in the Electron main process. The server uses libsignal only to verify signatures: on uploaded keys, device-link approvals, restore challenges and signed device lists. SANKET contains no reimplementations of the Signal Protocol and no alternative message cipher.

10.2.1 Keys per device

Table 18: libsignal key material per device

Key	Type	Lifetime in SANKET	Purpose
Identity key pair	Curve25519 (X25519 for agreement; XEdDSA signatures) ¹⁴	Long-term; account-wide rotation supported with proof of the old key and TOTP step-up	Authenticates the device in key agreement; signs prekeys, device-link approvals and restore challenges
Signed prekey	Curve25519, signed by identity key	Rotated regularly	Medium-term key for asynchronous session start
Kyber prekeys	Kyber-1024 KEM keys, signed by identity key	One-time Kyber prekeys, replenished automatically, deleted by the server when handed out and by the device once used; plus a last-resort key, rotated regularly with the signed prekey and used only when no one-time key remains	Post-quantum component of the initial key agreement
One-time prekeys	Curve25519	Uploaded in batches and replenished automatically; deleted by the server when handed out	Adds a single-use DH contribution to session start
Account list key pair	Curve25519 (XEdDSA signatures), a separate libsignal key pair per account	Long-term; generated on the primary device; sealed in the encrypted vault backup for recovery	Signs the account's device list, which sends verify before encrypting (Chapter 13)
Registration ID	Random integer	Per installation of the app	Detects reinstallation
Session state	Root key, chain keys, ratchet keys	Per session, continuously ratcheted	Derives a fresh key for every message

The server accepts a prekey bundle only from a device in the trusted state, refuses legacy or malformed bundles, and verifies both the signed-prekey and Kyber-prekey signatures against the identity key with libsignal before

¹³Signal Messenger LLC, "libsignal" (Rust implementation with Java, Swift and TypeScript bindings), AGPL-3.0, github.com/signalapp/libsignal.

¹⁴T. Perrin, "The XEdDSA and VEdDSA Signature Schemes", Signal specification, Revision 1, October 2016. signal.org/docs/specifications/xeddsa/

storing them. Every bundle must contain the Kyber last-resort prekey; one-time Kyber prekeys are verified in the same way, and the server refuses an id that is already published for the device with another key.

10.2.2 Session establishment: PQXDH

Sessions are established with PQXDH, the post-quantum extension of X3DH.¹⁵¹⁶ The initiator fetches the recipient device's bundle and computes three or four X25519 agreements¹⁷ (identity-to-signed-prekey, ephemeral-to-identity, ephemeral-to-signed-prekey, and ephemeral-to-one-time-prekey when one is available) plus a Kyber encapsulation to one of the recipient device's one-time Kyber prekeys, or to its last-resort Kyber prekey when none remains. The shared secret is derived from all of these inputs with HKDF.¹⁸ The first message carries the initiator's identity and ephemeral public keys and the KEM ciphertext, so the recipient can complete the agreement while offline when the message was sent.

$$SK = KDF(DH(IK_A, SPK_B) \parallel DH(EK_A, IK_B) \parallel DH(EK_A, SPK_B) \parallel DH(EK_A, OPK_B) \parallel SS)$$

where SS = Kyber shared secret encapsulated to PQPK_B (PQXDH, Revision 3)

The security properties follow from the specification. The X25519 components provide mutual authentication and classical secrecy; the Kyber component means that an adversary who records traffic today and later obtains a quantum computer cannot recover the session key from the recorded handshake alone. Authentication remains classical: PQXDH does not protect against an active quantum adversary impersonating a party. PQXDH has been formally analysed.¹⁹ The server hands out each one-time Kyber prekey once and the device deletes it after use, so the post-quantum component of a handshake is single-use, and devices replenish their stock automatically.

10.2.3 Message encryption: the Double Ratchet

After session start, messages are encrypted with the Double Ratchet.²⁰ A symmetric-key ratchet advances a chain key with HMAC-SHA256 for every message, producing a fresh message key; a Diffie-Hellman ratchet mixes a new X25519 agreement into the root key whenever the direction of conversation changes. Each message key is expanded with HKDF into an AES-256 key, an HMAC-SHA256 key and an IV; the message is encrypted with AES-256²¹ in CBC mode with PKCS#7 padding and authenticated with HMAC-SHA256²² computed over the identity keys and the message. Message keys are deleted after use. Out-of-order messages are handled by retaining skipped message keys within libsignal's limits.

The libsignal release in use also runs Signal's Sparse Post-Quantum Ratchet (SPQR) alongside the Double Ratchet for all sessions, mixing ML-KEM-derived secrets into message keys, a combination Signal calls the Triple Ratchet.²³ An attacker must therefore break both the elliptic-curve and the post-quantum components to recover message keys (Chapter 39).

¹⁵E. Kret, R. Schmidt, "The PQXDH Key Agreement Protocol", Signal specification, Revision 3, May 2023 (updated January 2024). signal.org/docs/specifications/pqxdh/

¹⁶M. Marlinspike, T. Perrin, "The X3DH Key Agreement Protocol", Signal specification, Revision 1, November 2016. signal.org/docs/specifications/x3dh/

¹⁷A. Langley, M. Hamburg, S. Turner, "Elliptic Curves for Security" (X25519, X448), IETF RFC 7748, January 2016.

¹⁸H. Krawczyk, P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", IETF RFC 5869, May 2010.

¹⁹K. Bhargavan, C. Jacomme, F. Kiefer, R. Schmidt, "Formal verification of the PQXDH Post-Quantum key agreement protocol for end-to-end secure messaging", USENIX Security 2024.

²⁰T. Perrin, M. Marlinspike, "The Double Ratchet Algorithm", Signal specification. signal.org/docs/specifications/doubleratchet/

²¹NIST, "Advanced Encryption Standard (AES)", FIPS PUB 197 (updated 2023).

²²H. Krawczyk, M. Bellare, R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", IETF RFC 2104, February 1997.

²³Signal, "Signal Protocol and Post-Quantum Ratchets" (Sparse Post-Quantum Ratchet, SPQR), signal.org/blog/spqr/, October 2025.

10.2.4 Content envelope

Inside every libsignal message the plaintext is a versioned content envelope with three kinds: an ordinary message, a reaction (the target message id and the emoji, or its removal) and an encrypted broadcast. The envelope carries a message id generated by the sender, identical in every per-device copy, and an optional classification label for the message and for each attachment (Chapter 38), so the label is end-to-end encrypted and the server can neither read nor alter it. An encrypted broadcast's priority, acknowledgement deadline, sent time and expiry are also inside its envelope, so a hostile server cannot downgrade a critical broadcast, move its deadline or replay an old copy: clients refuse a copy whose envelope disagrees with the server's record. The sender is always the libsignal-authenticated sender of the message, never a field in the content. Mobile and desktop produce byte-identical envelopes.

10.2.5 Message authentication, replay and identity

- **Integrity and origin.** Every message carries a MAC under a key that only the two session endpoints can derive; any modification causes decryption to fail.
- **Replay.** A replayed message fails because its message key has already been consumed; libsignal reports it as a duplicate and SANKET discards it.
- **Identity pinning.** The first identity key seen for a peer device is stored; a later different key is refused by the client's identity store (trust on first use, then fail closed).
- **Signed device lists.** Before encrypting, a sender checks the recipient account's device list, signed by that account's list key, and sends no copy to a device that is not on it (Chapter 13).
- **Verification.** On mobile, users can compare a 60-digit safety number, or scan it as a QR code. Each party's half is derived with SHA-256 from that party's account identifier and libsignal identity key, contributing 96 bits of hash output per party.
- **Identity-change notice.** When a contact's identity key changes, the server tells everyone who shares a conversation with them. Each of those conversations shows "<name>'s security code changed. Verify it before sharing sensitive information." with a link to verification, and the contact is marked unverified until the safety number is confirmed again, on mobile and desktop alike. A device that was offline when the key changed receives the notice when it next connects, because the server keeps it for each recipient device until that device acknowledges it. The notice only informs: libsignal's own trust handling is unchanged, and an event about someone the user shares no conversation with is ignored.
- **Session repair.** A session is reset only after repeated decryption failures on recent messages, excluding duplicates and missing-session errors; the peer is told through a rate-limited, membership-checked channel.

10.2.6 No fallback

A client refuses to send if libsignal encryption cannot be performed; devices that do not publish a libsignal identity key are skipped rather than sent plaintext; legacy plaintext markers are rejected; and the server rejects key bundles and devices that do not follow the libsignal format. There is no plaintext or alternative-cipher path for message content: no client derives a static per-conversation key, and conversation previews show only what libsignal decrypted on the device.

10.3 Group Encryption

SANKET encrypts group messages by pairwise fan-out: the sending device encrypts the message separately for every trusted device of every active group member, each through its own libsignal session, and submits all copies in one request. It does not use libsignal Sender Keys.²⁴ The consequences are:

- **Membership changes need no re-keying.** A removed member's devices are simply no longer in the fan-out list. Membership is locked inside the send transaction so a message is never addressed to a device that has just left.
- **Each copy has full Double Ratchet properties**, including forward secrecy and post-compromise recovery per device pair.
- **Authenticity is per sender device**, because each copy is authenticated by its own session.
- **Cost grows with group size times devices.** The server caps fan-out size, and the group size limit is configurable.
- **Compromised member handling.** Revoking a compromised member's device removes it from all future fan-outs immediately. Messages that device already decrypted are not recoverable by any design.

Multi-device delivery follows the per-device session model of Signal's Sesame specification: each device has its own address and its own sessions.²⁵ The list of a member's devices is supplied by the server, but every account's device list is signed by that account's list key, and the sending device verifies the signed list before it encrypts on every fan-out path (one-to-one and group messages, reactions, identity rotation, call frame keys, locate, encrypted broadcasts and the file keys inside messages). A device the server lists but the signed list does not receive no copy, and the user is told why. The server can still withhold delivery, which is an availability matter (Chapters 14 and 28).

Reactions and encrypted broadcasts use the same pairwise fan-out. A reaction is a small libsignal message to every device in the conversation. An encrypted broadcast is encrypted by the sending desktop of a broadcast-sender account for every recipient device, each through its own Double Ratchet session, with no sender keys (Chapter 15).

10.4 File Encryption

Attachments are encrypted on the sending device before upload. For each file the client generates a random 256-bit key and a random 96-bit IV, optionally compresses the file, and encrypts it with AES-256-GCM, producing IV, ciphertext and a 128-bit authentication tag.²⁶ File name and media type are encrypted separately. The file key and file identifier are placed inside the message body, which libsignal encrypts to each recipient device. Because every file has a fresh random key, a GCM key is never reused, which removes the nonce-reuse risk that limits GCM.²⁷ Integrity is the GCM tag; a modified object fails to decrypt. Clients never send the file key to the server, the server discards any copy it receives, and a file is always opened with the key carried in its message. Chapter 18 describes the full lifecycle.

²⁴Signal Messenger LLC, libsignal source, group cipher (Sender Key) implementation, github.com/signalapp/libsignal (rust/protocol/src/group_cipher.rs).

²⁵M. Marlinspike, T. Perrin, "The Sesame Algorithm: Session Management for Asynchronous Message Encryption", Signal specification, April 2017. signal.org/docs/specifications/sesame/

²⁶M. Dworkin, "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", NIST SP 800-38D, November 2007.

²⁷M. Dworkin, "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", NIST SP 800-38D, November 2007.

10.5 Endpoint Data at Rest

Each user's local database on mobile and desktop is a SQLCipher database²⁸ keyed with a random 256-bit data-encryption key (DEK). The mobile app refuses to open the database unless it can prove at run time that SQLCipher is compiled in, which prevents a silent fallback to unencrypted SQLite. Cached attachments, thumbnails, voice notes and avatars are stored as AES-256-GCM files under the same DEK. On iOS the attachment, thumbnail and voice files, and the short-lived decrypted copies a viewer needs, additionally carry the Complete data-protection class, so they are unreadable while the device is locked. Android keeps app files in credential-encrypted storage.

The DEK is never stored in the clear. It is wrapped with an envelope: an ephemeral X25519 key pair is agreed with the user's device X25519 public key, the shared secret is passed through HKDF-SHA256 with a salt bound to the Sanket ID and tenant, and the result wraps the DEK with AES-256-GCM. The device private key is in the phone's secure store with this-device-only accessibility, itself encrypted with AES-256-GCM under a hardware-backed key: on iOS a key derived from a Secure Enclave P-256 key (ECDH and HKDF-SHA256), on Android a StrongBox key where the phone has one and a TEE key otherwise (Chapter 11). On desktop, the corresponding secrets are held in the operating system's protected credential store and can be wrapped a second time under a passcode-derived key held only in memory. On sign-out the DEK is zeroed and the per-user keys deleted; a remote wipe erases the databases, files and keys.

10.6 Server Data at Rest

- **Content** on the server is already end-to-end ciphertext, so its confidentiality does not depend on server-side encryption.
- **Passwords** are hashed with Argon2id (64 MiB memory, 3 iterations, parallelism 4).²⁹
- **SMTP and SMS credentials and webhook secrets** are encrypted with AES-256-GCM under a server-held key.
- **TOTP secrets** are encrypted with AES-256-GCM under a key held outside the database, derived from a dedicated installation secret, and each ciphertext is bound to its account, so it cannot be moved to another account. The server refuses to start without that secret.
- **Signing and wrapping keys** live in OpenBao as non-exportable Transit keys.³⁰ The device certificate authority is an OpenBao PKI mount whose signing key stays inside OpenBao.
- **Identity-provider secrets** are held in the secrets service, never in the tenant configuration. Administrator security keys are stored only as public keys with their sign counters.
- **Volumes.** Database and object-store volumes rely on disk encryption provided by the host, which is part of the deployment baseline (Chapter 42). Object-store server-side encryption is configured with a KMS secret but is not relied upon for confidentiality, because objects are client-encrypted.

10.7 Backup Encryption

The backup tooling writes the database dumps, the attachment bucket, OpenBao data and configuration as separate artefacts, each under its own AES-256-GCM key. The recovery artefact has its key wrapped under a key derived from an operator passphrase with scrypt; every other artefact's key is wrapped by the installation's OpenBao. A manifest of SHA-256 checksums is signed with a non-exportable Ed25519 key, and the restore tool verifies the signature and every artefact before changing anything. Because message and file content inside the archive is already end-to-end encrypted, the backup never contains readable communications.

²⁸Zetetic LLC, "SQLCipher Design" (AES-256 page encryption with HMAC page authentication), zetetic.net/sqlcipher/design/.

²⁹A. Biryukov, D. Dinu, D. Khovratovich, S. Josefsson, "Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications", IETF RFC 9106, September 2021.

³⁰OpenBao project (Linux Foundation), "OpenBao: secrets management, encryption as a service and privileged access management", openbao.org.

Scheduled backups use envelope encryption. Each artefact is encrypted with its own randomly generated 256-bit key in AES-256-GCM; that key is wrapped by the installation's OpenBao Transit engine under a non-exportable key, and only the wrapped form is stored. Each run produces a manifest (source, copies, sizes, SHA-256 checksums, wrapped keys) signed with a non-exportable Ed25519 key in the same OpenBao and stored next to every copy. A restore verifies the manifest signature against that key before anything else, then the artefact checksum, and the authentication tag protects the data while it is decrypted; a manifest signed by any other key, an edited manifest or an altered artefact is refused. A stolen set of backup copies is therefore useless without the installation's OpenBao. Adapter credentials are themselves stored encrypted by OpenBao. Chapter 32 covers backup operations.

11 Key Management Architecture

Every key in SANKET has a defined origin, location, purpose and lifetime. The organising principle is simple: keys that protect content are created and held on endpoints; keys that protect the installation are created and held inside the customer's installation; Tosh Defence holds only the keys that sign licences and releases.

Key lifecycle management follows the general model of NIST SP 800-57: generation, distribution, storage, use, rotation, revocation and destruction are defined for every key class.³¹

Key Hierarchy and Lifecycle

Where each class of key is generated, held and rotated

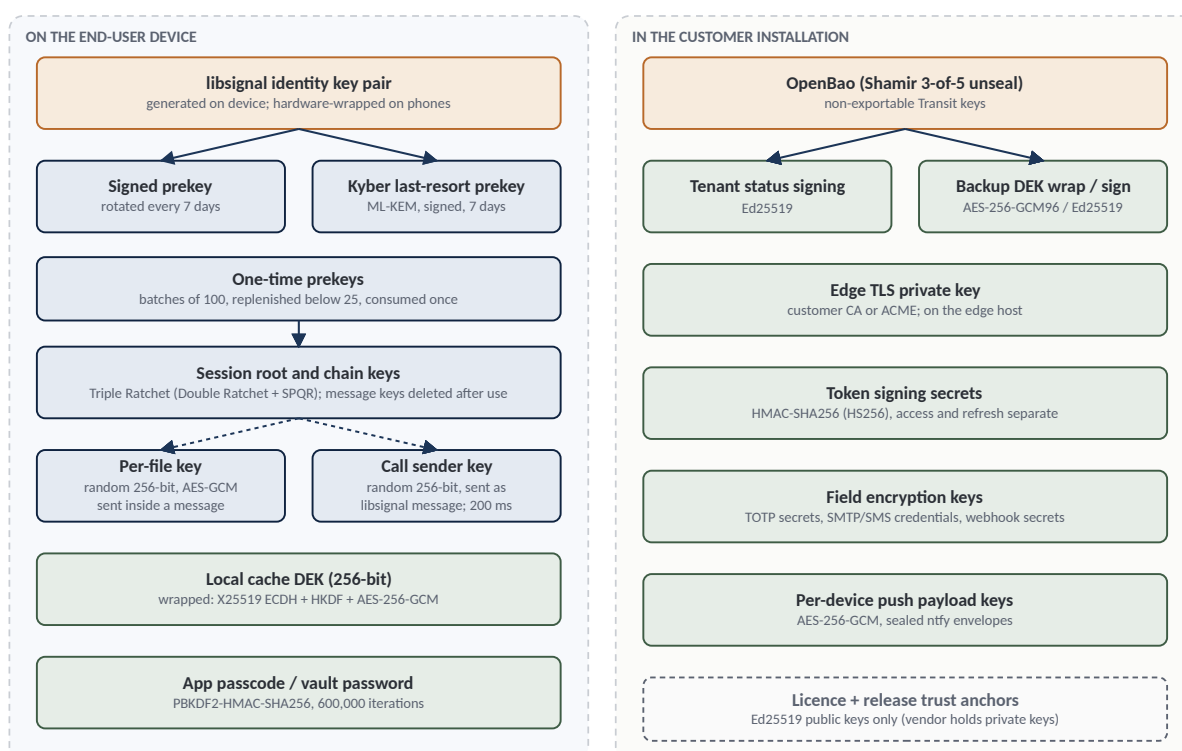


Figure 4: Key hierarchy and lifecycle

11.1 Key Generation and Entropy

All keys are generated from operating-system cryptographically secure random number generators: libsignal draws from the platform CSPRNG; the mobile JavaScript environment installs the platform CSPRNG as its random source before any cryptographic library loads; Node.js keys use its OpenSSL-backed generator; OpenBao generates its own Transit keys. No key is derived from a password except where a password is the intended protection (vault, app passcode, backup passphrase), and in those cases a deliberately slow key derivation function is used.³²

³¹E. Barker, "Recommendation for Key Management: Part 1 - General", NIST SP 800-57 Part 1 Rev. 5, May 2020.

³²E. Barker, J. Kelsey, "Recommendation for Random Number Generation Using Deterministic Random Bit Generators", NIST SP 800-90A Rev. 1, June 2015.

11.2 Hardware-Backed Storage

On iOS, the libsignal store and device secrets are kept in the Keychain with accessibility classes that restrict them to this device, so they are excluded from device backups and transfers, and every value is additionally encrypted with AES-256-GCM under a key derived from a **Secure Enclave** P-256 key (ECDH, then HKDF-SHA256 to a 256-bit key).³³ On Android, the same values are encrypted with AES-256-GCM in the module's own store under a key held in **StrongBox** where the phone has one, and in the TEE-backed Android Keystore otherwise; Android application backup is disabled for the app.³⁴ The libsignal identity key stays in libsignal's own format, because the Secure Enclave cannot hold a Curve25519 key, but it is stored only under the hardware wrapping key; the device certificate key and the account list key receive the same protection. Items copied from one phone to another do not decrypt.

Each phone records its protection level (Secure Enclave on iOS; StrongBox or TEE on Android; software only on simulators and emulators), the server holds it per device and the administrator console shows it, and policy can require a minimum level. On Android the level is backed by Key Attestation, verified offline by the installation (Chapter 14). Deleting the hardware key makes every wrapped item unreadable at once, which is the fast path of a wipe.

Attachment files and decrypted viewer copies on iOS carry the Complete data-protection class, whose file keys the operating system discards when the device locks. On desktop the same protocol runs with its secrets in the operating system's protected credential store, and an app passcode adds a second, passcode-derived wrap held only in memory; deployment guidance for desktop key storage on each operating system is given in the controlled edition.

³³ Apple Inc., "Apple Platform Security Guide" (Secure Enclave, Keychain data protection), support.apple.com/guide/security.

³⁴ Android Open Source Project, "Android Keystore system" and "Hardware-backed Keystore", developer.android.com and source.android.com.

11.3 Key Inventory

Table 19: Key inventory: origin, storage, purpose, rotation and exposure consequence

Key type	Generated where	Stored where	Purpose	Rotation	Exposure consequence
libsinal identity key pair	End-user device (libsinal)	Phone secure store (this device only), encrypted under the hardware wrapping key below; desktop: the operating system's protected credential store. Optionally in the encrypted vault backup	Device authentication in PQXDH; signs prekeys and device-link approvals	Account-wide rotation with proof of the previous key and TOTP step-up; peers are notified and re-establish sessions	Impersonation of that device to new sessions; does not decrypt past messages (forward secrecy)
Account list key pair (Curve25519, libsinal)	Primary device of the account (libsinal)	Primary device secure store under the hardware wrapping key; sealed in the encrypted vault backup	Signs the account's device list; every new device and desktop link adds a signed entry	Long-term; replaced only by vault restore with the restore proof or by an audited recovery procedure that warns every peer	Ability to add a device to the account's signed list (still subject to device approval, caps and the user-visible inventory); no message decryption
Hardware wrapping key	Phone secure hardware: Secure Enclave P-256 key (iOS); StrongBox key, or TEE key where StrongBox is absent (Android)	Inside the secure hardware, never exportable	Wraps the libsinal store, device secrets, the device certificate key and the account list key with AES-256-GCM (iOS: ECDH plus HKDF-SHA256 to a 256-bit key)	Per app installation; deleting it is the fast wipe	Requires running code on the unlocked device; the wrapped stores cannot be decrypted on another device
Device client certificate key (ECDSA P-256)	Device; certificate signing request built on the device	Phone: hardware-backed store; desktop: Electron main process (the renderer never sees it)	Mutual TLS to the edge on REST, WebSocket and media signalling	Certificate short-lived and re-minted before expiry; deleted on revoke, lost mode and wipe	Presenting that device's certificate; still needs the device's tokens, which are bound to the same device record
Signed prekey	Device	Device secure store; public half on server	Medium-term DH key for session start	Regularly	Affects only sessions started with it that also lack a one-time prekey; past messages protected by ratchet
Kyber prekeys (Kyber-1024): one-time keys and a last-resort key	Device	Device secure store; public halves on server, one-time keys until handed out	Post-quantum contribution to session start	One-time keys: consumed once and replenished automatically; last-resort key: rotated regularly	One-time key: a single session start. Last-resort key: post-quantum protection of handshakes made with it while no one-time key remained. Classical protection remains in both cases
One-time prekeys	Device	Device secure store; public halves on server until handed out	Single-use DH contribution to session start	Consumed once; replenished automatically	Minimal: a single session start
Root, chain and message keys	Device (Double Ratchet)	libsinal session store on device; message keys deleted after use	Per-message encryption and authentication	Every message (symmetric ratchet); every round trip (DH ratchet)	Exposes messages until the next DH ratchet step not observed by the attacker
Per-file key (256-bit)	Sending device (CSPRNG)	Inside the libsinal-encrypted message (Chapter 18)	AES-256-GCM encryption of one attachment	Never reused: one per file	One file

Key type	Generated where	Stored where	Purpose	Rotation	Exposure consequence
Call sender key (256-bit)	Each participant device	Memory only; delivered to peers as libsignal messages; never stored by server	AES-256-GCM frame encryption of that sender's media	HKDF ratchet every 200 ms; new random key on every join and leave	That sender's media for the current key epoch
Local cache DEK (256-bit)	Device (CSPRNG)	Wrapped by X25519 ECDH + HKDF + AES-256-GCM envelope; plaintext only in memory while signed in	SQLCipher database key and AES-256-GCM file cache key	Per user per device; destroyed on wipe	Local history on that device
Device X25519 key (cache envelope)	Device	Platform secure storage, this device only (under the hardware wrapping key on phones)	Unwraps the cache DEK	Deleted on sign-out and wipe	Local cache on that device
App passcode verifier and duress keys	Device	Secure storage; PBKDF2-HMAC-SHA256 600,000 iterations	Local unlock; duress detection	User-changed	Local unlock bypass (still bounded by OS protections)
Vault keys (optional)	Device, from vault password and recovery code	Encrypted blob on server; key derived by PBKDF2-HMAC-SHA256 600,000 iterations, AES-256-GCM, per-vault DEK wrapped twice	Encrypted backup of the identity key, the account list key and account state for recovery	On password change; backups refreshed	Offline guessing of the vault password exposes the identity key
Push payload key (per device)	Installation	Server database and device	AES-256-GCM sealing of Android wake-up envelopes	Per device registration	Wake-up metadata for that device; no message content
Edge TLS private key	Customer CA process or ACME	Edge host filesystem	TLS server authentication	Per certificate lifetime (customer policy)	Impersonation of the edge until revoked; recorded sessions stay protected (ECDHE); E2EE content unaffected
Customer CA keys	Customer PKI	Customer HSM or offline storage (recommended)	Issue edge and syslog certificates and anchor the device certificate CA	Customer policy	Ability to mint trusted certificates for the installation
Device certificate CA key	OpenBao PKI mount, under the customer CA	OpenBao, sealed at rest	Issues device client certificates with opaque names and short lifetimes; publishes the revocation list the edge loads	Customer policy	Minting device certificates; a certificate alone does not authenticate a request without the device's tokens
Administrator security keys (FIDO2)	The administrator's authenticator	Private key on the authenticator only; the installation stores the public key, algorithm and sign counter	Phishing-resistant second factor for the administrator console (ES256 or EdDSA)	Added and removed by the administrator; an owner can revoke another administrator's keys	Holder of the authenticator plus the password can sign in as that administrator
Identity-provider secrets	Customer identity provider and directory	OpenBao	OpenID Connect client secret; LDAP service account password	Customer policy	Exchanging authorisation codes or querying the directory as the installation; no account can be created and the local second factor is still required
Token signing secrets	Installation secret generation	Installation secrets, outside the database	HMAC-SHA256 signing of access and refresh tokens (separate secrets)	Operator rotation (invalidates sessions)	Forgery of session tokens; content still E2EE

Key type	Generated where	Stored where	Purpose	Rotation	Exposure consequence
TOTP secrets	Server at enrolment	Database, encrypted with AES-256-GCM under a dedicated key held outside the database and bound to the account	Second-factor verification	Re-enrolment	Second factor bypass only if the encryption key is also obtained
Field encryption key	Installation secret generation	Installation secrets, outside the database	AES-256-GCM encryption of SMTP / SMS credentials and webhook secrets	Operator rotation	Exposure of those integration credentials
OpenBao Transit keys	OpenBao	OpenBao, non-exportable, sealed at rest	Ed25519 tenant status signing; AES-256-GCM backup DEK wrapping; Ed25519 backup manifest signing	Transit key versioning	Forged status reports or backup manifests while unsealed access persists
OpenBao unseal shares	Initialisation ceremony	Held by separate custodians (Shamir threshold scheme in production procedure)	Unseal the secrets service	Re-key ceremony	With a quorum: access to all Transit keys
Backup passphrase	Operator	Operator custody, separate from media	Encrypts operational backup archives	Operator policy	Backup contents, including installation secrets
Licence and release verification keys	Tosh Defence (OpenBao Transit, non-exportable Ed25519)	Public keys in the installation trust bundle and environment	Verify licences and release manifests (a separate vendor key signs the registry timestamp statements checked by connected upgrades)	Trust bundle supports active, previous and revoked keys	Not applicable to the customer (public); vendor key compromise would allow forged licences or releases (Chapter 30)

11.4 Rotation, Expiry and Revocation

- **Prekeys** rotate on a regular cycle; a scheduler on each client replenishes one-time prekeys before they run out. When a device's registration changes, the server discards that device's stale one-time prekeys.
- **Session keys** rotate continuously through the Double Ratchet; no administrator action is needed.
- **Identity keys** rotate account-wide through an operation that requires proof of the previous identity and a TOTP step-up. Other devices of the account clear their protocol state and sign out, and peers re-establish sessions.
- **Devices** are revoked by the user or an administrator. Revocation stops token refresh and request authorisation on the next check, removes the device from future fan-outs, and, when requested, triggers a remote wipe that the device executes only after confirming the revocation with the server.
- **Device client certificates** are short-lived and re-minted by the device before they expire; revocation, lost mode and wipe delete the certificate and its key on the device, and the edge refuses a revoked certificate through the OpenBao revocation list.
- **Account list keys** are long-lived. A device leaves the list through a signed removal entry; the key itself is replaced only by a vault restore that passes the restore proof or by an audited recovery procedure, which shows every peer an identity-change style warning.
- **Administrator security keys** are added and removed by their administrator after a fresh step-up; an owner can revoke all of another administrator's keys with the owner's own fresh security-key assertion; a sign counter that goes backwards is treated as a suspected clone and refused.
- **Refresh tokens** are single-use. A replay of an already-rotated token revokes the device and raises a critical security event.
- **Server secrets** are rotated by the operator. Token-secret rotation invalidates existing sessions, which then require sign-in.
- **Licence and release keys** rotate through the trust bundle, which lists active, previous and revoked keys and is itself signed by a root key.

11.5 Compromised Key Response

Table 20: Response to suspected key compromise

Suspected compromise	Immediate action	Follow-up
A user device	Mark lost or revoke; remote wipe if appropriate	Rotate the identity key from a trusted device; review device inventory and audit events
A user password	Lock or suspend the account; reset password	Review sign-in events; re-enrol TOTP if the second factor may also be exposed
An administrator security key	The administrator removes it from their profile, or an owner revokes that administrator's keys; suspend the administrator if the password may also be exposed	Review the security-key audit events (registration, failed assertions, suspected clone)
A primary device holding the account list key	Mark lost or revoke the device	Restore the list key from the vault with the restore proof, or use the audited recovery procedure (peers are warned)
Device certificate CA	Rotate the OpenBao PKI issuer under the customer CA and update the CA and revocation list the edge loads	A certificate is bound to its device record and tokens are still required, so a forged certificate alone authenticates nothing

Suspected compromise	Immediate action	Follow-up
Edge TLS key	Replace certificate and key; revoke at the CA	Review edge logs; content remained end-to-end encrypted
Token signing secrets	Rotate secrets; all sessions end	Review audit for unusual sessions in the exposure window
Database or backup copy	Rotate field and TOTP keys if they may also be exposed; force password reset if hashes are a concern	Assess metadata exposure (Chapter 28)
OpenBao quorum or Transit key	Re-key OpenBao; rotate Transit keys	Re-issue anything signed in the exposure window

11.6 Disaster Recovery of Keys

End-user message keys are deliberately not recoverable by the organisation: if a user loses every device and has no vault backup, past message history on those devices is gone and new sessions are created on the replacement device. This is a direct consequence of keeping keys off the server. Organisations that need recovery of the user's identity can enable the encrypted vault, accepting the trade-off described in Chapter 2. Installation keys are recovered from the operational backup (which includes OpenBao data and configuration) together with the unseal quorum and the backup passphrase, which must be held by separate custodians.

The account list key, which signs each account's device list, follows the same principle. It is sealed into the user's encrypted vault backup on the device, and a restored copy is released on a new device only after the server has verified the device-restore proof (Chapter 13); a vault blob alone never yields a usable list key. An account without a vault backup is recovered only through an audited procedure in which every peer sees an identity-change style warning, so recovery cannot happen silently.

In the high-availability profile OpenBao runs as a three-node raft cluster with TLS between the nodes, and each node is unsealed by the key-share ceremony after a restart; a node started without the unseal quorum serves nothing. Encrypted raft snapshots are shipped to the second site, where a restore is verified before use. The reference topology is designed for a recovery point objective of 5 minutes and a recovery time objective of 1 hour with an asynchronous second site; these are design targets, and each deployment's disaster-recovery drill records its own measured figures (Chapter 32).

11.7 Hardware Security Modules

SANKET does not require an HSM. The released configuration unseals OpenBao with a Shamir key-share ceremony, on every node of the cluster in the high-availability profile. Customers whose policy requires hardware protection are recommended to hold their root CA in an HSM and to evaluate OpenBao's auto-unseal options against their HSM or key-management infrastructure (Chapter 33). Neither is part of the released configuration.

12 Forward Secrecy and Post-Compromise Security

Forward secrecy and post-compromise security are different properties that are often conflated. One protects the past against a future compromise; the other restores protection for the future after a compromise has ended.

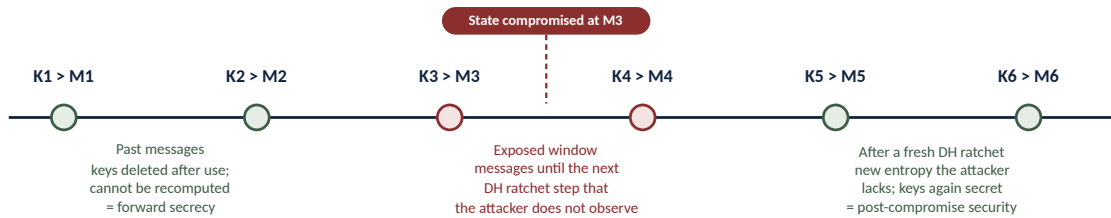
12.1 Definitions

- **Forward secrecy (FS):** If an attacker obtains a party's current key state at time T, messages sent before T remain confidential. The Double Ratchet achieves this because chain keys are advanced through a one-way function and message keys are deleted after use, so earlier keys cannot be recomputed.
- **Post-compromise security (PCS), or break-in recovery:** If an attacker obtains key state at T but then loses access, messages sent after the next Diffie-Hellman ratchet step that the attacker did not observe are confidential again. The Double Ratchet achieves this by mixing fresh ephemeral X25519 agreements into the root key.³⁵

Both properties have been established for the Signal Protocol in peer-reviewed formal analysis.³⁶ SANKET inherits them by using libsignal unmodified; it does not claim to have independently re-proved them.

Forward Secrecy and Post-Compromise Security

Illustrative message timeline under the Double Ratchet



Caveats

An attacker who keeps control of the device keeps reading; healing requires the attacker to lose access to the endpoint state.
Plaintext already stored on a device (message history, decrypted files) is not protected by forward secrecy. Local storage controls apply instead.
Server-held backups of identity keys (the optional vault) are protected by the vault password, not by the ratchet.

Figure 5: Forward secrecy and post-compromise security on a message timeline

12.2 Worked Timeline

Table 21: Effect of a key compromise at message M3

Message	Key	Status after compromise of session state at M3	Why
M1	K1	Protected	K1 was deleted after use and cannot be derived from later chain keys

³⁵T. Perrin, M. Marlinspike, "The Double Ratchet Algorithm", Signal specification. signal.org/docs/specifications/doubleratchet/

³⁶K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, D. Stebila, "A Formal Security Analysis of the Signal Messaging Protocol", IEEE EuroS&P 2017; extended version in Journal of Cryptology 33, 2020.

Message	Key	Status after compromise of session state at M3	Why
M2	K2	Protected	Same as M1
M3	K3	Exposed	The attacker holds the state that derives K3
M4	K4	Exposed if sent before the next DH ratchet	Symmetric ratchet steps are computable from the compromised chain key
M5	K5	Protected after a DH ratchet the attacker did not observe	A new ephemeral X25519 agreement contributes entropy the attacker lacks
M6	K6	Protected	Recovery persists while the attacker remains out of the endpoint

12.3Behaviour in SANKET Specifically

- **Per device pair.** Because group messages are pairwise fan-out, every recipient device has its own ratchet. A compromise of one session does not expose other members' sessions.
- **Reactions and encrypted broadcasts.** Both travel as libsignal messages with the same per-device Double Ratchet properties. An encrypted broadcast is encrypted by the broadcast-sender desktop for every recipient device through its own session, with no sender keys, so it has the same forward secrecy and post-compromise security as a one-to-one message; its server-side copy for a device is deleted as soon as that device confirms it has stored the broadcast, and otherwise at expiry.
- **Server-side ciphertext lifetime.** Delivered ciphertext is removed from the server after a short window, so a later server compromise cannot collect long history to combine with a later key compromise.
- **Session repair.** When a session is reset after repeated decryption failure, a fresh PQXDH handshake starts a new ratchet; this also refreshes the post-quantum contribution.
- **Calls.** Every call, one-to-one included, has its own server-issued call id that names its media room and binds its frame keys, so nothing from an earlier call can enter a later one. Call sender keys ratchet forward every 200 ms with HKDF and are replaced with fresh random keys whenever a participant joins or leaves, giving forward secrecy at the granularity of a key epoch and preventing joiners from decrypting earlier media or leavers from decrypting later media.
- **Post-quantum dimension.** PQXDH protects the initial handshake against harvest-now-decrypt-later, and the SPQR ratchet in the libsignal release in use extends forward secrecy and post-compromise security with a post-quantum component (Chapter 39).

12.4Scope of the Properties

NOTE - WHAT FORWARD SECRECY DOES NOT DO

- It does not protect plaintext already stored on a device. Message history in the local encrypted database is protected by local storage controls and the device's lock, not by the ratchet.
- It does not heal an attacker who keeps control of the endpoint. Post-compromise security requires the attacker to lose access to the key state.
- It does not protect a user's identity key held in the optional vault backup. That copy is protected by the vault password.
- It does not apply to routing metadata or to notice broadcasts (the announcement type that is labelled as not end-to-end encrypted), which are handled by the server as described in Chapter 20. Replies, reactions, classification labels, encrypted broadcasts, link previews, locations, contact cards and forwarded-message references travel inside the ciphertext and are covered.

13 Identity and Authentication

Every SANKET user is a customer-controlled identity, identified by a Sanket ID, and authenticated with a memorised secret and a second factor before any device-bound session is issued. Administrators are authenticated with a password and a FIDO2 security key.

13.1 Provisioning

Table 22: Ways an identity is created

Method	How it works	Status
Administrator provisioning	An administrator with the user-creation permission creates the account, subject to the password policy and the licensed seat limit	IMPLEMENTED
Single-use invitation	The organisation issues an invitation code; codes are stored only as SHA-256 hashes and administrator-issued invitations are single-use. Default registration mode	IMPLEMENTED
Directory provisioning (SCIM 2.0)	The customer's identity system creates, updates, suspends and reactivates users through the SCIM 2.0 Users endpoint. ³⁷³⁸ A SCIM-created user activates with a one-time code shown once to the administrator and stored only as an HMAC. SCIM delete suspends rather than destroys	IMPLEMENTED
SAML, OIDC, LDAP / Active Directory sign-in	The customer's own identity provider can authenticate administrators to the console (OpenID Connect or SAML 2.0, as a first factor only) and can add a directory password check for end users at enrolment, device restore and new-device sign-in (LDAP). Federated sign-in never creates an account: a provider subject must already be mapped to an existing account (see Federated Sign-In below)	CONFIGURABLE
Bulk import	Large populations are provisioned through SCIM or the Integration API	IMPLEMENTED

The registration mode is a tenant policy. A Sanket ID is permanent: when an account is deleted under the optional self-deletion policy, it is marked deleted, its devices are revoked and its Sanket ID stays reserved so that it can never be reassigned to someone else.

³⁷P. Hunt et al., "System for Cross-domain Identity Management: Core Schema", IETF RFC 7643, September 2015.

³⁸P. Hunt et al., "System for Cross-domain Identity Management: Protocol", IETF RFC 7644, September 2015.

13.2 Sign-In

Authentication Flow

Password, second factor, device-bound tokens and continuous re-checks

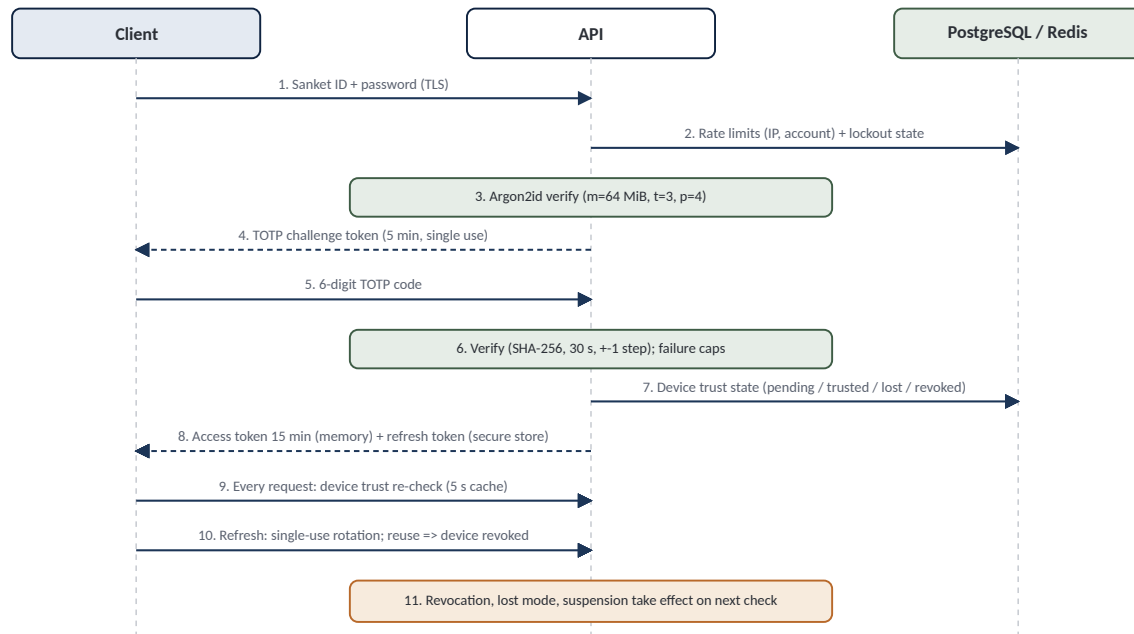


Figure 6: Authentication flow

13.2.1 First factor: password

Users sign in with Sanket ID and password. Passwords are verified against Argon2id hashes with 64 MiB of memory, three iterations and parallelism four, parameters within the range recommended by RFC 9106 and OWASP.³⁹⁴⁰ The password policy enforces a configurable minimum length and character classes, and a new password may not repeat any of the account's recent passwords. Periodic password changes are configurable, in line with current NIST guidance, which discourages them in favour of strong passwords and a second factor;⁴¹ an organisation that requires them sets a maximum password age, and a sign-in with an older password then changes it before any session is issued, after every factor has been verified.

13.2.2 Second factor for end users: TOTP

The end-user second factor is a time-based one-time password (TOTP) following RFC 6238⁴² with HMAC-SHA256, six digits and a 30-second step, accepting one step of clock drift either side. Each code is accepted once: after a code is used to sign in, step up, confirm enrolment or change a protected setting, the same code is refused for the rest of its validity, so a code observed over a shoulder or captured in transit cannot be replayed. SANKET never uses SHA-1 for TOTP, which means some popular authenticator apps that silently ignore the algorithm parameter cannot be used; SANKET displays the algorithm during enrolment and recommends compatible applications. TOTP is required for end users by default. After the password step the server issues a short-lived, single-use TOTP challenge token, with a limited number of failures allowed per token and per account.

³⁹A. Biryukov, D. Dinu, D. Khovratovich, S. Josefsson, "Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications", IETF RFC 9106, September 2021.

⁴⁰OWASP Foundation, "Password Storage Cheat Sheet", cheatsheetseries.owasp.org.

⁴¹P. Grassi et al., "Digital Identity Guidelines: Authentication and Lifecycle Management", NIST SP 800-63B.

⁴²D. M'Raihi et al., "TOTP: Time-Based One-Time Password Algorithm", IETF RFC 6238, May 2011.

13.2.3 Second factor for administrators: FIDO2 security keys

KEY POINT - PHISHING-RESISTANT ADMINISTRATOR SIGN-IN

Administrators sign in with FIDO2 security keys (WebAuthn), verified by the installation itself with no external service.⁴³⁴⁴ A security key is phishing-resistant: it signs only for the customer's own administrator origin.

After a valid password, an administrator with a registered key signs a fresh, single-use challenge with it. The installation verifies the assertion itself: the challenge, the exact administrator origin, the relying-party identifier (the customer's administrator origin), the user-presence and user-verification flags, the signature and the sign counter. Only ES256 and EdDSA credentials are accepted, attestation must be `none` or `packed`, and user verification (a PIN or biometric on the key) is required. Synced passkeys, recognisable by their backup-eligible flag, are refused by default, because they copy the private key into a vendor cloud. A sign counter that goes backwards is treated as a suspected clone, refused and audited. The verifier was reviewed against the W3C WebAuthn Level 2 specification; SANKET's policy is to use independently audited libraries, and any exception is reviewed and recorded, with detail in the controlled edition.

Use of security keys is set by tenant policy. Recovery from lost administrator keys follows audited procedures that alert the other owners; they are described in the controlled edition.

Registration, removal, revocation by an owner, failed assertions (with the reason), suspected clones, recovery actions and changes of the policy are audit events, exported through syslog and webhooks like every other event; they carry the administrator, the key label and the authenticator model, never the public key or the challenge. Phone passkeys are not offered, because the iOS and Android association checks they depend on are served by Apple and Google and cannot work in an air-gapped installation.

13.2.4 Phone and email verification

SMS and email one-time codes are used only to verify a phone number or email address during onboarding, never as a sign-in factor. SMS is off by default; supported providers include Indian aggregators as well as an international verification service, chosen by the customer. A deployment with no SMS provider simply does not collect phone numbers.

13.3 Federated Sign-In

An installation can use the customer's own identity provider. The provider's host must belong to the customer: public identity providers are refused when the configuration is saved. Provider secrets are held in OpenBao, never in the tenant configuration, and federated sign-in works only when it is enabled, licensed and configured.

Table 23: Federated sign-in

Protocol	Who and when	Checks	Status
OpenID Connect	Administrators, at console sign-in, as the first factor	Authorisation code flow with PKCE (S256); single-use state and nonce; ID token signature verified against the provider's published keys with an algorithm allowlist that excludes none and HMAC; issuer, audience, authorised party, expiry, issue time and nonce checked	CONFIGURABLE

⁴³W3C, "Web Authentication: An API for accessing Public Key Credentials, Level 2", W3C Recommendation, April 2021.

⁴⁴FIDO Alliance, "Client to Authenticator Protocol (CTAP) 2.1", fidoalliance.org.

Protocol	Who and when	Checks	Status
SAML 2.0	Administrators, at console sign-in, as the first factor	Service-provider-initiated only; signature checked against the pinned, configured provider certificate (never one carried in the message) and only the verified element read; InResponseTo single-use; Destination, Recipient, Audience and validity window checked; assertion replay cache; DTD and external entities disabled; RSA-SHA256 or stronger	CONFIGURABLE
LDAP / Active Directory	End users, at enrolment, device restore and new-device sign-in only	A bind as the mapped directory entry over LDAPS or StartTLS with the configured CA, under the installation's AES-256-GCM-only outbound TLS policy; a plain connection is refused; the client receives the same single failure answer as for a wrong password	CONFIGURABLE

The identity provider is only ever a first factor. An administrator who signs in through OpenID Connect or SAML must still complete the local second factor, and the provider's own claims about how the user authenticated are not trusted in place of it. For end users the directory password is an additional check beside TOTP, device approval and the restore proof, never a replacement.

- **Identity mapping.** A provider subject is mapped to an administrator's internal identifier or to an end user's Sanket ID by an owner in the console or through SCIM. There is no just-in-time account creation: an unmapped subject is refused, and a provider username that equals a local name is never linked automatically.
- **Roles.** Administrator roles come from the console. If the owner enables a group-to-role map, it can never grant the owner role.
- **Outage.** A provider outage cannot lock the customer out of the console. For end users an outage affects only enrolment, restore and new-device sign-in.
- **Libraries.** Federated sign-in follows SANKET's audited-library policy; any exception is reviewed and recorded, with detail in the controlled edition.

13.4 Sessions and Tokens

Table 24: Session token design

Element	Design
Format	JSON Web Tokens signed with HMAC-SHA256 (HS256) using separate access and refresh secrets of at least 32 characters; verification accepts only HS256, preventing algorithm-confusion attacks ^{45,46}
Binding	Every token carries the device identifier; authorisation re-checks that device's trust state on every request. With device mutual TLS, the certificate fingerprint verified by the edge must also match the device record (Chapter 16)
Lifetimes	Short-lived access tokens for end users and administrators; refresh tokens with an operator-configurable maximum life
Refresh rotation	Single-use refresh tokens: the first use marks the token spent atomically. A short grace window tolerates lost responses on unreliable networks. Replay after the session has moved on revokes the device and raises a critical event ⁴⁷
Re-checks at refresh	Account status, device trust, lost state and administrative blocks

⁴⁵M. Jones, J. Bradley, N. Sakimura, "JSON Web Token (JWT)", IETF RFC 7519, May 2015.

⁴⁶Y. Sheffer, D. Hardt, M. Jones, "JSON Web Token Best Current Practices", IETF RFC 8725, February 2020.

⁴⁷T. Lodderstedt et al., "Best Current Practice for OAuth 2.0 Security" (refresh token rotation), IETF RFC 9700, January 2025.

Element	Design
Storage	Mobile: access token in memory, refresh token in platform secure storage. Administrator browser: access token in memory, refresh token in an HttpOnly, Secure, SameSite=Strict cookie scoped to the authentication path. Desktop: session held in main-process secure storage
Sign-out	Revokes the refresh token for its remaining life and revokes the device
WebSocket	Opened with a single-use ticket consumed atomically, then re-checked against device trust, revocation and suspension
Idle devices	Devices idle beyond a configurable period are revoked by the scheduler
Administrator session limits	Administrator sessions end after the configured idle period and absolute duration, enforced by the server on every request and refresh. The session start and the way the second factor was met are carried in the signed tokens; only sign-in and requests made while the administrator is using the console count as activity, so background refreshes never keep an unattended console signed in. The console shows when the session ends and signs out at the limit. End-user sessions are device-bound and not subject to these limits

Effective session limits are the token lifetimes above and the automatic revocation of idle devices. On mobile, the application locks after a configurable time in the background when an app passcode or biometric lock is in use.

13.5 Brute-Force and Abuse Resistance

- **Account lockout:** after a configurable number of failures the account locks for a configurable period; administrators are alerted by email and SMS where configured. The thresholds are not disclosed to clients.
- **Compound rate limits:** sign-in attempts are limited per IP address and per account, with separate limits for activation, one-time codes and security-key assertions.
- **Global limits:** a shared request limit per verified user and device, or per IP address before authentication, and edge limits on authentication paths and on the API.
- **Adaptive limits:** limits tighten automatically when error rates rise.
- **Block manager:** administrators can block users, devices, IP addresses or network ranges, as hard denials or soft throttles.

13.6 Local Unlock

On mobile, an optional (or administrator-required) app passcode is verified locally against a PBKDF2-HMAC-SHA256 derivation with 600,000 iterations. Repeated failures lock the app for a time, and the organisation can set a number of failures after which the app erases its data; the desktop app applies the same protection. Biometric unlock uses the platform biometric API and can be disabled by policy or MDM; the key never leaves the device.

13.7 Device Restore and Recovery

A user who replaces a device can restore their account from the optional encrypted vault, opened on the new device with the vault password or recovery code, or from an exported backup file. Either one reproduces the user's identity key on the new device; a vault restore also returns the account list key (see Signed Device Lists).

The server then requires proof that the device holds the identity private key, not only the public key. It issues a fresh, short-lived challenge bound to the Sanket ID and to the device, usable once, and the device signs it with the identity private key. The server verifies the signature with libsodium before it reads or changes any account state. A request that carries only the identity public key, a signature by any other key, or a challenge that was already used or issued for another account or device is refused. An unknown account and a bad proof receive the same

generic error, so the restore path does not reveal which Sanket IDs exist. Where the installation uses a directory, the user's directory password is also checked. Restore attempts are rate-limited and recorded in the audit log, so administrators can review every recovery.

The account's TOTP enrolment is kept only when the server itself confirms that the vault was opened: a successful vault opening returns a single-use vault-restore ticket bound to the Sanket ID and the device, and the restore request must present it. Any other restore, including one from an exported backup file or a vault restore without a valid ticket, resets TOTP and the user sets it up again, so a person holding only an exported file cannot keep an unverified second factor.

13.8 Signed Device Lists

Senders need to know which devices belong to an account. Each account therefore has an **account list key**: a separate libsignal key pair, generated on the account's primary device, which signs through libsignal only. The primary device signs the account's device list, a canonical structure with a version number, the hash of the previous list, a timestamp and one entry per device (its libsignal identity key and whether it is a mobile or desktop device), plus removals. Every new device and every desktop link approval adds a signed entry produced by the approving device.

- **Append-only log.** The installation keeps an append-only log of each account's signed lists and refuses a list that does not extend the previous one (a lower version, a broken previous-hash link, a bad signature or another account's list). Clients check that the lists they see only ever extend each other and never fork.
- **Sender-side verification.** Before encrypting on any fan-out path (one-to-one and group messages, identity rotation, call frame keys and the group-call key wrap, locate, encrypted broadcasts and the file keys inside messages), mobile and desktop senders verify the recipient's signed list. A device the server lists but the signed list does not get no copy, and the user is told why.
- **Lost primary device.** The list key is sealed into the user's encrypted vault backup on the device and restored only after the device-restore proof above. Without it, the account is recovered only through an audited procedure in which every peer sees an identity-change style warning, so it cannot happen silently.
- **Self-hosted.** The log runs on the installation. SANKET does not use Signal's key transparency service or any external log.

Signed lists mean that a server that adds a device to an account cannot get messages encrypted to it. Safety-number comparison is the way to verify a contact's identity out of band (Chapter 10).

13.9 Identity Boundaries

- End users sign in on their phones with password and TOTP, with device binding and, optionally, device certificates; security keys are an administrator factor.
- The customer's identity provider never replaces the local second factor and never creates accounts.
- User X.509 certificates are not used; devices hold client certificates for mutual TLS (Chapter 26), and messaging identity remains libsignal identity keys.

14 Device Trust Model

In SANKET a device is a security principal in its own right. Access requires the conjunction of four things.

$$\textit{User identity} + \textit{Authorised device} + \textit{Cryptographic device identity} + \textit{Policy} = \textit{Access}$$

Device Registration and Trust Establishment

Primary mobile enrolment, then companion desktop linking with the phone as root of trust

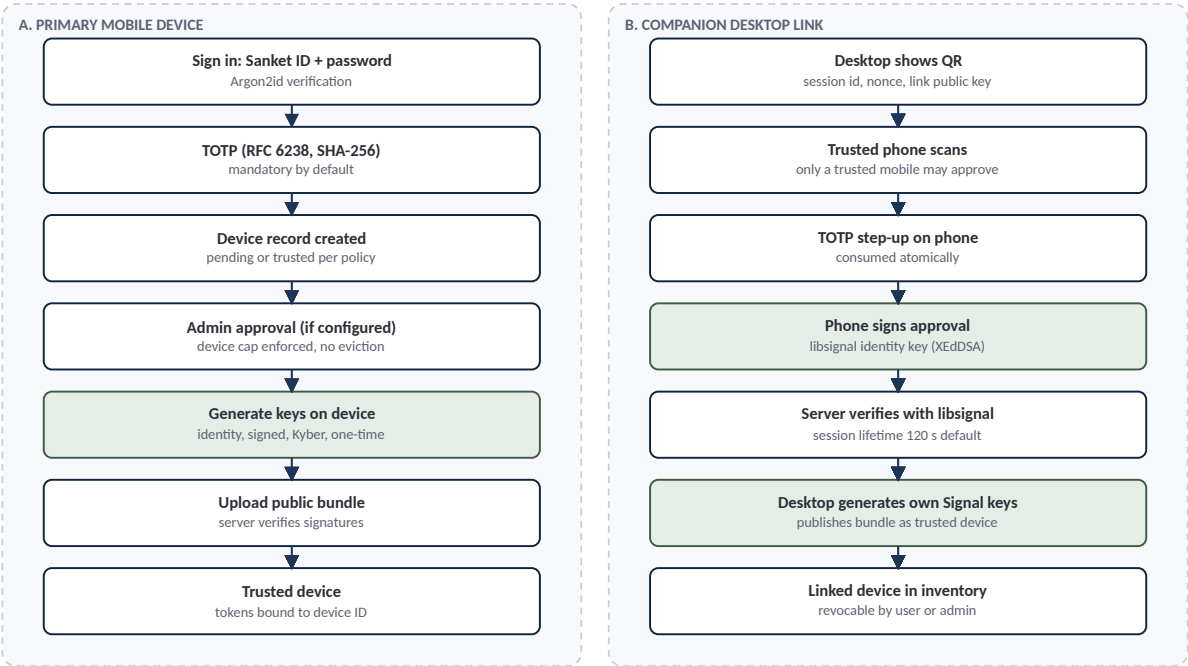


Figure 7: Device registration and trust establishment

14.1 Device States

Table 25: Device trust states

State	Meaning	What the device can do
pending	Registered but not yet approved (administrator approval mode, or user at the device cap)	Nothing; no tokens and no client certificate are issued; administrators are notified
trusted	Approved device with a published libsinal bundle and an entry in the account's signed device list	Full use; included in message fan-outs; obtains its client certificate and re-mints it before expiry
lost	Marked lost by the user from another device or by an administrator	Locked out of the service and excluded from new messages until marked found
revoked	Removed from the account	Cannot refresh or authenticate; excluded from fan-outs; its client certificate is deleted and refused by the edge; wipe executes if requested

14.1.1 Client certificate lifecycle

Every approved device holds a client certificate for mutual TLS (Chapters 10 and 16). The device generates the ECDSA P-256 key and the certificate signing request itself, on phones inside the hardware-backed store; the first certificate is minted only after the device has been approved. The device re-mints the certificate before it expires, and a revoked device cannot re-mint. Revocation, lost mode and a wipe delete the certificate and its key on the device, and the edge refuses a revoked certificate through its revocation list. Mobile and desktop follow the same lifecycle; on desktop the key is held by the Electron main process and the renderer never sees it.

14.2Registration

- **Approval mode** is a tenant policy: automatic (a new device is trusted while the user is below the trusted-device cap) or administrator approval (recommended for high-trust deployments).
- **Per-platform caps** are enforced by refusing the new device; SANKET never silently evicts an existing device to make room.
- **Ownership:** a device record cannot be taken over by another account, and a banned handset remains banned across reinstallation.
- **Keys:** a trusted device generates its libsignal identity and prekeys locally and publishes only public keys, which the server verifies with libsignal.
- **Signed entry:** a new device is added to the account's signed device list by the approving device; a device the server records without a valid signed entry receives nothing.

14.3Companion Desktop Linking

A desktop client cannot sign in with a password. It becomes a device of the account only when a trusted phone approves it. The desktop displays a QR code containing a session identifier, a nonce and its link public key; the phone scans it, the user completes a TOTP step-up (consumed atomically), and the phone signs a digest of the version, session, nonce, companion key and client metadata with its libsignal identity key (XEdDSA). The server verifies that signature with libsignal before accepting the link. The approving phone also adds a signed entry for the desktop to the account's device list; a link approval without a valid list signature leaves the desktop off the list, so other users' devices never encrypt to it. The link session is short-lived and the number of companion devices is capped. The same flow applies on mobile and desktop.

14.4Device Inventory and Remote Actions

Table 26: Device governance actions

Action	Who	Effect	Status
Approve	Administrator	Moves a pending device to trusted	IMPLEMENTED
Revoke	User (other device) or administrator	Ends authentication and future delivery; the device's client certificate is deleted and refused	IMPLEMENTED
Remote wipe	Administrator (licence and policy gated), per device or per user	Erases SANKET's data on a revoked device; completion is reported	IMPLEMENTED
Lost / found	User from another device, or administrator	Locks the device out until it is found; an automatic wipe can follow (configurable)	IMPLEMENTED

Action	Who	Effect	Status
Locate	Administrator (enabled by the organisation)	Requests a single location fix from a lost phone, sealed end to end to a one-time key held in the requesting administrator's browser (HPKE with X25519, HKDF-SHA256, AES-256-GCM) ⁴⁸ and signed with the phone's identity key	IMPLEMENTED
Duress	User	Lets a user under coercion protect the account's data and alert administrators	IMPLEMENTED

14.5 Anti-Theft and Reset

SANKET's anti-theft model combines the controls above: a lost device is locked out and stops receiving messages, can be located if the organisation has enabled Locate, and is then either marked found or remotely wiped. A wipe erases everything SANKET holds on the device, returning the app to its first-run state. A full factory reset of the handset itself is an operating-system function performed through the customer's MDM, and SANKET's managed-configuration support is designed to work alongside it.

14.6 Hardware-Backed Keys

On phones, the libsignal store, device secrets, the device certificate key and the account list key are encrypted with AES-256-GCM under a hardware-backed key: a Secure Enclave key on iOS, a StrongBox key on Android where the phone has one and a TEE key otherwise. They remain this-device-only and excluded from backups and device transfers (Chapter 11). Each phone reports its protection level, the server holds it per device, the administrator console shows it, and policy can require a minimum level. On desktop the same protocol runs with its secrets in the operating system's protected credential store (Chapter 11).

14.7 Posture and Integrity

Device posture, such as patch level and root or jailbreak state, is enforced through the customer's mobile device management, which is the authoritative source of posture on a managed fleet and avoids duplicating, and potentially contradicting, MDM compliance decisions inside the app. Alongside MDM compliance, SANKET detects common root and jailbreak indicators on the phone itself, using local checks only (no Google Play Integrity and no Apple App Attest, both foreign services). Detections are audited and alerted to administrators, and enforcement is configurable: when the organisation or its MDM requires it, a flagged device cannot use the app and the server refuses that device's requests as well, so a tampered app gains nothing. Local heuristics can be defeated on a device the attacker fully controls; they raise the bar and give administrators a signal, and MDM remains the authoritative source. SANKET supports vendor-neutral managed configuration on iOS (Managed App Configuration) and Android (managed configurations), which can only tighten policy: an MDM can require an app passcode, shorten the lock timeout, disable biometric unlock, tighten the wipe-after-failures policy and pin the expected tenant.

14.7.1 Android Key Attestation

On Android the phone proves where its keys live with Key Attestation:⁴⁹ it answers a fresh server challenge with the attestation certificate chain of a hardware key, and the installation verifies the chain offline against the vendor root certificates shipped in the release, with a certificate revocation snapshot that is also shipped with each release, so verification makes no call to Google. A self-signed chain, a revoked intermediate or a replayed challenge

⁴⁸R. Barnes, K. Bhargavan, B. Lipp, C. Wood, "Hybrid Public Key Encryption", IETF RFC 9180, February 2022.

⁴⁹Android Open Source Project, "Android Keystore system" and "Hardware-backed Keystore", developer.android.com and source.android.com. Key attestation.

is refused, and a client cannot raise its reported protection level without an attestation that verifies. Enforcement is configurable, as for root detection.

14.8 Forced and Recommended Updates

The administrator sets a minimum app version for iPhone and for Android separately (and one for the desktop client), and a signed licence may set a minimum that applies to every platform. A phone below its platform's minimum is blocked with an update screen whose button opens the tenant's own MDM or private-channel link, or the store listing; with no link configured it tells the user to contact their unit administrator. Between the minimum and a recommended version the app shows a dismissible "update recommended" banner. The phone re-checks the policy regularly, on return to the foreground and when the administrator changes the configuration. The server can also refuse outdated apps itself, per platform, with a dedicated "update required" answer that the apps show as the same update screen (configurable). Device revocation, lost-mode and wipe answers always take precedence over it.

14.9 Managed Devices and BYOD

Table 27: Managed devices compared with personally owned devices

Consideration	Managed device (recommended for high-trust use)	Personally owned device (BYOD)
Operating system patching	Enforced by MDM	Depends on the user
Root / jailbreak posture	Enforced by MDM compliance, and blocked by SANKET when the MDM requires it	Detected and reported by SANKET; blocked when the organisation's policy requires it
Key protection	Hardware level reported and, on Android, attested; policy can require a minimum level	Same reporting and attestation; the level depends on the handset
App distribution	Enterprise or managed distribution	Public stores or side-loading
Policy tightening	SANKET managed configuration applies	Tenant policy only
Device loss	MDM wipe plus SANKET remote wipe and lost mode	SANKET remote wipe and lost mode
Residual risk	Lower; still subject to endpoint compromise	Higher; mixed personal use enlarges the attack surface

14.10 Device-List Integrity

Senders learn a recipient's devices from the installation, but they no longer have to trust that list as served. Each account's device list is signed by its account list key, the installation keeps an append-only log of the signed lists, and every sender, mobile and desktop alike, verifies the signed list before it encrypts on any fan-out path (Chapter 13). A device that the server lists but the signed list does not receives no copy, and the user sees why. Identity keys are pinned, and device caps, administrator approval, user-visible device inventories, audit events and safety-number comparison continue to make any unexpected device visible to users and administrators.

When a contact's identity key changes, users are notified in every shared conversation. A device that was offline when the key changed receives the notice when it next connects: the server keeps the notice for each recipient device until that device acknowledges it, and a device can acknowledge only its own notices.

15 Role and Policy Architecture

SANKET separates five things that are often merged: who a person is, what they are allowed to do, which content they can cryptographically read, which devices are trusted, and what administrative power they hold.

Table 28: Five distinct control concepts

Concept	Question it answers	Where enforced in SANKET
Authentication	Is this really the account holder?	Password (Argon2id) and TOTP at sign-in for end users, with an optional directory check at enrolment and restore; password (or the customer identity provider as first factor) and a FIDO2 security key for administrators; device-bound tokens
Authorisation	May this account perform this action?	API route checks: group roles, administrator permissions, tenant policy, licence features
Cryptographic access	Can this device decrypt this content?	Only devices included in a libsignal fan-out, and present on the recipient account's signed device list, receive a decryptable copy
Device trust	Is this device allowed to act for the account?	Device state checked on every request and refresh; device client certificate at the edge; key protection level and attestation where policy requires them
Administrative privilege	May this administrator change identities, devices or policy?	Permission-scoped administrator roles, a FIDO2 security key and audit

The separation matters because authorisation and cryptographic access can diverge in useful ways. Removing a member from a group is an authorisation change that immediately removes their devices from future fan-outs; no administrator can, by changing a role, give a device the ability to decrypt messages that were never encrypted to it.

15.1 Administrator Roles (RBAC)

Administrator permissions are organised into categories (users, broadcast, vault, audit log, settings, alerts, licence, integrations and location), each with read, create, edit and delete actions. Five built-in roles are provided and cannot be altered; customers can define additional roles, but an administrator cannot create or assign a role with more permission than they hold. Every administrator, whatever the role, is subject to the same strong-authentication policy (Chapter 13). When administrators sign in through the customer's identity provider, roles still come from the console unless the owner enables a group-to-role map, and that map can never grant the Owner role.

Table 29: Built-in administrator roles

Permission category	Owner	User Manager	Config Manager	Threat Monitor	Auditor
Users	All	All	-	-	Read
Broadcast	All	-	All	-	Read
Vault	All	-	Read	-	Read
Audit log	All	Read	-	Read	Read
Settings	All	-	All	-	Read

Permission category	Owner	User Manager	Config Manager	Threat Monitor	Auditor
Alerts	All	-	-	Read, edit	Read
Licence	All	-	-	-	Read
Integrations	All	-	-	-	-
Location	All	-	-	-	-

Customers map these to their own structure, for example Organisation Administrator (Owner), Security Administrator (Threat Monitor plus a custom role with user-edit permission for lost-device handling), and Auditor.

15.2Group Roles

Table 30: Group roles and permissions

Action	Owner	Group administrator	Member
Send and read messages	Yes	Yes	Yes
Add or remove members	Yes	Yes	-
Remove a group administrator	Yes	-	-
Change roles, transfer ownership, delete group	Yes	-	-
Rename group	Yes	Yes	If policy allows
Approve pending members (when member approval is on)	Yes	Yes	-
Leave group	If policy allows	If policy allows	If policy allows

Member approval is on by default: an added member is pending until they accept, so nobody is placed into a conversation without consent. Group creation is itself a policy; a deployment chooses the model that suits its structure. Under the administrators-only policy, the default, administrators with the user-creation permission create groups from the console, naming an active end user as the owner and the initial members by Sanket ID; member approval applies to them as to any group, and each creation is audited. Notice broadcasts, the announcement type that is labelled as not end-to-end encrypted, can be sent only by administrators with the broadcast-creation permission, and only the administrator who created a notice can edit or send it.

15.3Broadcast-Sender Accounts

Administrators hold no libsignal identity, so they cannot send end-to-end encrypted content. Encrypted broadcasts are therefore sent by a dedicated **broadcast-sender account**: a user account with its own Sanket ID, created, suspended and assigned to named administrators in the console, with every such action audited. It is excluded from contact search, from "all users" audiences and from ordinary chats. Its root of trust is a mobile device in the organisation's custody, and its sending desktop must be approved both by that mobile and in the administrator console. The desktop app encrypts each broadcast for every recipient device through its own libsignal Double Ratchet session, with no sender keys; recipients pin the sending device and identity on the first broadcast and are warned if a later broadcast comes from another device or key. Sending is a desktop function by design; receiving is identical on mobile and desktop. Chapter 17 describes the broadcast lifecycle.

15.4End-User Policy Controls

Policies apply to the whole installation and are set by administrators with the settings permission. Clients receive a reduced copy of the policy from which sensitive values, such as lockout thresholds and rate limits, are removed. The principal enforced controls are:

Table 31: Enforced policy controls (selection)

Area	Controls	Enforced at
Access	Registration mode; TOTP requirement; administrator security keys; identity providers; lockout thresholds; device approval mode; trusted-device and per-platform caps; idle-device revocation	Server
Local security	App passcode requirement; biometric unlock; lock timeout; wipe after failed unlocks; duress PIN	Client (mobile); MDM can tighten
Messaging	Message editing, delete-for-everyone window, disappearing messages, read receipts, typing indicators	Server
Messaging	Forwarding, reactions (end-to-end encrypted; an optional emoji-free server count for very large groups), quoting, link previews, starring	Client
Content handling	Clipboard copy of message text; forwarding of files and of vault files into chats, and sharing attachments to other apps	Client (mobile and desktop)
Classification	Customer-defined ranked list of labels and a default label; content labelled above the lowest rank has forwarding, export and download turned off (Chapter 38)	Client (mobile and desktop); label inside the end-to-end encryption
Files	File sharing on or off per source (camera, gallery, documents, location, contacts, voice); maximum size; allowed and blocked types; download to device; data export	Server and client
Calls	Voice, video and group calls; maximum duration; participants; per-user call rates	Server
Groups	Creation permission; member approval; maximum members; renaming; leaving	Server
Device governance	Remote wipe per device and per user; lost mode and automatic wipe; locate; device mutual TLS; minimum key protection level; attestation enforcement	Server
Privacy	Notification sender names and previews (both off by default); last-seen visibility; profile photo and display-name policy	Server and client
Retention	Hot message window, broadcast, call log and alert retention (Chapter 44)	Server
Integrations	Integration API enablement; audit export	Server and licence

NOTE - CLIENT-SIDE CONTROLS

Client-enforced controls such as forwarding restrictions govern the behaviour of the SANKET applications. As with any messaging platform, they cannot prevent an authorised reader from copying content by other means, which is why SANKET pairs them with audit, download and export policy, and managed devices. Administrator sign-in and sessions can be limited to configured IPv4 and IPv6 address ranges, enforced by the API (Chapter 27); a firewall or reverse proxy in front of the console remains recommended.

15.5 Attribute-Based and Hierarchical Policy

SANKET's current model is role-based for administrators and membership-based for content, with installation-wide policy. It does not implement attribute-based access control, per-department policy or an organisational hierarchy with inherited permissions. Large organisations typically deploy separate installations for separate security domains, which gives strong compartmentalisation at the cost of cross-domain communication, and installation-level separation remains the recommended separation between formally accredited levels. Per-message and per-file classification labels are implemented (Chapter 38); hierarchical units, attribute-based policy, conversation-level labels and a per-label handling matrix are roadmap directions.

16 Zero Trust Architecture

Zero trust, as defined by NIST, means that no implicit trust is granted to assets or accounts based on their network location or ownership, and that authentication and authorisation are discrete functions performed before each session to a resource.⁵⁰

In practice SANKET refuses to treat any of the following as sufficient grounds for access:

- the request comes from an internal network or a trusted VLAN;
- the user authenticated earlier;
- the device was authorised last month;
- the caller is an administrator who owns the infrastructure;
- the connection arrived through the edge proxy.

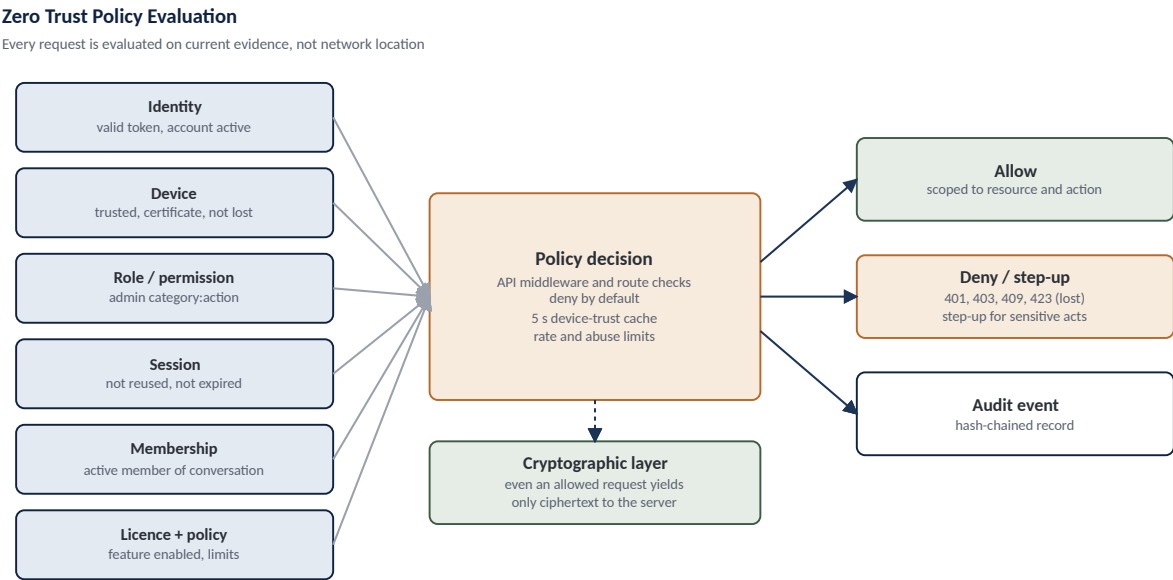


Figure 8: Zero trust policy evaluation

16.1 Evaluation Inputs

Table 32: Evidence evaluated per request

Input	Evidence	Freshness
Identity	Valid, unexpired HS256 token whose subject account is active	Short-lived access token
Device	Device in the trusted state, not lost or revoked	Re-checked every request
Device certificate	Client certificate verified by the edge against the device CA and its revocation list; the verified fingerprint must match the device record that the token names	Every connection on the device routes
Key protection	Reported hardware protection level of the device, attested on Android	Policy can require a minimum level

⁵⁰S. Rose, O. Borchert, S. Mitchell, S. Connelly, "Zero Trust Architecture", NIST SP 800-207, August 2020.

Input	Evidence	Freshness
Session	Refresh token not previously used; generation counter current	Every refresh
Role and permission	Administrator category and action; group role	Every request
Resource relationship	Active membership of the conversation for messages, files and calls	Every request; locked inside send transactions
Policy and licence	Feature enabled by tenant policy and licence; limits	Every request
Abuse state	Rate limits, blocks, lockout	Every request
Step-up	TOTP re-authentication for sensitive operations such as device linking and identity rotation; a fresh security-key assertion for administrator key management and for an owner revoking another administrator's keys	Per operation

16.2 Cryptography as the Last Control

Zero trust is usually discussed as an access-decision architecture. SANKET adds a cryptographic layer underneath it: even when every access check passes, including for an administrator or a compromised server process, what the server can hand over is ciphertext. The access decision governs metadata and service; the cryptography governs content.

The same principle extends to the list of devices a sender encrypts for. The server supplies that list, but the sender does not take it on trust: every account's device list is signed by its own account list key, and the sending device verifies the signed list before it encrypts, on mobile and desktop alike. A device that the server adds without a valid signed entry receives nothing (Chapter 13).

16.3 Device Mutual TLS

A device-bound token proves that a session was issued to a device; a client certificate proves, on every connection, that the connection comes from that device's key. Every approved device holds a short-lived certificate whose ECDSA P-256 key never leaves it (Chapter 10). The edge verifies the certificate and the revocation list and passes the verified fingerprint to the API, which binds it to the device record: a token presented with another device's certificate is refused, and where certificates are required a stolen access token is useless on a machine that does not hold the device key. Enforcement is configurable per tenant.

16.4 Integration with the Customer's Zero Trust Architecture

- Device posture (patch level, root or jailbreak state) is supplied by the customer's MDM, and SANKET also checks the phone for root and jailbreak indicators itself, reporting them and, by policy, blocking use.
- Network context (IP reputation, country) is used for rate limiting and blocking; administrator access can be limited to configured address ranges, enforced by the API, with the customer's edge as a further layer.
- Internal service traffic stays on isolated container networks inside the host in the single-host profile. In the multi-host high-availability profile, database replication and traffic between OpenBao nodes run over TLS with AES-256-GCM only, and the device CA and revocation list are loaded on every edge host (Chapter 32).

17 Message Security Lifecycle

This chapter follows a single message from composition to deletion and records, at each stage, where plaintext, ciphertext and keys exist, what metadata is produced and which threat each stage mitigates.

Message Encryption Lifecycle

Pairwise libsodium encryption to every recipient device; the server relays ciphertext

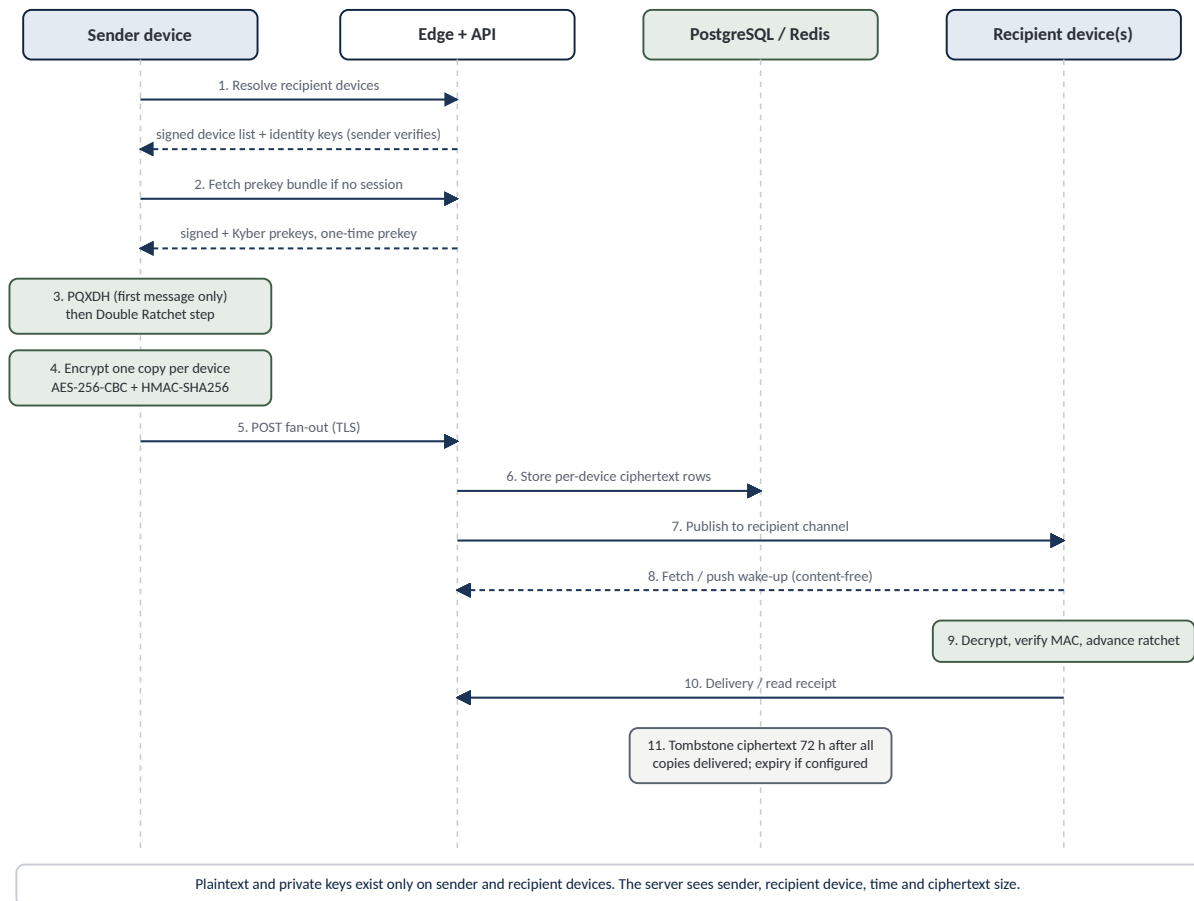


Figure 9: Message encryption lifecycle

17.1 Delivery Reliability

Each composed message carries an idempotency key that is identical on every retry, and the server answers a repeat with the message it already stored, so a retried send can never be delivered twice. While offline, mobile and desktop clients hold outbound text messages in an encrypted outbox for up to seven days and send them on reconnection; both hold attachments and voice notes the same way, as encrypted copies, sending each with the same key and naming the file it uploaded. Mobile and desktop use the same API contract for sending, retry and receipts, and decide which failures to retry by one shared rule: no response, a timeout, rate limiting and server errors (an upgrade's maintenance window included) are retried with a backoff that doubles from five seconds to at most one hour; any other rejection by the server, and a message that cannot be encrypted for its recipients or encoded, marks the message as failed in the conversation and is not retried.

Before encrypting, the sending device checks every recipient device the installation names against that account's signed device list (Chapters 13 and 14). The check runs on every fan-out path, on mobile and desktop alike: one-to-one and group messages, identity rotation, call frame keys, location requests, encrypted broadcasts and the

attachment keys carried inside messages. A device that is not on the signed list receives no copy, and the user is shown why, so a device added by the server alone never receives content.

Table 33: Message lifecycle: plaintext, ciphertext, keys, metadata and threats by stage

Stage	Plaintext location	Ciphertext location	Key location	Metadata generated	Threat mitigated
1. User authenticated	Sender device (UI)	None	Device secure store	Sign-in audit event	Credential theft (TOTP), session hijack (device binding)
2. Device authorised	-	-	Device secure store	None per request; audit on state changes	Revoked, lost or rogue devices
3. Recipient devices resolved and verified	-	-	Public keys from server; each recipient's signed device list checked on the sender	Server learns the sender is about to message the conversation	Sending to removed members (active-membership check); a device added by the server (signed device lists)
4. Session established (first message)	-	PQXDH handshake material	New session state on sender	One-time prekey consumed on server	Harvest-now-decrypt-later (Kyber); impersonation (signed prekeys)
5. Encrypted locally	Sender memory	One libsignal ciphertext per recipient device	Message keys derived and deleted	None	Server and network disclosure
6. Transmitted	-	Inside TLS	-	TLS endpoints, size, time	Network observers, tampering (TLS + MAC)
7. Received by server	-	Per-device rows	-	Sender account and device, recipient devices, conversation, time, padded size	Duplicate sends (idempotency key); flooding (rate limits)
8. Stored for delivery	-	PostgreSQL	-	Delivery status	Database theft exposes ciphertext only for bodies
9. Delivered	-	WebSocket or fetch over TLS	-	Wake-up push (content-free)	Push-channel disclosure (sealed or generic payloads)
10. Decrypted locally	Recipient memory, then SQLCipher	-	Recipient session advanced	None	Tampering (MAC); replay (duplicate detection)
11. Delivery acknowledgement	-	-	-	Delivery receipt	-
12. Read acknowledgement	-	-	-	Read receipt (if enabled by policy and user)	Unwanted presence disclosure (policy)
13. Retention applies	Recipient device	Tombstoned 72 h after full delivery	-	Stub row for ordering	Long-term server-side ciphertext accumulation
14. Expiry applies (if configured)	Removed from devices at expiry	Deleted from server at expiry	-	-	Exposure of stale content on devices
15. Server stub deleted	-	Hard-deleted after 90 days	-	-	-
16. Audit metadata retained	-	-	-	Message-sent audit event with conversation identifier, never content	Repudiation of administrative actions

17.2 Content Envelope

Inside every libsignal message is a versioned content envelope (version 3). It declares what the message is (an ordinary message, a reaction or a broadcast) and carries a message identifier generated by the sender, identical in every per-device copy, so that every device refers to the same message by the same identifier without the server assigning it. It also carries the optional classification label of the message and of each attachment (Chapter 38), so the server can neither read nor alter a label. Mobile and desktop produce byte-identical envelopes for the same content.

Envelope versions are introduced under tenant control and only once every client platform can read them; a message in an envelope version the app cannot yet read is kept on the device and shown once the app is updated.

17.3 Message Adjuncts

Replies (the quoted excerpt), link previews, shared locations, shared contact cards and forwarded-message references travel inside the end-to-end encrypted message, together with the text, in the content envelope. The envelope is padded to fixed-size blocks of 160 bytes before encryption, so the ciphertext size reveals only a size band, not the exact length of the text. The server keeps only two delivery flags next to the ciphertext: the idempotency key that makes a resend harmless and a flag that the sender dismissed a link preview. Each device keeps the decrypted fields in its encrypted local database. Reactions travel as end-to-end encrypted messages like any other; each device computes the counts (below). Each of these features is controlled by tenant policy: link previews are off by default, and location sharing, contact sharing, quoting and reactions can each be disabled where an organisation's handling rules require it.

A malformed or oversized envelope is shown as a notice that the message could not be displayed; one from a later version asks the user to update the app and is kept on the device so the updated app can show it.

17.4 Reactions

A reaction is a small libsignal message whose envelope names the target message by its identifier and carries the emoji, or its removal. The server stores it like any other message, as opaque per-device rows tombstoned on the same schedule, sends no push for it, and does not count it as unread or as a conversation's last message. Each device computes the counts itself: for each sender the latest reaction wins, and the sender is the one libsignal authenticated, never a name written inside the content. The earlier path that kept reactions in clear on the server has been removed, and the clear reaction records it had stored were deleted.

A device that joins a large group later, or whose copies of older reactions have already been tombstoned, cannot compute counts for old messages. For that case a tenant option, off by default, lets the server keep an emoji-free count per message in large groups: the server then learns who reacted to which message, never with which emoji, and clients show the full breakdown from their own copies whenever they have them (Chapter 20). Reactions can still be disabled by policy.

17.5 Encrypted Broadcasts

Priority broadcasts to large audiences are end-to-end encrypted messages too. They are sent by a dedicated broadcast-sender account (Chapter 27) from its one approved desktop, whose main process (never the user interface process) encrypts the broadcast separately for every recipient device with a per-device libsignal Double Ratchet session, without sender keys. The fan-out is a resumable job kept in the desktop's SQLCipher database, with progress, an estimated time and a cancel control; the sender's libsignal sessions live in their own SQLCipher

store. An attachment is encrypted once on the sender desktop with a per-file AES-256-GCM key and uploaded once, and the key travels inside each device's copy (Chapter 18).

The server holds the broadcast's metadata (audience, priority, acknowledgement deadline, expiry and receipts) and the per-device ciphertext, which it deletes as soon as each device confirms it has stored the broadcast, and otherwise at expiry. Priority, deadline, sent time and expiry are also carried inside the encrypted envelope, and a client refuses a copy whose envelope disagrees with the server's record, so a hostile server cannot downgrade a critical broadcast, move its deadline or replay an old one. Encrypted broadcasts are sent immediately (no scheduling and no drafts held by the server), must carry an expiry and have a server-enforced recipient limit counted in devices. A device provisioned while a broadcast is still live is queued and encrypted for by the sender desktop. Recipients pin the sending device and identity and are shown an identity-change notice if a later broadcast comes from another device or key.

Push messages, notification records, escalation reminders and administrator alerts for an encrypted broadcast carry only identifiers, priority and deadlines; the lock-screen text on iOS and Android is generic by priority (for example "Urgent broadcast"). Acknowledgement, deadline and escalation work on this metadata, and the deadline starts when the fan-out has finished. Sending is desktop-only by design; receiving, display and acknowledgement are identical on mobile and desktop. A separate notice type, which the server can read, remains for non-sensitive announcements and is always labelled as not end-to-end encrypted (Chapter 27). The sending desktop shows the recipient count before it sends, and only devices on each recipient's signed device list receive a copy.

17.6 Editing and Deleting

- **Edit** (configurable): only the sender can edit, within a window; the edited message is re-encrypted with a fresh fan-out.
- **Delete for everyone** (configurable, within a time window): the server blanks every copy. Recipients' clients remove the message; a recipient who has already read or copied it may retain that knowledge.
- **Delete for me**: removes the message from the user's own view and device.
- **Disappearing messages** (configurable): when allowed by policy, users choose durations from one hour to ninety days; administrators can force a maximum duration, from 30 seconds to 30 days, and the shorter of it and the user's own choice applies. Expired messages are deleted from the server and purged from device caches and attachments.

18 File Security Lifecycle

Files are encrypted before they leave the device, stored as opaque objects, and decrypted only on recipient devices.

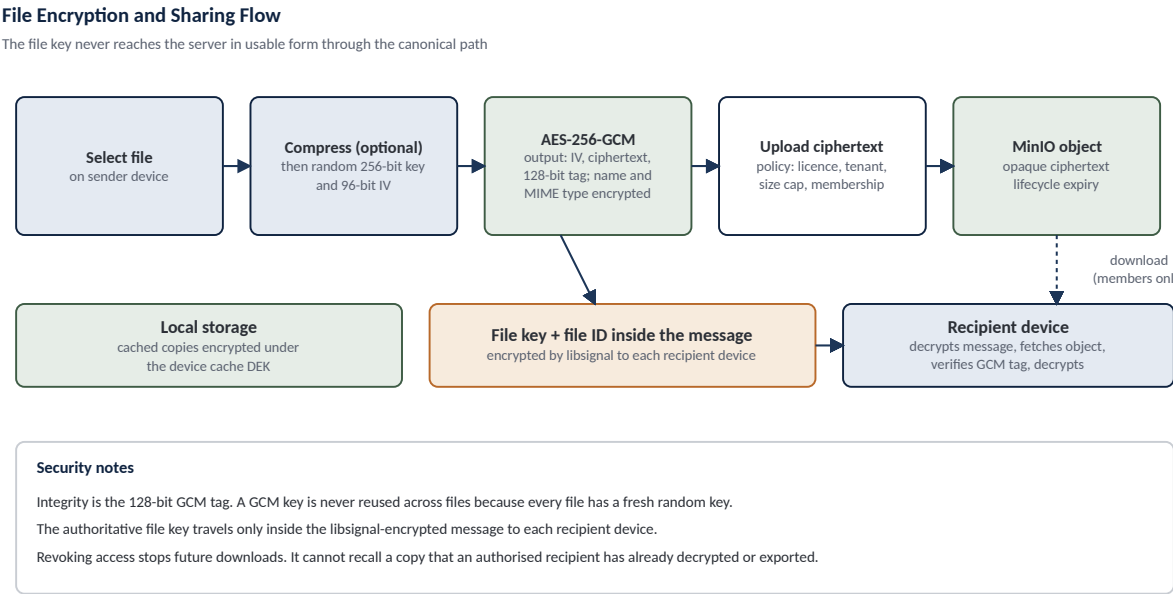


Figure 10: File encryption and sharing flow

Table 34: File lifecycle

Stage	What happens	Control
Selection	User chooses a file, photo, video, voice note, location or contact	Per-source enablement and type allow/block lists
Local encryption	Optional compression, then AES-256-GCM with a fresh random 256-bit key and 96-bit IV; name and type encrypted	Strict 256-bit key helper; per-file key
Policy check	Upload request carries metadata only; server checks licence, tenant policy, size and duration caps and active membership	Server
Upload	Ciphertext sent through the API to object storage; Content-Length checked before buffering	TLS; size limits
Key delivery	File key and identifier sent inside the message body, encrypted by libsinal to each recipient device on its signed device list	End-to-end
Classification label	Optional label per attachment, carried with the file key inside the end-to-end encrypted envelope; the upload request carries no label, so the server never learns it (Chapter 38)	End-to-end; client policy
Access	Download allowed only to active members of the conversation; always served as opaque binary	Server membership check
Preview	Rendered on the device from decrypted data; no server-side thumbnails	Client
Download to device	Saving outside the app is off by default; desktop has a separate export permission. A file whose effective label ranks above the lowest configured label cannot be forwarded, exported or saved to the device on mobile or desktop, the desktop secure viewer included, whatever the general policy says	Policy and label (mobile and desktop)

Stage	What happens	Control
Watermarking	Desktop and mobile file viewers can overlay a visible Sanket ID or name watermark (configurable)	Desktop and mobile
Local caching	Cached copies stored encrypted under the device cache DEK	SQLCipher / AES-256-GCM
Expiry	Object-storage lifecycle expires attachment objects after 90 days; disappearing-message expiry also removes cached copies on devices	Server lifecycle; client purge
Audit	Upload, download, preview and export events recorded	Audit chain
Deletion	Objects removed by lifecycle or account purge	Server

18.1 Broadcast Attachments

An attachment to an encrypted broadcast is encrypted once, on the broadcast-sender desktop, with its own random AES-256-GCM key, and uploaded once, however many devices receive it. The file key travels inside each recipient device's libsinal-encrypted copy of the broadcast, so the server stores a single opaque object and never the key (Chapter 17). An attachment to a notice-type broadcast, which the server can read, is not end-to-end encrypted and is labelled accordingly (Chapter 27).

18.2 Viewing Files on Mobile

Where the operating system viewer requires a file on disk, a short-lived decrypted copy is placed in the application's private sandbox, protected by the device's storage encryption, and is removed as soon as it is no longer in use, with a start-up sweep and every erase path removing any that remain. Inline images and media in a conversation are shown again from the encrypted attachment store when the app returns to the foreground. Customers with strict handling requirements combine the download-to-device restriction with managed devices.

Documents open inside the application, so the sharing restriction never leaves a user unable to read a file they received: text formats are shown as plain text and never rendered (received HTML and images in SVG form cannot run or load anything), PDF and office documents are shown by platform renderers with scripts and navigation disabled, under the same deletion rules.

18.3 Revocation

NOTE - REVOCATION AND PLAINTEXT ALREADY OBTAINED

Removing a member, revoking a device, or deleting an object prevents future downloads and future decryption on revoked devices. It cannot recall a file that an authorised recipient has already downloaded, decrypted, exported, printed or photographed. File access follows conversation membership, so removing a member is the revocation action.

19 Secure Voice and Video

SANKET calls run through a self-hosted selective forwarding unit (SFU), with two independent layers of protection: end-to-end frame encryption between participants, and SRTP between each participant and the SFU.

Real-Time Call Protection

Two independent layers: per-sender frame encryption end to end, and SRTP to the media server

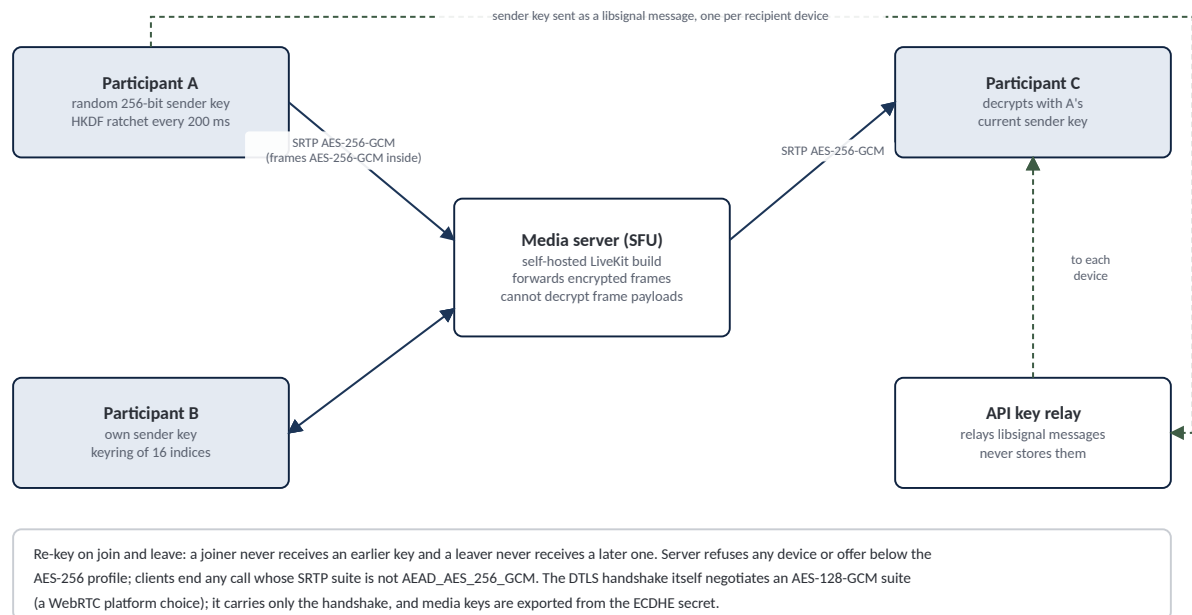


Figure 11: Real-time call protection

19.1 Media Server

The SFU is Tosh Defence's build of the open-source LiveKit server,⁵¹ pinned to a fixed upstream release at a fixed source commit (the build fails if the upstream tag moves) and distributed to customers as a release image. Every call, one-to-one included, has its own server-issued call identifier that names its media room and binds its frame keys, so nothing from an earlier call can enter a later one. Room tokens are issued only to active members of the conversation, and the server signs each participant's device identity into the token so a client cannot claim another device. Media uses a single multiplexed UDP port, the primary and lowest-latency path, with a TCP fallback; where a network allows only outbound TCP 443, media is relayed by TURN over TLS on port 443 (Calls in Isolated and Restrictive Networks, below). In the high-availability profile the media server runs as several nodes coordinated through the installation's state store, each with its own media address and the same AES-256 SRTP policy (Chapter 32).

19.2 Frame Encryption (End to End)

Table 35: Call frame encryption parameters

Parameter	Value
Frame cipher	AES-256-GCM via WebRTC insertable streams (frame cryptor)

⁵¹LiveKit project, "LiveKit open source WebRTC SFU" and "End-to-end encryption" documentation, github.com/livekit/livekit and docs.livekit.io.

Parameter	Value
Keying	One random 256-bit sender key per participant device (not a shared room key)
Ratchet	Every 200 ms the sender advances its key: next = HKDF-SHA256(key, fixed salt, 32 bytes); receivers follow the same ratchet within a window of 16 steps
Key ring	16 key indices per sender
Membership change	On every join or leave, each sender generates a new random key at the next index and sends it only to the current participants; senders switch when all acknowledge or after 750 ms. A joiner never receives an earlier key and a leaver never receives a later one
Key transport	Each sender key is the plaintext of a libsignal message to each recipient device, relayed by the API without storage, bound to the call identifier, and accepted only between members of a live call with trusted devices; it is sent only to devices on the participants' signed device lists (Chapter 13); a monotonic generation counter rejects replayed or older keys
Mobile implementation	Patched WebRTC SDK with 256-bit frame keys at all key-derivation sites; the build refuses artefacts whose SHA-256 does not match the pinned manifest
Desktop implementation	livekit-client key provider configured for 256-bit HKDF per-sender keys
Fail closed	Clients refuse to publish media without end-to-end encryption and their own key

The SFU forwards encrypted frames and cannot decrypt their payloads. This design is comparable in intent to the IETF SFrame approach⁵² but uses LiveKit's frame-cryptor format rather than RFC 9605 wire encoding.

19.3Key Rotation Rate

Because every participant ratchets its sender key every 200 milliseconds, a call derives **five new frame-encryption keys per second for each sender**, continuously, for the whole call, whatever its length. On top of that schedule, every join or leave triggers a fresh random key. A key captured from memory at one moment therefore decrypts at most the media sent in that 200 ms epoch and those that follow until the next change of membership; earlier epochs cannot be recovered from it, because each step is a one-way HKDF derivation. Messages are keyed even more finely: the Double Ratchet derives a new key for every single message (Chapter 12).

Table 36: Key renewal in calls and messages

Channel	Key renewal	Rate
Call media (each sender)	HKDF ratchet of the sender key	5 keys per second (every 200 ms)
Call media (membership change)	New random 256-bit sender key distributed over libsignal	On every join and leave
Messages	Double Ratchet symmetric step, plus a Diffie-Hellman and SPQR step on each change of direction	A new key for every message
Session establishment	PQXDH with a one-time prekey where available	Per new session

19.4Transport Encryption (Client to SFU)

WebRTC media is protected with SRTP keyed by DTLS.⁵³⁵⁴ SANKET's LiveKit build sets AEAD_AES_256_GCM as the only acceptable SRTP protection profile;⁵⁵ the build includes proof tests with real handshakes in both DTLS roles

⁵²E. Omara et al., "Secure Frame (SFrame): Lightweight Authenticated Encryption for Real-Time Media", IETF RFC 9605, August 2024.

⁵³M. Baugher et al., "The Secure Real-time Transport Protocol (SRTP)", IETF RFC 3711, March 2004.

⁵⁴D. McGrew, E. Rescorla, "DTLS Extension to Establish Keys for SRTP", IETF RFC 5764, May 2010.

⁵⁵D. McGrew, K. Igoe, "AES-GCM Authenticated Encryption in SRTP", IETF RFC 7714, December 2015.

and fails if a client without AES-256-GCM is accepted. Clients independently inspect the negotiated suite on both transports shortly after connecting and periodically thereafter, and end the call if it is anything other than AEAD_AES_256_GCM.

NOTE - DTLS HANDSHAKE SUITE

No voice, video or message data is ever protected by the DTLS cipher. WebRTC uses DTLS only as a key-agreement handshake, negotiated with an ECDHE-ECDSA suite (AES-128-GCM in the upstream WebRTC library, which does not offer AES-256 DTLS suites). The SRTP master keys are exported from the ECDHE secret rather than encrypted under that suite, and the DTLS records carry only the handshake and an unused data channel. Media confidentiality therefore rests entirely on AES-256: AES-256-GCM frame encryption inside AES-256-GCM SRTP. Mobile and desktop compute the same verdict on the negotiated handshake suite by one shared rule and record it in every call's log, so an assessor can confirm it on any installation; the verdict is evidence and never ends a call.

19.5Operation on 3G, 4G and Constrained Networks

SANKET calls are engineered for narrow and variable bandwidth such as 3G and congested 4G links. One shared network-profile module decides every encoding parameter, so that iOS, Android and desktop behave identically.

Table 37: Codecs and adaptation for constrained networks

Element	Behaviour
Audio codec	Opus in every call, with discontinuous transmission (DTX) so that silence costs almost no bandwidth, and Opus in-band forward error correction to conceal packet loss
Audio bit rate	24 kbps (speech quality) on a normal link; 12 kbps (telephone quality) when the link is constrained
Video codec	VP8 with simulcast: the sender publishes up to three quality layers, and the media server forwards a lower layer to a receiver on a weak link without the sender re-encoding
Automatic adaptation	A network-adaptation state machine watches link quality; on a poor link it lowers the audio bit rate and pauses outgoing video, so that voice remains intelligible, and restores both when the link recovers
Transport	Media over a single multiplexed UDP port, with a TCP fallback and, where only TCP 443 is open, TURN over TLS on port 443; signalling over the authenticated WebSocket, which reconnects with jittered back-off after a drop
Networks	3G, 4G, 4.5G and 5G mobile networks, Wi-Fi and wired networks

19.6Server Enforcement

- The server refuses a room token to any device below the AES-256 call profile, refuses any offer that does not declare it, and refuses to ring anyone without a capable device; a refused call is never relayed or rung.
- A device on a call is marked busy and is not rung again; its other free devices still ring.
- Call signalling travels over the authenticated WebSocket. The call offer carries the caller's display name and call type, which the server can read.
- Call wake-up pushes carry only routing identifiers and a generic title; the caller's name is filled in on the device from its local cache. The room token is fetched afterwards over the authenticated API.

19.7Calls in Isolated and Restrictive Networks

Many hotel, ministry and mobile networks allow only outbound TCP on port 443. On an internet-facing installation one public TCP 443 port carries both HTTPS and TURN, separated at the edge without decrypting. TURN over TLS

is terminated by the edge with TLS 1.3 and TLS_AES_256_GCM_SHA384 only (AES-128 and ChaCha20 are refused), so the media server no longer holds that certificate key, and the relay is restricted by SANKET's build of the media server to the node's own media address, so it cannot be used as an open relay. Inside that TLS tunnel the media keeps its AES-256-GCM frame encryption and AES-256-GCM SRTP. TURN is used on internet-facing installations and not on air-gapped ones, where no outside client exists; UDP media remains the primary path. Calls have been demonstrated over TCP 443 alone on iOS, Android and desktop.

Server upgrades are designed to preserve calls in progress: services are recreated only when their image or configuration changes, and where a release allows it the new application server takes over from the old one without interrupting call media; other releases are applied in a planned maintenance window.

Every installation sets the media server's address explicitly: the public address of an internet-facing server, or the enclave address of an air-gapped one. The media server advertises that address to calling devices and never contacts a public STUN server to discover it, and SANKET's build of the media server no longer hands calling devices the public STUN servers the upstream release supplies by default, so neither side of a call reaches a foreign STUN service; the installer, the upgrade tool and the install-time security gates refuse a non-local installation without it, and refuse a configuration that would turn STUN discovery back on (Chapter 43).

19.8 No Recording

SANKET has no server-side call recording capability and no lawful-intercept interface, and its design excludes them: the server never holds frame keys.

20 Metadata Security

End-to-end encryption hides what is said. It does not, by itself, hide who said it, to whom, when, how much or from where. For many high-trust organisations that metadata is as sensitive as content, so this chapter sets out exactly what SANKET's server handles.

20.1 Metadata Inventory

Table 38: Metadata handled by the server in the described release

Metadata	Visible to server?	Why it is needed	Minimisation and retention
Sender account and device of each message	Yes	Authorisation, routing, receipts, audit	Required for accountability in an organisational platform; sealed sender is not provided
Recipient devices of each message	Yes	Fan-out delivery	Per-device rows tombstoned 72 h after delivery
Conversation membership and group names	Yes	Access control	Group names and descriptions are stored in clear
Timestamps and delivery / read state	Yes	Ordering and receipts	Read receipts and typing can be disabled by policy and user
Ciphertext size	Yes, as a size band	Inherent to any transport	Padded to 160-byte blocks; disappearing messages and short server-side retention limit its value
Message adjuncts (reply excerpts, link previews, locations, contact cards, forwarded-from)	No (inside ciphertext)	Carried in the end-to-end encrypted content envelope	Not stored by the server outside the ciphertext; each feature controllable by policy
Reactions	No (inside ciphertext)	Sent as end-to-end encrypted messages; each device computes the counts	Opaque per-device rows tombstoned like messages; optional emoji-free count for large groups, off by default (the server then learns who reacted to which message, never the emoji); disable by policy
Classification labels	No (inside ciphertext)	Carried in the end-to-end encrypted envelope for each message and attachment	The label list itself is tenant configuration, never sent to unauthenticated callers; devices report coarse label events to the audit log without message or conversation identity (Chapter 38)
Encrypted broadcasts: priority, audience, deadline, expiry, receipts	Yes (title, body and attachments: No, inside ciphertext)	Fan-out to every recipient device, acknowledgement, deadline and escalation	Per-device ciphertext deleted once each device confirms storage, otherwise at expiry; push and alerts carry identifiers, priority and deadlines only (Chapter 17)
Notice broadcasts: title, body, attachments	Yes	Non-sensitive announcements from the administrator console, labelled as not end-to-end encrypted	Retention 90 days by default

Metadata	Visible to server?	Why it is needed	Minimisation and retention
Call metadata (participants, type, start, duration, caller display name in offer)	Yes	Signalling, one-call-per-device, call logs	Call logs retained 365 days by default and rolled up
Device information (platform, model, app version, hardware protection level, client certificate fingerprint)	Yes	Compatibility, update enforcement and device governance; binding the device certificate to the device record	Shown to administrators; device certificates carry an opaque random name, no user or device identifier
IP addresses and user agents	Yes	Abuse prevention and audit	In audit records; masked identifiers in edge logs
Sign-in and security events	Yes	Security and accountability	Audit log retained per policy (Chapter 21)
Account profile (Sanket ID, display name, email, phone)	Yes	Directory and verification	Phone and email optional; stored in clear
Identity-provider mapping (when federated sign-in is configured)	Yes	Linking the customer's own identity provider or directory to administrators and to Sanket IDs	No account is created from the provider; provider secrets held in the key service (Chapter 13)
Signed device lists	Yes	Letting senders verify every recipient device before encrypting	Append-only log of each account's signed lists; reveals the same devices and device kinds as the device inventory
Usage analytics	Yes, pseudonymous (when on)	First-party usage statistics	On by default; the organisation can switch it off and an installation-level lock can force it off, after which nothing is collected; counts only, no content, no third party; rollup retention configurable (Chapter 22)
Push wake-ups	Partially (to relay)	Waking devices	Content-free: sealed or generic payloads carrying only routing identifiers, with fixed size and decoy traffic where the platform allows. Wake-ups for encrypted broadcasts carry no content, only a generic text by priority

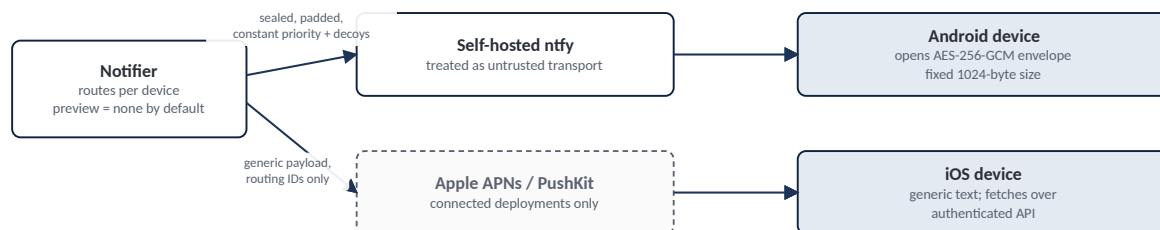
20.2 What SANKET Does to Minimise Metadata

- **Self-hosting** keeps all metadata within the customer's boundary; nothing is sent to Tosh Defence except, if enabled, a signed status report containing counts of users, administrators and groups and the release version.
- **Content-free notifications** by default: no sender name and no preview.
- **Reactions, encrypted broadcasts and classification labels inside the ciphertext**, so the server holds none of their content.
- **Sealed wake-ups** that hide which events occurred, padded to a fixed size and mixed with decoy traffic where the platform allows.
- **Non-enumerable directory**: users can find others only by exact Sanket ID or among people they already share a group with.
- **Short server-side ciphertext lifetime** and configurable retention windows.

- **Edge logs mask identifiers** that appear in request paths.
- **Last-seen visibility** can be restricted to contacts or nobody.

Notification Path

Wake-ups carry no message content; Android payloads are sealed per device



What a push can reveal

ntfy: that a device received an envelope at a time (cover traffic blurs this). APNs: that an iOS device was woken; title and body are generic unless the tenant enables sender names. Encrypted broadcasts show generic text by priority. Air-gapped iOS: no background wake-up; messages are delivered when the app is opened.

Figure 12: Notification path

20.3 Content Confidentiality Versus Traffic-Analysis Resistance

SANKET provides strong content confidentiality. It does not provide traffic-analysis resistance against the operator of the installation, nor against a network observer who can see device-to-installation traffic volumes and timing. Systems that resist traffic analysis use techniques such as sealed sender, constant-rate cover traffic, mixing or onion routing,⁵⁶ which carry significant cost. SANKET does not provide sealed sender: the server sees the sending account and device of every message. For most customers the decisive mitigation is that the operator is the customer itself; where insider metadata analysis is a concern, access to the database and audit records should be separated and monitored (Chapter 27).

⁵⁶Signal, "Technology preview: Sealed sender for Signal", signal.org/blog/sealed-sender/, October 2018.

21 Audit Architecture

SANKET makes the use and administration of the system accountable without making communications readable. Audit records describe security and administrative events; they never contain message content, ciphertext, keys or file contents.

Table 39: Separation of content and audit metadata

CONTENT (never audited)	SECURITY AND ADMINISTRATIVE METADATA (audited)
Message bodies, attachments, call media, keys, ciphertext	Who did what, to which object, from which device and address, when, with what result and severity

21.1 Events Recorded

Every audited action is mapped to a category, severity and result. Events cover registration and sign-in, administrator authentication, threat detections, data access (never content), calls, the device lifecycle, classification labels, administration and integrations.

Each record carries an event identifier, the actor and actor type (user, administrator, system or integration), action, resource, target, IP address, user agent, endpoint, device, category, severity, result and timestamp. End users are identified by Sanket ID in exports and in the console; internal identifiers are not exposed in the exported stream. Integration calls record the route pattern rather than the full URL, so that a credential mistakenly placed in a query string is never written to the log. End users can review their own recent security activity.

21.2 Tamper Evidence

The audit log is a hash chain. Each record stores a sequence number, the previous record's hash and its own hash, computed with SHA-256 over the previous hash, the sequence number and the record's fields. Writes join the chain one at a time under a database lock. A verifier, available to holders of the audit-read permission and run by the licence-and-audit monitor, walks the chain and reports sequence gaps, broken links, recomputation mismatches and deleted rows. Exported events carry their hash links so that a SIEM can check continuity independently.⁵⁷

The chain head is signed at regular intervals with an Ed25519 key held non-exportable in the tenant's key service, and the newest signed anchor is also recorded outside the database. The verifier checks every anchor's signature, that the anchors form an unbroken sequence and that the chain passes through each anchored hash, so a database superuser who rewrites history and recomputes every later hash to hide it is still detected, as is a forged, altered or deleted anchor. The pull API and every syslog message carry the latest anchor, so a receiver that pins the public key can check it independently.

RECOMMENDED PRACTICE - AN INDEPENDENT COPY

The hash chain makes alteration, insertion or deletion of records detectable, and the signed anchors make a rewrite detectable even by a database superuser. For the strongest assurance, customers forward the audit stream to an external SIEM or write-once archive administered by a separate team: that independent copy, with the anchors it receives, holds the chain outside the installation and satisfies WORM retention requirements.

⁵⁷K. Kent, M. Souppaya, "Guide to Computer Security Log Management", NIST SP 800-92, September 2006.

21.3 Access, Retention and Export

- **Access:** reading the audit log requires the audit-read permission. The Owner, User Manager, Threat Monitor and Auditor roles include it; the Config Manager role does not.
- **Retention:** audit records are retained indefinitely by default to preserve chain continuity; a retention period can be configured (default setting 365 days) and pruning enabled explicitly by the operator.
- **CSV export** from the administrator console.
- **Syslog:** RFC 5424 messages framed per RFC 5425 over TLS (AES-256-GCM suites), with the customer's CA pinned rather than the system trust store, server name checked, at-least-once delivery in chain order, and a persisted cursor.⁵⁸⁵⁹
- **Pull API:** paginated audit events by Sanket ID with hash links, through a scoped integration credential.
- **Webhooks:** device and administrator security events, signed with HMAC-SHA256 over a timestamp and body.

21.4 Administrators Cannot Audit Plaintext

Because audit is metadata-only and the server holds no content keys, there is no audit view, export or administrative action that reveals message plaintext. Organisations that need content supervision must implement it at the endpoint under their own policy; SANKET does not provide a server-side mechanism for it.

⁵⁸R. Gerhards, "The Syslog Protocol", IETF RFC 5424, March 2009.

⁵⁹F. Miao, Y. Ma, J. Salowey, "Transport Layer Security (TLS) Transport Mapping for Syslog", IETF RFC 5425, March 2009.

22 Monitoring, Threat Detection and Observability

SANKET's detection capabilities are deterministic and rule-based. This chapter separates the protections that are always active from the optional components, and is explicit that the platform contains no machine-learning detection.

22.1 Always-Active Protections

Table 40: Built-in detection and response

Control	Behaviour	Type
Account lockout	Locks after configurable failures; alerts administrators	Deterministic
Compound and global rate limits	Per IP and account, per user and device, at edge and API	Deterministic
Adaptive rate limiting	Tightens limits in steps as the error rate rises, with gradual recovery	Statistical (simple)
Refresh-token reuse detection	Revokes the device; critical event	Deterministic
Block manager	Hard or soft blocks of users, devices, addresses and networks	Administrator action
Protective degraded mode	Administrator-triggered degraded mode that throttles non-essential traffic	Administrator action
Integration abuse limits	Failure throttles and change caps with critical alerts	Deterministic
Lost-device contact, duress, locate	Critical and high-severity events and webhooks	Deterministic
Administrator security keys	Anomalous security-key use is refused and raises a critical alert to the other owners	Deterministic
Alerts feed	Real-time administrator view of security events from the audit log	Deterministic

22.2 Optional Stream Processor

An optional stream processor, deployed as a separate service with its own database, correlates security events over time and raises alerts that appear in the administrator console in real time, naming end users by Sanket ID. Most customers combine the built-in protections above with correlation in their own SIEM.

22.3 What Is Not Claimed

SANKET's detection is deterministic and rule-based; no machine-learning detection is claimed.

For most high-trust customers the right model is to forward SANKET's audit stream to their own security operations platform and correlate it there with network, endpoint and identity telemetry.

22.4 Usage Analytics on the Installation

Usage analytics (active users, messages and calls per day, app versions and platforms) are first-party: they are aggregated by a processor that ships with every release and runs on the installation itself. Clients and the server add content-free, pseudonymous events to a capped stream inside the installation; the processor rolls them up into counters in the installation's own database, in day buckets in the organisation's time zone, which the console's Analytics pages read. Only counts are kept, never content. Nothing leaves the installation and no third-party analytics SDK or service is used. The processor is signed, verified and upgraded like every other service, runs with a read-only file system and no added privileges, and publishes no port.

Table 41: Control of usage analytics

Control	Behaviour	Status
Organisation switch	On by default. An administrator can switch analytics off (or on again) on the console's analytics settings page; the change needs a fresh TOTP step-up, is recorded as a high-severity audit event with the old and new value and raises an administrator alert	CONFIGURABLE
Installation lock	Set at provisioning for customers who must not collect usage data at all (for example intelligence organisations): analytics is then always off, and no console action, vendor tool or stored setting can turn it on; the console shows that the installation has turned it off	OPTIONAL
When off	Mobile and desktop collect nothing and discard events they had queued; the server refuses new events and stores nothing; the processor does no work; events already waiting in the stream are cleared at once, so nothing queued before the change is ever processed; the Analytics pages say that analytics is off	IMPLEMENTED
Delete all collected analytics	Empties the event stream and every rollup, after a fresh TOTP step-up and a typed confirmation, and is audited; it never touches audit logs or device records	IMPLEMENTED
Retention	Rollups are kept for a configurable period (default 400 days, minimum 30 days)	CONFIGURABLE
Out of scope of the switch	Security audit logs, and the app version, platform and build that each device reports for update enforcement, are operational records rather than analytics and stay on	BY DESIGN

Analytics never sits on a user's path. On the devices an event is only appended to a bounded in-memory queue, never awaited on a send, receive, decryption, call or rendering path, and sent in one batched request on an interval, deferred during calls and start-up; on the server, accepting events is a single capped append to the stream with no database write, and the processor runs in its own container under fixed resource limits, so it may fall behind but never slows the API. Written performance budgets, each enforced by a test, show that message sending, delivery and calls are unaffected with analytics on.

The processor's health is visible and alerted. Every Analytics page reports whether the processor is running, when the newest rollup was made and how many events are waiting, and shows a banner instead of empty charts when the processor is down or has stalled. A scheduled monitor raises an administrator alert when the backlog of unprocessed events, or the age of the oldest waiting event, passes a configurable threshold, and clears it when the processor recovers.

22.5 Observability Without Content Exposure

Administrators with the alerts-read permission see live operational monitors that read only counters, health probes and metadata:

Table 42: Operational monitors

Monitor	What it shows
Overview and synthetic checks	Composite counters; live Redis write and read, database round trip, bucket check
Infrastructure topology	Component graph with live reachability
API and WebSocket	Request and error rates, latency percentiles, busiest routes; session, user, device and platform counts
Database and storage	Connection saturation, sizes, largest tables, long-running queries; object-store health and usage
Retention and jobs	Partition inventory, last retention run, recent scheduled jobs
Broadcast and calls	Delivery and acknowledgement progress; call activity
Cryptographic keys	Signal key inventory and one-time prekey exhaustion
Licence and audit	Plan, expiry, seats, audit chain head and verification
Identity provider	Where federated sign-in is configured, the result of the latest check of the customer's identity provider or directory
Analytics processor	Running state, newest rollup and backlog of waiting events
Backup and recovery	Backup policy, run and restore state
Capacity and cost	Scaling signals; seats, bytes, message and call volumes

The edge records, for each TLS connection, only the time, the protocol version, the negotiated key-exchange group and the server name, with no client address, path or user, so the share of connections that use hybrid post-quantum key exchange can be counted without identifying anyone (Chapter 39).

A health endpoint reports database and licence state, and container health checks cover the principal services. A metrics endpoint is available internally and is not exposed through the edge. Tracing can be exported to a customer-operated collector. None of these channels carries message content; a container-log viewer for administrators redacts passwords, tokens and credential patterns before display and records every viewing in the audit log.

NOTE - CONTAINER-LOG VIEWER AND HOST PRIVILEGE

The standard installation gives the application no host-level access for reading container logs; the administrator log viewer works only after an explicit operator opt-in. Customers with strict separation requirements collect container logs with their own host agents.

23 Air-Gapped Deployment

An air-gapped SANKET installation runs on a network with no path to the public internet. Everything it needs at run time is inside the enclave, and everything it needs from outside arrives on controlled media.

23.1 Definition

For the purposes of this document, a true air-gapped deployment is one in which the installation, its clients and its supporting services operate with no runtime access to any of: public DNS, cloud APIs, SaaS identity providers, public push services, external certificate authorities, content delivery networks, external analytics or telemetry, internet time servers, public package repositories, cloud key management, public object storage or external licence validation. SANKET supports this mode; the remaining dependencies are a function of the customer's client devices, as set out below.

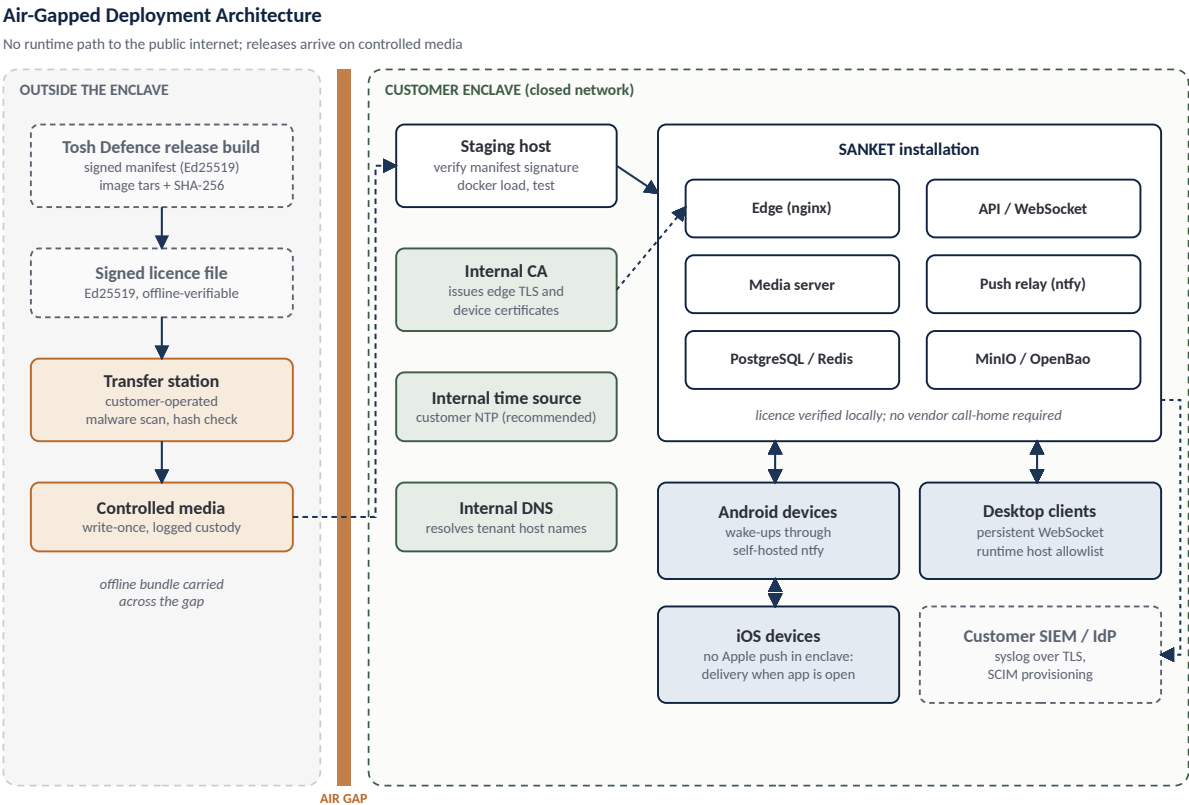


Figure 13: Air-gapped deployment architecture

23.2 Replacing External Dependencies

Table 43: Conventional external dependencies and their air-gapped alternatives

Conventional external dependency	SANKET air-gapped alternative	Status
Public certificate authority	Certificate issued by the customer's internal CA, installed in import mode; internal CA trusted by clients	IMPLEMENTED
Cloud object storage	MinIO in the installation	IMPLEMENTED
Cloud database and cache	PostgreSQL and Redis in the installation	IMPLEMENTED

Conventional external dependency	SANKET air-gapped alternative	Status
Cloud key management	OpenBao in the installation (non-exportable keys)	IMPLEMENTED
Public push services (Android)	Self-hosted ntfy (UnifiedPush) with sealed payloads	IMPLEMENTED
Public push services (iOS)	No equivalent: iOS background wake-up requires Apple's service. Messages are delivered when the app is in the foreground or reconnects	DEPLOYMENT-DEPENDENT
Hosted media service and STUN	Self-hosted SFU with an explicit node address; public STUN is never used; TURN is off, because every client is inside the enclave	IMPLEMENTED
Device attestation service (Android)	Android Key Attestation verified offline by the installation against vendor root certificates and a revocation snapshot shipped in each release; no runtime call to Google	IMPLEMENTED
Hosted identity provider	None required. Optional OIDC or SAML for administrators and LDAP for end-user enrolment only against the customer's own identity provider inside the enclave; public identity providers are refused	OPTIONAL
Online licence validation	Ed25519-signed licence verified locally; offline renewal bundles	IMPLEMENTED
Online package registry and updates	Offline release package with signed manifest, image archives and SBOMs, verified before load; the signed manifest's sequence number and expiry protect against rollback and stale releases without any online check	IMPLEMENTED
Third-party base images	Pre-staged by the customer from a verified source into a local registry or archive	CUSTOMER RESPONSIBILITY
Cloud logging and SIEM	Syslog over TLS to the customer's internal SIEM; local monitors	IMPLEMENTED
Internet NTP	Customer's internal authoritative time source; SANKET detects clock rollback for licensing	RECOMMENDED
Public DNS	Internal DNS for the installation's host names	DEPLOYMENT-DEPENDENT
SMS and email providers	SMS disabled; email via the in-installation relay to internal mail, or disabled	CONFIGURABLE
External analytics / crash reporting	Not used by product applications	IMPLEMENTED
Vendor status reporting and connected upgrade	Offline-mode licence: no status report; upgrades by offline package only	CONFIGURABLE

23.3 Client Devices in an Enclave

- **Android** devices connect to the installation over the enclave network and receive wake-ups through the self-hosted push relay. Applications are distributed through the customer's managed channel.
- **Desktop** clients maintain a persistent connection while running and are restricted at run time to the installation's own hosts.
- **iOS** devices can run SANKET in an enclave, but without Apple's push service they are not woken in the background; incoming messages and calls are received when the app is open. Distribution and device enrolment of iOS devices also depend on Apple's infrastructure at provisioning time.

23.4 Operational Considerations

- **Time.** Token validity, TOTP and licence checks depend on accurate clocks. Provide an internal authoritative time source to servers and devices.
- **Certificates.** Plan the internal CA hierarchy and certificate renewal before installation (Chapter 26).
- **Releases.** Establish a controlled media transfer procedure with malware scanning and custody records (Chapter 30). A release manifest expires, and installation refuses an expired one. Bring current releases into the enclave before they expire; an installation that is already running never stops because its manifest expires.
- **Device attestation data.** Attestation roots and revocation data arrive with each release; keep releases current.
- **Licences.** Renewals are delivered as signed offline bundles; the platform rejects bundles for another tenant and enforces a monotonic renewal counter.
- **Backups.** Hold encrypted backups on separate media inside the enclave and test restoration (Chapter 32).

SECURITY PROPERTY - RESULT

In an air-gapped installation, no SANKET component makes a connection outside the enclave (device attestation, identity providers and release checks all work offline), and no vendor or third party can observe, alter or interrupt the service. The residual dependencies are the customer's own infrastructure and, for iOS devices, the limits of the operating system.

24 Offline-First Operation

SANKET is designed for intermittent connectivity. Clients keep working with local data while disconnected, and the server holds encrypted messages until devices return.

KEY POINT - OFFLINE-CAPABLE IS NOT PEER-TO-PEER

Offline-capable means that a device can compose and read while disconnected and that messages are stored and forwarded when connectivity returns. It does not mean that two devices can communicate with each other without a SANKET installation between them. SANKET has no peer-to-peer or mesh mode in this release.

24.1Store and Forward

Table 44: Behaviour under intermittent connectivity

Aspect	Behaviour
Outbound while offline	Mobile and desktop hold composed text messages, attachments and voice notes in an encrypted local outbox (mobile: inside the SQLCipher database; desktop: in its encrypted secure storage) for up to seven days and send them on reconnection. An attachment or voice note is kept only as a copy encrypted with the local cache key (desktop: AES-256-GCM in the main process file cache, never handed to the renderer), never in clear, and is uploaded again if its upload had not finished; one the server refuses stays in the conversation as failed, with Retry and Discard, also after the app restarts. Mobile and desktop decide which failures to retry by the same rule (Chapter 17)
Inbound while offline	Per-device ciphertext waits on the server until the device fetches it
Identity changes while offline	Notices that a contact's identity key changed are kept by the server and replayed to each device that was offline when the key changed, so no device misses the warning
Encrypted broadcasts	A recipient device that is offline receives its encrypted copy when it reconnects, until the broadcast expires. A device provisioned while a broadcast is still live is queued, and the sender desktop encrypts the broadcast for it
Reconnection	Mobile WebSocket reconnects with jittered back-off; sync cursors fetch only what is new
Ordering	Messages carry server timestamps; the client orders by them
Deduplication	Idempotency keys make retries safe; libsignal rejects duplicate ciphertext
Stale sessions	Repeated decryption failures on recent messages trigger a controlled session reset and a fresh handshake
Expired credentials	Access tokens refresh on reconnection, tolerating lost responses; revoked or lost devices are refused
Clock drift	TOTP tolerates small clock drift; licence time is protected by rollback detection; servers should use an internal time source
Message expiry	Disappearing messages are deleted on the server at expiry and on devices when they next run the retention check
Queue protection	Server queues hold only ciphertext; local outboxes are inside the encrypted database, and a queued file only as an encrypted copy that is deleted once it is sent, dropped or discarded
Queue size	Per-user send rates and fan-out caps bound queue growth; delivered rows are tombstoned
Low bandwidth	A narrowband call profile reduces audio and video bit rates for constrained links

25 Network Architecture and Segmentation

A SANKET installation exposes a deliberately small network surface. This chapter describes what the released profile exposes, how internal traffic flows, and the reference architectures Tosh Defence recommends for different environments.

25.1 Exposed Surface of the Released Profile

Table 45: Network exposure of an installation

Listener	Protocol	Purpose	Exposure
Edge	TCP 443	HTTPS: API, WebSocket, administrator console, media signalling, push relay. TURN over TLS: media relay for networks that allow only TCP 443	Published (the only public TCP port)
Media	UDP (single multiplexed port)	Voice and video media to the SFU; the primary, lowest-latency path	Published
Optional media fallbacks	TCP	Media fallback where UDP is blocked, enabled by the customer	Optional
Application and data services	Internal	API, database, cache, object storage, key service	Not published; no host ports

One public TCP port. The edge separates HTTPS and TURN on TCP 443 without decrypting the connection, and carries the real client address to the services behind it, so rate limits, the administrator network allowlist and audit records judge the real client; forwarding headers sent from outside are never believed. The TURN hop is terminated by the edge with TLS 1.3 and TLS_AES_256_GCM_SHA384 only, the media server holds no certificate key, and the relay reaches only the installation's own media address, so it is not an open relay. Calls over TCP 443 alone have been demonstrated on iOS, Android and desktop.

Internally, the key service sits on a dedicated internal-only network reachable only from the components that need it, and the other services share an isolated application network. Inter-service traffic never leaves the host in the single-host profile. In the high-availability profile, database replication and the OpenBao cluster use TLS between hosts; place inter-host links on a dedicated network.

one host. Each host's role and every peer address are set in the installation's environment file and role overlays; the files shipped in a release stay identical for every customer. Chapter 32 describes the topology and its failure behaviour. Status: IMPLEMENTED.

25.2.5 Multi-site and disaster-recovery deployment

A primary site with an asynchronously replicated second site: database write-ahead logs archived to the second site, object storage replicated between sites, and encrypted OpenBao snapshots shipped to it. Drill tooling and a runbook cover declaring a disaster, fencing the primary, promoting the second site, switching names and failing back. The reference topology is designed for a recovery point objective of 5 minutes and a recovery time objective of 1 hour; these are design targets, and each deployment's disaster-recovery drill records the figures it actually measured (Chapter 32). Independent installations per site remain an alternative where sites must not share data. Status: IMPLEMENTED.

25.2.6 Kubernetes-based deployment

Draft Helm charts exist, but the built high-availability profile is multi-host Docker Compose. Status: ROADMAP; not supported for production in the described release.

25.3 Network and Protocol Categories

Table 46: Flows to permit at customer firewalls

Flow	From	To	Notes
Client API and WebSocket	User access zone	Edge	HTTPS only
Client media	User access zone	Edge host media port	UDP media port preferred; ICE over TCP optional; TURN over TLS on TCP 443 for networks that allow nothing else
Administration (console)	Management zone	Edge	Restrict by source address at the firewall
Host administration	Bastion	Installation host	Key-based SSH with MFA on the bastion
Syslog	Installation	SIEM	TLS, outbound only
Webhooks	Installation	Customer endpoints	HTTPS, outbound only, to configured hosts
Identity provider (optional)	Installation	Customer identity provider	OIDC or SAML over HTTPS, LDAP over Idaps or StartTLS; the customer's own provider only
Site replication (high availability)	Primary site	Second site	Database log archive and replication, object-storage replication, encrypted OpenBao snapshots
Connected upgrade (optional)	Installation	Tosh Defence release registry	HTTPS, outbound only; connected-mode licences only
Backups	Installation	Backup zone	Encrypted archives
Apple push (connected, optional)	Installation	Apple push service	Only if iOS background wake-up is required
Everything else	Installation	Anywhere	Deny by default

26 PKI Architecture

SANKET uses three distinct trust systems, and it is important not to confuse them: the customer's X.509 PKI for transport, libsignal identity keys for messaging, and Tosh Defence's signing keys for licences and releases.

PKI Hierarchy

Customer PKI for transport; vendor signing keys for licences and releases; libsignal identity for messaging

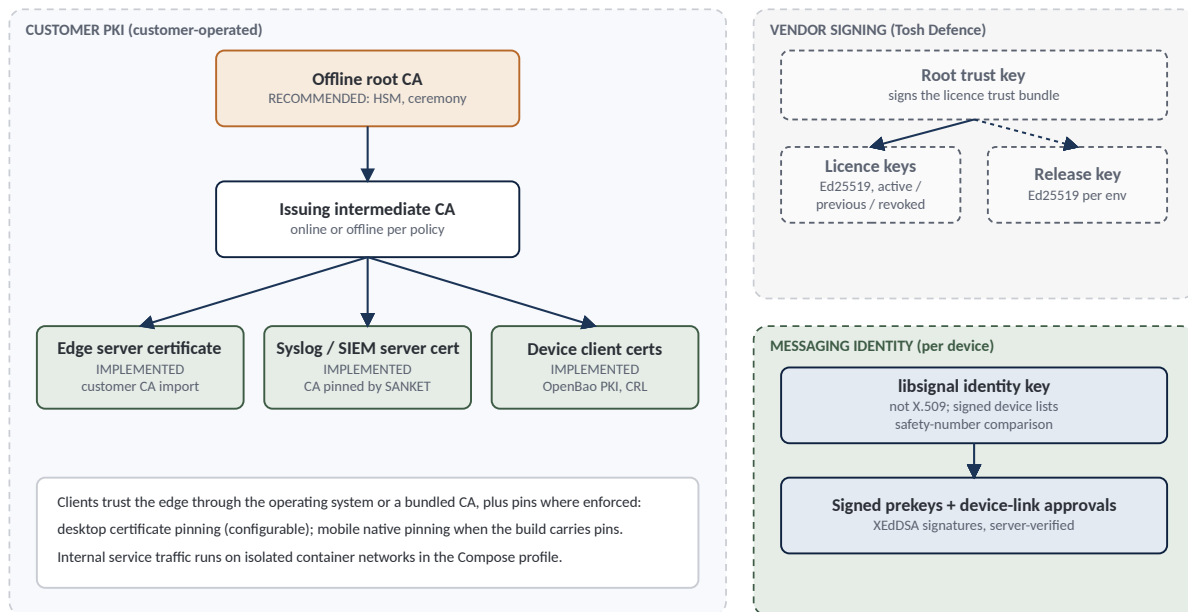


Figure 15: PKI hierarchy

26.1 Customer Transport PKI

Table 47: Transport PKI elements

Element	Role	Status
Offline root CA	Anchors the customer's trust; kept offline, ideally in an HSM, used only to sign intermediates	RECOMMENDED
Issuing intermediate CA	Issues server and service certificates	RECOMMENDED
Edge server certificate	Authenticates the installation to clients; installed in import mode	IMPLEMENTED
SIEM / syslog server certificate	Authenticates the customer SIEM to SANKET; SANKET pins the configured CA	IMPLEMENTED
Device certificate CA	An OpenBao PKI mount in the installation, chained to the customer CA; its signing key stays in OpenBao; a role fixes the key type, the names allowed (no client-chosen names) and the lifetime	IMPLEMENTED
Device client certificates	One per approved mobile and desktop device: ECDSA P-256 key generated on the device, CSR built on the device, opaque random name with no user or device identifier, short lifetime, re-minted before expiry; verified by the edge under tenant policy	IMPLEMENTED
Device certificate revocation	OpenBao certificate revocation list loaded by the edge; certificates deleted on the device on revoke, lost mode and wipe	IMPLEMENTED

Element	Role	Status
Administrator certificates	Client certificates for administrator access, applied at the customer's reverse proxy (administrators authenticate with FIDO2 security keys in the installation itself)	DEPLOYMENT-DEPENDENT
Revocation (CRL / OCSP)	Revocation status of server certificates	DEPLOYMENT-DEPENDENT

26.1.1 Lifecycle

- **Issuance:** certificates are issued by the customer's CA process; the private key is generated on the edge host or in the CA workflow and installed with restrictive permissions.
- **Rotation:** replace the certificate before expiry; the edge reloads without changing client trust as long as the chain is unchanged.
- **Expiry monitoring:** track certificate expiry in the customer's monitoring.
- **Revocation in disconnected networks:** clients in an enclave cannot reach public revocation services. Use short certificate lifetimes, an internal CRL distribution point or OCSP responder, and certificate pinning on desktop and mobile so that a substituted certificate is rejected.⁶⁰⁶¹
- **CA backup:** hold the root CA key and its backups in separate secure locations under dual control.

26.2Client Trust in the Edge

Clients trust the edge through the platform trust store (with the customer CA installed on managed devices) or a CA bundled in a customer-specific build, and pin the edge certificate on mobile and desktop wherever the deployment configures pins. Mobile pins are enforced natively, as a current and a backup pin compiled into the customer build and never accepted from the server. Rotating the edge certificate therefore means issuing the new certificate on the backup key, or shipping a build with the new pins before the change. The edge certificate must also name the TURN host when the media relay runs on TCP 443, because the edge terminates that TLS hop too.

Trust also runs the other way. Under tenant policy, each device presents its own client certificate to the edge on the REST API, the chat WebSocket and the media signalling connection, on iOS, Android and desktop; a device presents it only to the tenant's own hosts. The certificate name is an opaque random handle that never identifies the user; the mapping from handle to device exists only in the installation's database. The edge passes the verified fingerprint to the API, which binds it to the device record (Chapter 16).

26.3Messaging Identity Is Not X.509

libsignal identity keys are not certificates and are not issued by any CA. Their authenticity rests on trust on first use, signatures over prekeys, server verification of those signatures, and out-of-band comparison of safety numbers. Which devices belong to an account is established by signed device lists: each account's list is signed with its own libsignal account list key, kept by the installation in an append-only log and verified by every sender before it encrypts (Chapter 13). This is intentional: a compromise of the transport PKI, including the device certificate CA, must not let anyone read messages or add a device that senders will encrypt to.

26.4Vendor Signing Keys

Tosh Defence signs licences and release manifests with Ed25519 keys held as non-exportable keys in its own OpenBao. Installations hold only the public keys: a licence trust bundle, itself signed by a root key and listing active,

⁶⁰D. Cooper et al., "Internet X.509 Public Key Infrastructure Certificate and CRL Profile", IETF RFC 5280, May 2008.

⁶¹S. Santesson et al., "X.509 Internet PKI Online Certificate Status Protocol - OCSP", IETF RFC 6960, June 2013.

previous and revoked licence keys, and a release verification key configured in the installation environment. A bundle cannot introduce its own trust anchor. Each release manifest also carries a monotonic sequence number and an expiry, and connected upgrades check a short-lived timestamp statement from the vendor registry signed with a separate key (Chapter 30). These signatures are classical Ed25519; hybrid Ed25519 plus ML-DSA signatures are on the roadmap and are held until an ML-DSA implementation has a published independent audit or an issued CMVP certificate (Chapter 39).

27 Administrative Security

Administrators of a SANKET installation have significant power over identities, devices and policy, but not over content. This chapter describes how that power is constrained and how customers should divide it.

27.1 Central Management Console

Every installation includes one administrator console from which the organisation manages its users, devices, policy and licence. Administration is delegated through roles (Chapter 15): each administrator sees and changes only what their role permits.

Table 48: Central console functions

Area	Functions
Users	Create, edit, suspend, reactivate and delete users; invitations and activation codes; roles; per-user device lists
Devices	Approve, revoke, mark lost or found, locate, remote wipe per device or per user; managed-device (MDM) status; hardware protection level and client certificate state per device; outdated devices by platform and app version
Licence	Plan, validity and days remaining, licensed features, user and storage limits; export of an offline renewal request and import of a signed renewal bundle for disconnected sites
Policy	Security, messaging, files, calls, groups, retention, notifications and integrations settings; classification labels; usage analytics; device client certificates and required protection level
Administration	Built-in and custom administrator roles under a permission ceiling; administrator security keys; the customer's own identity provider
Broadcasts	Broadcast-sender accounts (create, assign to named administrators, suspend); encrypted broadcasts shown as metadata and receipts only (audience, priority, deadline, expiry, acknowledgement progress); notice broadcasts composed, targeted, scheduled and sent from the console, labelled "Notice (not end-to-end encrypted)"
Monitoring and audit	Operational monitors, alerts, block manager, audit log with chain verification and export
Integrations	API clients, SCIM, syslog, webhooks, MDM templates, OpenAPI documentation

27.2 Platform Controls

Table 49: Administrative controls in the platform

Control	Implementation	Status
Dedicated administrator accounts	Administrator accounts are separate from end-user accounts; the database forbids giving an end user the administrator role	IMPLEMENTED
Mandatory MFA	A second factor is always required for administrators, regardless of tenant policy: a FIDO2 security key, or a TOTP code where security keys are not required	IMPLEMENTED

Control	Implementation	Status
Administrator security keys	FIDO2 security keys (WebAuthn) verified by the installation itself, with no external service, and required on new installations for every administrator. ES256 and EdDSA keys only, user verification required, and synced passkeys refused by default because they copy the key to a vendor cloud; a suspected cloned key is refused. The relying party is the customer's own console address. Recovery from a lost key is a narrow path recorded as a critical audit event	IMPLEMENTED
Identity provider	Optional sign-in through the customer's own identity provider: OpenID Connect or SAML 2.0 for the console, as a first factor only (the local second factor is still required), and an LDAP directory check for end users at enrolment, device restore and new-device sign-in only. The provider must be the customer's own host (public identity providers are refused); accounts are never created from the provider, an unmapped subject is refused and a group mapping can never grant the Owner role. At least one local administrator is always kept, local sign-in is never disabled, and provider secrets are held in the key service (Chapter 13)	OPTIONAL
Broadcast-sender accounts	Dedicated accounts, each with a Sanket ID, that alone can send encrypted broadcasts (Chapter 17): created, assigned to named administrators and suspended in the console, every step audited; excluded from contact search, from all-user audiences and from ordinary chats. Each has a mobile trust root held by the unit's signal office and exactly one sending desktop, approved both by that mobile and in the console. The console never sees an encrypted broadcast's text; it sends only notice broadcasts, labelled as not end-to-end encrypted	IMPLEMENTED
Classification labels	A customer-defined ranked list of up to eight labels and a default label (Chapter 38); every change audited; label names are never sent to unauthenticated callers	CONFIGURABLE
Usage analytics settings	Switch (fresh TOTP step-up, high-severity audit and alert), batching interval, rollup retention and "Delete all collected analytics"; disabled and shown as turned off when the installation lock is set (Chapter 22)	CONFIGURABLE
Device client certificates	Device mutual TLS at the edge, enforced under tenant policy (Chapter 26)	CONFIGURABLE
Device protection level	The hardware protection of each device's keys shown per device, with Android key attestation verified offline; policy can require a minimum level (Chapter 14)	IMPLEMENTED
Least privilege	Nine permission categories by four actions; five immutable built-in roles; custom roles capped at the creator's own permissions	IMPLEMENTED
Session protection	Access token in memory; refresh cookie HttpOnly, Secure, SameSite=Strict, path-scoped; device revocation applies to administrator devices; sessions end after the configured idle period (default 60 minutes) and absolute duration (default 24 hours), enforced by the server	IMPLEMENTED
Administrative audit	Every configuration, user, device, integration and broadcast change is recorded in the hash-chained audit log	IMPLEMENTED
Sensitive-action gating	Wipe and locate each require a specific permission plus policy and licence gates; syslog target changes are critical audit events	IMPLEMENTED
Outdated devices	Per platform, how many active devices run each app version and how many are below the minimum, with a list by Sanket ID and a daily adoption chart kept as counts only; server-side refusal of outdated apps is configurable	IMPLEMENTED
Dual control / four-eyes approval	Achieved through role separation and audit review; in-product approval workflows on the roadmap	ROADMAP

Control	Implementation	Status
Network restriction for administrators	The API accepts administrator sign-in, every request made with an administrator session and every administrator session refresh only from configured IPv4 and IPv6 address ranges, judged by the address the edge proxy saw; end users are not affected. A save that would exclude the saving administrator is refused. A firewall or reverse proxy in front of the console remains recommended	IMPLEMENTED (address ranges)
No vendor access	Tosh Defence has no account on, and no remote access path into, the customer's installation	IMPLEMENTED

27.3 Lost Security Keys

Recovery from lost security keys follows narrow paths that allow nothing but enrolling a new key, and every use is recorded as a critical audit event. An owner can revoke another administrator's keys, never their own, only by confirming with a fresh assertion from the owner's own security key.

27.4 Separation of Duties

A self-hosted installation involves several kinds of privilege, only some of which are inside the SANKET application. Customers should assign them to different people.

Table 50: Recommended separation-of-duties matrix

Duty	Organisation administrator	Security administrator	Auditor	Infrastructure administrator	Database administrator	PKI administrator
Create and suspend users	Yes	-	-	-	-	-
Approve, revoke, wipe devices	Yes	Yes	-	-	-	-
Change security policy	Yes	Propose	-	-	-	-
Review alerts and audit	Read	Yes	Yes	-	-	-
Configure integrations and SIEM	Yes	Review	Review	-	-	-
Host and container administration	-	-	-	Yes	-	-
Database access	-	-	-	-	Break-glass only	-
OpenBao unseal share	-	One share	One share	One share	-	One share
Certificate issuance	-	-	-	-	-	Yes
Backup passphrase custody	-	-	Witness	Yes	-	-

27.5 Recommended Practices

- **Privileged access workstations** for administrator console and host access, on a management network.
- **Break-glass accounts** for the console and the host, with credentials sealed under dual custody and every use reviewed.
- **Recovery credentials** (OpenBao unseal shares, backup passphrase, root CA) held by different custodians.

- **Database separation:** routine operation never requires direct database access; any direct access should be logged and reviewed, because database operators can read server-visible metadata.
- **Host separation:** the host is the root of trust for the installation. Host administrators should not also be SANKET organisation administrators.
- **Review cadence:** verify the audit chain and review administrator activity regularly; forward the audit stream to a SIEM administered by a different team.

RECOMMENDED PRACTICE - HOST PRIVILEGE

Anyone with root on the installation host can read every server-side secret and all server-visible metadata, and can modify the running software. End-to-end encryption still protects past message content from such a person, but they could alter the server to attack future sessions (Chapter 28). Host access is therefore the most sensitive privilege in a self-hosted deployment.

28 Security Impact of Server Compromise

A server compromise is serious in any system. The purpose of end-to-end encryption is to change what it means: an attacker who controls SANKET's servers should gain metadata and the ability to disrupt service, but not message or file content.

Tosh Defence analyses the compromise of every server-side component, from the messaging service and its data stores through the edge, certificate and release infrastructure to the application host and the infrastructure beneath it. Across all of them, three statements hold:

1. **Past content stays confidential.** Message bodies, file contents and call media were encrypted with keys the server never held.
2. **Metadata is exposed.** The server necessarily knows who communicates with whom and when (Chapter 20).
3. **Future sessions are the attack surface, within limits.** A party controlling the server can withhold or delay messages. It cannot silently add a device to an account, because senders encrypt only for devices on the account's signed, append-only device list, and it cannot downgrade or replay an encrypted broadcast, because priority, deadline and expiry travel inside the encrypted envelope. Safety numbers, device inventories and audit make any attempt at key substitution visible, which is why those checks form part of high-trust operating procedure.

Recovery from any server compromise follows one pattern: rebuild from a verified, signed release on trusted media, rotate server-side keys and secrets, force re-authentication, and review the audit record against the independent copy held outside the installation. Content confidentiality never depends on a server component remaining uncompromised; what a compromise can expose is the metadata the server already handles (Chapter 20) and the ability to disrupt service while the attacker remains in control.

29 Endpoint Compromise Analysis

End-to-end encryption does not protect plaintext on a fully compromised authorised endpoint. The endpoint is where messages are decrypted, displayed and typed; an attacker who controls it sees what the user sees.

Table 51: Endpoint threats and SANKET mitigations

Threat	What the attacker can do	SANKET mitigation
Malware with full device control	Read the screen, keystrokes and decrypted memory	Managed devices; minimal local plaintext; remote wipe after detection
Screen capture	Record displayed content	Screenshot and screen-recording blocking on mobile and content protection on desktop; in-chat screenshot alerts
Clipboard compromise	Read copied text	User awareness; managed-device clipboard policies
Memory scraping	Extract keys or plaintext from process memory	Keys stored under a hardware-backed wrapping key on phones; desktop keys confined to the main process; cache key zeroed on sign-out
Unlocked device in another's hands	Use the app as the user	App passcode or biometric lock; background lock timeout; duress PIN
Rooted or jailbroken device	Bypass OS protections	MDM compliance blocks non-compliant devices; SANKET detects root and jailbreak indicators, alerts administrators and can refuse use; Android Key Attestation verified offline; remote wipe and revocation
Copied app storage	Copy the app's files and keychain items to another device and open them there	Protocol store and device secrets encrypted under a non-exportable Secure Enclave, StrongBox or TEE key; items copied to another device do not decrypt
Physical access to a locked device	Forensic extraction	OS storage encryption; keys bound to this device and to first unlock, wrapped by secure hardware on phones; SQLCipher; wipe after failed unlocks; lost mode; remote wipe
Malicious authorised user	Copy, photograph or forward content they can read	Download, export and forwarding restrictions, turned off automatically for content labelled above the lowest classification rank; visible watermark (Sanket ID or name) in the desktop and mobile file viewers (configurable); audit of file events

29.1 Mitigations in Combination

- **Managed devices** with MDM-enforced patching, posture checks and SANKET managed configuration (Chapter 14).
- **Secure local storage:** SQLCipher database and AES-256-GCM file cache; on phones the protocol store and device secrets encrypted under a hardware-backed key (Secure Enclave, StrongBox or TEE), with the protection level reported per device and a minimum level enforceable by policy.
- **Local lock:** passcode, biometric unlock, timeout, wipe after failures.
- **Screenshot controls:** screenshot and screen-recording blocking for high-trust deployments.
- **Revocation and wipe:** lost mode, device revocation and server-confirmed remote wipe.
- **Policy:** restrict download to device, export and forwarding; keep notification previews off.
- **Limited local data:** disappearing messages where the use case allows.

29.2 Desktop and Mobile

The desktop app runs the same protocol, the same message, call and broadcast semantics and the same certificate lifecycle as the mobile apps; differences between platforms lie in key storage, not in behaviour. Each device's key protection level is reported to administrators, policy can require a minimum level, and desktop use can be limited to managed machines where the endpoint threat model requires it.

NOTE - RESIDUAL ENDPOINT RISK

A device that is compromised before it is revoked exposes everything it could decrypt during the compromise. Revocation and wipe protect against continued exposure and against an attacker who obtains the device later; they do not undo exposure that has already happened. Plaintext that a user has legitimately exported, photographed or forwarded is outside SANKET's control.

30 Software Supply Chain and Secure Updates

A communications platform is only as trustworthy as the process that produces and delivers its software. SANKET gives the customer the means to verify a release independently and to decide when it is installed.

30.1 Supply-Chain Controls

Table 52: Software supply-chain controls

Control	Current state	Status
Dependency inventory and pinning	Single workspace lockfile; frozen installs in every container build; exact-version saving for new dependencies	IMPLEMENTED
Upstream source pinning	The media server is built from a pinned upstream tag verified against a fixed commit; patched WebRTC artefacts are pinned by SHA-256; every base and third-party container image is pinned by version and content digest, enforced by a release check	IMPLEMENTED
Cryptographic policy checks	Pre-commit checks reject non-AES-256 constructions, WebRTC dependency changes without a rebuilt patched artefact, console logging and native-project drift	IMPLEMENTED
Release manifest	Records git commit, creation time, a monotonic sequence number and an expiry date, every image archive with its SHA-256 and size, every SBOM with its SHA-256, every database migration with its class, the configuration bundle hash and, optionally, the minimum client app version per platform	IMPLEMENTED
Release signing	Canonical manifest signed with Ed25519 by a non-exportable key in OpenBao, on a self-hosted build runner. Hybrid Ed25519 and ML-DSA signatures are held back until an ML-DSA implementation has a published independent audit or an issued CMVP certificate (Chapter 39)	IMPLEMENTED
Freshness and rollback protection	Install and upgrade refuse an expired manifest and any manifest whose sequence number is below the floor the installation has recorded; two releases claiming the same sequence with different content are always refused. A running installation never stops because its manifest expires. Connected upgrades also require a short-lived signed timestamp statement from the release registry, under a key separate from the release key, and refuse one that is missing, expired, names a different manifest or is older than the newest already seen, so an installation cannot be silently held on an old release. This is signed expiry, sequence and timestamp protection, not the full TUF framework	IMPLEMENTED
Artefact inventory and SBOM	Release package includes an inventory of services, images, hashes and source commit, and a package-level CycloneDX SBOM for every image, generated offline by a pinned tool at release time, listed with its SHA-256 in the signed manifest, shipped in the offline package and verified when it is installed or upgraded; the release registry also keeps each SBOM with the release, accepted at publication only when it matches the signed manifest, and delivers it to connected installations, which verify it against the signed manifest in the same way ⁶²	IMPLEMENTED
Image integrity	Every image archive is bound by SHA-256 into the Ed25519-signed manifest, so one signature covers the whole release	BY DESIGN

⁶²OWASP Foundation, "CycloneDX Bill of Materials Standard" (ECMA-424); and The Linux Foundation, "SPDX Specification" (ISO/IEC 5962:2021).

Control	Current state	Status
Vulnerability scanning	Every release image is scanned with a pinned scanner and an offline vulnerability database before it is signed or published; a blocking finding stops the release. A container benchmark workflow is also available	IMPLEMENTED
Static analysis and secret scanning	Strict TypeScript, linting and policy checks; repository secret scanning on every change and over the full history weekly, and static analysis with project rules, both run offline on the project's own build runner from digest-pinned tools (a developer pre-commit check is also available)	IMPLEMENTED
Android release hardening	Android release builds are minified and obfuscated, ship no test libraries and expose only required components	IMPLEMENTED
Desktop code signing	macOS hardened runtime and notarisation; Windows Authenticode with update signatures pinned to the publisher	IMPLEMENTED
Third-party analytics in builds	None in any product application; dependency lockfile contains no analytics or crash-reporting SDK	IMPLEMENTED
External review of the release chain	Planned independent review	CERTIFICATION TARGET

The approach aligns with the practices of the NIST Secure Software Development Framework and the SLSA provenance model in intent;⁶³⁶⁴ SANKET does not claim a formal SLSA level.

30.2 Release Contents

An offline release package is a single archive with a SHA-256 sidecar, containing the signed manifest, image archives for SANKET's own services, the base and release Compose definitions, the hardening overlay, SQL migrations, edge, secrets-service and media-server configuration, deployment scripts and the artefact inventory. It excludes environment files, certificates, keys, secrets and licences. Third-party base images (database, cache, object store, edge, secrets service, push relay) are not bundled and must be pre-staged by the customer from a verified source.

⁶³M. Souppaya, K. Scarfone, D. Dodson, "Secure Software Development Framework (SSDF) Version 1.1", NIST SP 800-218, February 2022.

⁶⁴OpenSSF, "Supply-chain Levels for Software Artifacts (SLSA)" v1.0, slsa.dev.

30.3 Reference Update Process

Offline Software Update Architecture

Implemented steps (solid) and customer-procedure steps (dashed)

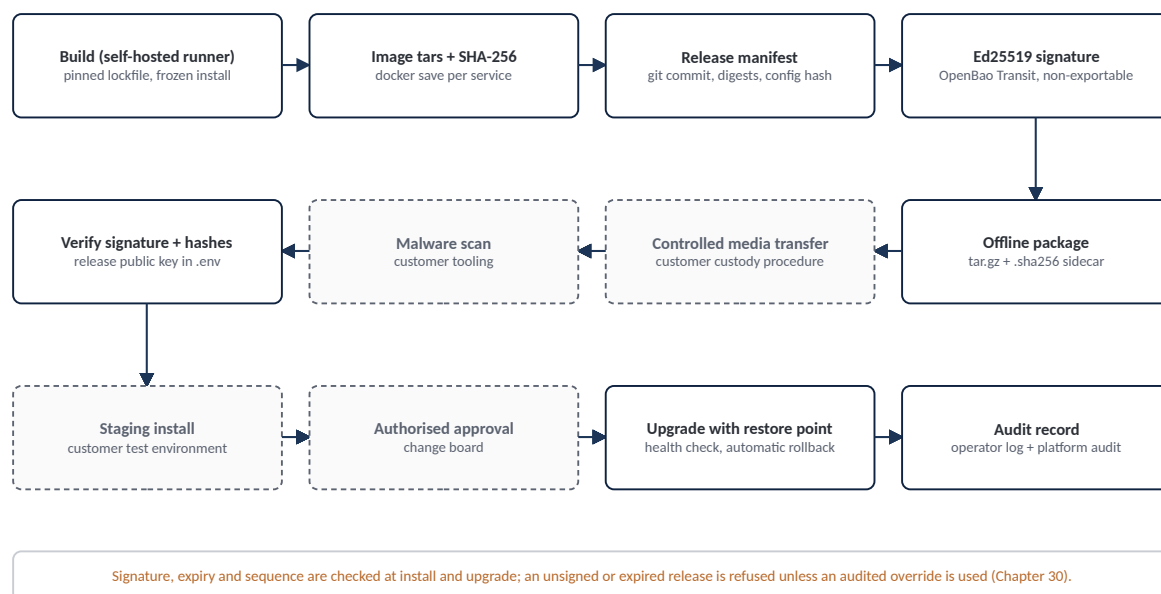


Figure 16: Offline software update architecture

- 1. Build.** Images are built from the pinned lockfile on a self-hosted runner and saved as archives.
- 2. Manifest and signature.** The manifest of hashes is signed with the release key.
- 3. Package.** The offline package and its checksum are published to the customer through an agreed channel.
- 4. Transfer.** The customer moves the package across the air gap on controlled media under its custody procedure.
- 5. Scan.** The customer scans the package with its own tools.
- 6. Verify.** The verification tool recomputes the manifest digest, every image archive hash and size, and the configuration bundle hash, and verifies the Ed25519 signature with the release public key configured in the installation. It refuses a manifest that has expired or whose sequence number is below the installation's recorded floor. The compose file that names the images to run is never taken from the package: the verification tool writes it from the verified manifest. Each archive is checked at the exact path it is then loaded from, only the archives the manifest lists are loaded, and a connected upgrade refuses an artifact list that differs from the signed manifest. The manifest may also state, per platform (iOS, Android and desktop), the lowest client app version the release needs; that statement is covered by the same signature, its shape is checked, and it is reported only from a manifest whose signature was verified.
- 7. Stage.** The customer installs the release in a staging environment and tests it.
- 8. Approve.** The customer's change authority approves deployment.
- 9. Upgrade.** The upgrade tool verifies again, loads images, snapshots the current state, switches versions, runs a health check and rolls back automatically on failure. Images are verified and loaded, and every image the new version needs is checked to be present, before any service is interrupted: a release that cannot be staged fails with the running version still serving. The manifest also lists every database migration with its class (expand: the previous version keeps working; contract: it may not) under the same signature. In the high-availability profile the upgrade runs host by host with health gates between steps, and every host performs the same signature, expiry, sequence and downgrade checks itself: a host given an unsigned or older package refuses it even when the others accept.

10. Record. The upgrade is recorded; post-upgrade gates check that running images match the manifest.

RECOMMENDED PRACTICE - CONFIGURE THE RELEASE PUBLIC KEY

Signature verification uses the release public key configured in the installation environment, obtained from Tosh Defence through an independent channel. Installation and upgrade refuse a release whose signature cannot be verified, whether the key is missing or the signature does not match, and so do the connected upgrade from the administrator console and the server's own start-up check.

30.4 Independent Verification by an Air-Gapped Customer

The verification tool uses only built-in platform cryptography and can be run on an isolated machine. A customer can therefore: obtain the release public key once, out of band; recompute every hash in the package; verify the signature; compare the manifest's source commit with any source-escrow arrangement; and confirm after installation that running images match the manifest.

30.5 Rollback, Pinning and Emergency Patches

- **Version pinning:** every base and third-party image is pinned by version and content digest in the container builds and the deployed configuration, and a release check refuses a reference pinned by tag alone. SANKET's own images are pinned by the SHA-256 of their signed release archives, and post-install gates check the running images against the signed manifest.
- **Rollback:** automatic. An upgrade is complete only after the new version is healthy, has applied its database migrations and answers smoke checks; on any failure it restores the previous images, configuration and, where the switch requires it, databases from its encrypted restore point, then checks the previous version the same way. Restore points are integrity-checked and bound to their installation before use. Every upgrade and rollback is recorded in the release audit log.
- **Release selection:** which signed releases may be installed is governed by the customer's change management. An expired release is refused, so an old release cannot be presented as current indefinitely, and downgrades are refused by the signed sequence number as well as the release version unless explicitly authorised and recorded in the release audit log. A server also refuses a signed licence that requires a newer backend release. An organisation can therefore still roll back deliberately to an earlier signed release when operationally required.
- **Emergency patches** follow the same signed path; a configuration-only release can update the configuration bundle without new images.
- **Connected upgrades** (optional) fetch packages and their SBOMs with short-lived, signed download authorisations, check the registry's signed timestamp statement, verify the package and run the same process. Upgrade execution from the console is off by default.

31 Offline Licensing

A sovereign platform must not stop working because a vendor server is unreachable. SANKET licences are signed files verified entirely inside the installation.

31.1 Licence Format

A licence is a canonical, sorted-key JSON document and an Ed25519 signature, each base64url-encoded.⁶⁵ It is signed by Tosh Defence with a non-exportable key and verified by the installation against a trust bundle that lists active, previous and revoked licence keys and is itself signed by a root key. Licence and trust-bundle signatures are Ed25519; hybrid Ed25519 and ML-DSA signatures are on the roadmap, held until an ML-DSA implementation has a published independent audit or an issued CMVP certificate (Chapter 39).

Table 53: What a licence binds

Field group	Contents
Identity	Licence identifier, signing key identifier, environment, tenant identifier, licensee organisation and country
Term	Plan, issue and expiry dates, update-subscription end, grace period, offline renewal counter
Limits	Users, administrators, storage, concurrent calls, group-call size, groups, audit retention, rate limits
Features	Entitlement flags such as voice, video, group calls, file sharing, encrypted vault, broadcast, integration API, audit export, remote wipe and administrator security keys
Clients	An optional minimum app version, applied as a floor on every platform
Security	Permitted deployment type (trial, standard, air-gapped) and an optional list of permitted image digests
Communication	Connected or offline update mode and the status-report destination

31.2 Validation

- Signature verified against the trust bundle; a bundle cannot supply its own trust anchor.
- Environment must match: a production installation rejects non-production licences.
- Tenant identifier must match the installation.
- If a permitted-image list is present, the running image digest must be on it.
- A licence may state a minimum app version. It is a floor on every platform (iOS, Android and desktop): the server raises the minimum it sends each app to at least that version, and no tenant setting can lower it.
- A production installation will not start without a valid licence.
- Renewals for disconnected sites are delivered as signed offline bundles, bound to the tenant, with a counter that must increase.
- Clock rollback is detected and enforced against.

⁶⁵S. Josefsson, I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", IETF RFC 8032, January 2017.

31.3Expiry Behaviour

Table 54: Licence expiry stages

Stage	Behaviour
Valid	Full operation
Grace period	Operation continues; new registrations blocked
Read-only (30 days after grace)	Existing data accessible; no new messages or calls
Hard lock	Service refuses use until renewed

31.4Design Notes

- Licences bind to the tenant identity and, optionally, to permitted image digests rather than to hardware, so that installations can be migrated, restored or rebuilt on new hardware without a licence reissue.
- Deployment attestation evidence (Secure Boot state and measured-boot values) is reported for operational visibility.
- Independent review of the licensing and release chain is a certification target (Chapter 40).

32 Availability, High Availability and Disaster Recovery

Communications matter most when other systems are under stress. This chapter covers denial-of-service resistance, the resilience of the single-host profile, the multi-host high-availability profile with second-site recovery, behaviour when an identity provider is unavailable, and backup and recovery.

32.1 Denial of Service and Resource Exhaustion

Table 55: Availability controls

Threat	Controls
External request floods	Edge request-rate limits; Redis-backed global limit per user and device or address; adaptive tightening on error spikes; block manager; customer upstream filtering
Authentication abuse	Compound per-address and per-account limits; lockout; integration failure throttles
Expensive operations	Fan-out caps, file size and duration limits, prekey fetch limits, call rate limits, broadcast caps, metadata size limit
Internal abuse by a user	Per-user message send rate; typing throttle; group size limit
Queue growth	Tombstoning and retention jobs; Redis memory cap set per installation; state keys (sessions, single-use tokens, restore challenges, call rooms) held on a separate instance that never evicts, so cache pressure cannot remove them
Resource exhaustion on the host	Per-service CPU and memory limits in the Edge profile; log rotation
Degraded operation	Administrator-triggered protective degraded mode; maintenance mode
Media floods	Room tokens required; one call per device; participant limits; TURN allocations only with media-server-issued credentials; per-address connection limit on TCP 443
Analytics load	Analytics never sits on a user path: bounded client queues and measured budgets; if the processor stops, chat and calls are unaffected and its lag is alerted

A single-host installation cannot absorb volumetric attacks larger than the customer's network capacity. Deployments on hostile networks should front the edge with the customer's own DDoS protection, or run on private networks.

32.2 Resilience of the Single-Host Profile

The Edge profile is a single host by design. Services restart automatically and are health-checked; clients queue outbound messages, attachments and voice notes in an encrypted outbox and reconnect; the server holds ciphertext for offline devices. Redis runs as two instances on the host as well: an evictable cache and a persistent state instance that never evicts, with the memory cap set per installation. There is no in-profile redundancy: a host failure interrupts service until the host is repaired or the installation is restored on a standby host. Customers that need redundancy use the high-availability profile below. Tosh Defence does not publish an availability SLA figure for self-hosted installations, because availability depends on the customer's infrastructure and operations.

32.3High-Availability Reference Topology

Status: IMPLEMENTED. The high-availability profile runs on multi-host Docker Compose (Kubernetes packaging is on the roadmap). Its reference topology comprises:

- two edge hosts behind a shared address or the customer's load balancer, each with the TLS certificates and the device CA and revocation list, with the client address preserved end to end;
- N+1 API and WebSocket nodes, with real-time delivery fanned out across nodes through Redis publish and subscribe, and exactly one elected runner for each singleton job, so no scheduled job runs twice;
- Redis split into an evictable cache and a persistent state instance that never evicts, with Sentinel failover;
- PostgreSQL streaming replication with fenced failover, so a former primary can never accept writes, and write-ahead-log archiving to the second site, for both the main and the threat-processor databases;
- distributed MinIO in each site with site replication between sites, keeping AES-256-GCM encryption at rest;
- a three-node OpenBao raft cluster with TLS between nodes, a per-node unseal ceremony under the same M-of-N custody, and encrypted snapshots shipped to the second site;
- several media-server nodes routed through the state Redis, each with the same AES-256 SRTP policy;
- the push relay on more than one host;
- a maintenance mode for planned work.

Host roles and every peer address are set in the installation's environment file and role overlays, so the files shipped in each release stay identical for every customer. Installation, upgrade, backup and restore are orchestrated across the hosts with health gates between steps, and every host performs the release signature, expiry, sequence and downgrade checks itself (Chapter 30). Backups are taken from a replica.

32.4Behaviour When an Identity Provider Is Unavailable

Federated sign-in is optional and only ever uses the customer's own identity provider (Chapter 13). Its outage is contained by design:

- **Administrators.** At least one local administrator, signing in with a password and a second factor, always remains: the installation refuses any configuration that would leave none, and federation never disables local sign-in. When the provider cannot be reached, the console directs administrators to a local account.
- **End users.** The directory is consulted only at enrolment, device restore and new-device sign-in. Everyday sign-in on an existing device and all messaging and calls never contact it, so field devices are unaffected by an outage; enrolment and restore wait until the directory is back, and any exception is an audited owner decision.
- **Monitoring.** The administrator monitors show the result of the latest provider check.

32.5Backup

Table 56: Released backup capability

Aspect	Released behaviour
Scope	Main and threat-processor database dumps (taken from a replica in the high-availability profile); attachment bucket; OpenBao data and unseal material; installation configuration
Not included	Vault bucket and Redis (short-lived state)

Aspect	Released behaviour
Encryption	Each component is a separate artefact encrypted with its own AES-256-GCM key. The recovery artefact (OpenBao data, unseal material, configuration) has its key wrapped under a key derived from an operator passphrase with script, because OpenBao itself is inside it; every other artefact has its key wrapped by the installation's OpenBao, so it cannot be read without that OpenBao
Integrity	A manifest listing every artefact with its size and SHA-256 is signed with a non-exportable Ed25519 key in the installation's OpenBao. The restore tool opens the recovery artefact with the passphrase, checks that the signing key sealed there is the one the manifest names, verifies the signature and every artefact, and refuses before changing anything if any of them differs
Retention	14 days with a minimum of seven archives, pruned only after a successful run
Restore	Restore script with post-restore health check. The data keys are unwrapped by the restored OpenBao only after it is unsealed by its key custodians, and any recovery credential the custodians issue is short-lived and restore-scoped, so no long-lived token has to be kept for recovery
Scheduling and offline copies	Operator-scheduled; alternating offline media recommended in the operations runbook
Upgrade restore point	Each upgrade takes a protected restore point of the running service images and configuration before it switches anything and, where the switch changes the database schema, of the databases too, each dump encrypted with AES-256-GCM under its own key wrapped by the installation's OpenBao and never written in clear
Upgrade switch	An upgrade whose database changes the previous version can still run with switches without maintenance mode: the new API starts beside the old one, which stops only once the new one is healthy, so requests continue to be answered and chat connections reconnect within seconds; a new API that does not become healthy is removed and the old one keeps serving. Other releases switch inside a planned maintenance window. The other services the release changes are restarted after the API, and each is awaited until healthy
Scheduled database backups	Backup policies run by the platform scheduler on their own schedule: each database dump is encrypted with its own AES-256-GCM key, that key is wrapped by the installation's OpenBao (non-exportable), and the run's manifest is signed with an installation Ed25519 key and stored next to every copy. A restore verifies the signature, then the checksum, before it uses a copy, and refuses one signed by any other key
Backup destinations	Local or network-attached storage, S3-compatible object storage, or SFTP, with an optional second copy; no foreign cloud destination exists. A policy aimed at anything else is paused and raises an administrator alert

RECOMMENDED PRACTICE - PROTECT THE BACKUP PASSPHRASE AND THE BACKUP KEYS

The full-installation archive contains the installation's own secrets, so its passphrase protects the whole installation: keep the passphrase in separate custody from the media. Scheduled backups need no passphrase, but they can be decrypted and verified only by the installation's own OpenBao, so the OpenBao unseal material and its recovery procedure are what must be held in separate custody. Sources other than the databases (attachment bucket, OpenBao data, configuration) are covered by the full-installation archive.

32.6 Disaster Recovery Methodology

The high-availability reference topology, with an asynchronous second site, is designed for a recovery point objective (RPO) of 5 minutes and a recovery time objective (RTO) of 1 hour. These are design targets, not measured claims: each deployment's disaster-recovery drill records the figures it actually achieved, and those recorded figures, not the targets, are what an installation can rely on. An RPO of zero with an RTO of 15 minutes would need synchronous replication and a third site or witness; it is available as a separately priced option and is not part of the built profile.

The drill tooling writes markers at the primary site, fences it, promotes the second site, switches the service names and later fails back. The achieved RPO is measured as the gap between the last marker acknowledged at the primary site and the last marker present at the second site; the achieved RTO runs from the declaration of the disaster to the first successful message and call on iOS, Android and desktop.

For single-host installations, and for any installation's backup-based recovery, RPO and RTO are set by the customer's backup frequency, media handling and standby capacity. The recommended methodology:

1. **Classify** the installation's criticality and set target RPO (maximum tolerable data loss) and RTO (maximum tolerable outage).⁶⁶
2. **Schedule** backups at an interval no longer than the RPO, and replicate encrypted archives to a recovery site or offline media.
3. **Prepare** a standby host with the same release and pre-staged images.
4. **Separate custody** of the backup passphrase, OpenBao unseal shares and CA keys.
5. **Test** restoration and, in the high-availability profile, site failover on a schedule, measuring the achieved recovery point and recovery time, and record each test.
6. **Recover keys:** installation keys come back with the backup; end-user message keys are not recoverable by design, and users re-establish sessions after restoration.
7. **Recover PKI:** re-issue the edge certificate from the customer CA if the original key is unavailable.

Customers should validate restoration on their own infrastructure as part of acceptance and at a regular interval thereafter.

⁶⁶M. Swanson et al., "Contingency Planning Guide for Federal Information Systems", NIST SP 800-34 Rev. 1, May 2010.

33 Hardware Security

SANKET does not require special hardware, but it benefits from, and is designed to be deployed with, hardware security features on servers and endpoints.

Table 57: Hardware security options

Feature	Use with SANKET	Status
Mobile secure storage (Keychain, Keystore)	Protects protocol state, device keys and refresh tokens on phones, this device only	IMPLEMENTED
Secure Enclave (iOS)	A non-exportable P-256 key in the Secure Enclave; a 256-bit AES-GCM key derived from it (ECDH and HKDF-SHA256) wraps the libsignal store, device secrets, the device certificate key and the account list key. Usable after first unlock, so background delivery and calls work while locked	IMPLEMENTED
StrongBox, with TEE fallback (Android)	A StrongBox key where the phone has a StrongBox secure element; a TEE-backed Android Keystore key otherwise; wraps the same items with AES-256-GCM	IMPLEMENTED
Protection level per device	Each phone reports Secure Enclave, StrongBox, TEE or software (simulators and emulators); stored per device, shown in the console; policy can require a minimum level	IMPLEMENTED
Desktop OS keychain	Protects desktop protocol state and secrets	IMPLEMENTED
TPM 2.0 on servers	Full-disk encryption key protection; measured boot	RECOMMENDED
Secure Boot on servers	Boot-chain integrity of the host	RECOMMENDED
Encrypted server storage	Full-disk encryption (for example LUKS with TPM) for all installation volumes	RECOMMENDED
HSM	Root CA keys; evaluation of OpenBao auto-unseal against customer key infrastructure	RECOMMENDED
FIDO2 security keys	Phishing-resistant administrator sign-in (WebAuthn, ES256 or EdDSA, user verification required), verified by the installation; required on new installations for every administrator	IMPLEMENTED
Redundant infrastructure	Redundant power, storage and network for the host	RECOMMENDED
Android Key Attestation	Proof that a phone's keys are held in StrongBox or the TEE, verified offline against vendor roots and a revocation snapshot shipped with each release	IMPLEMENTED

34 Security Control Matrix

The matrix maps each security objective to the threats it addresses, the controls that implement it, where those controls operate, the relevant standards, and the residual risk.⁶⁷

Table 58: Security control matrix

Security objective	Threat	Control	Security layer	Relevant standard	Residual risk
Confidentiality (content)	Server, network or storage compromise	libsignal E2EE; per-file AES-256-GCM; call frame encryption	Endpoint	Signal PQXDH, Double Ratchet; NIST SP 800-38D	Endpoint compromise
Confidentiality (transport)	Network eavesdropping	TLS 1.3 / 1.2 AES-256-GCM only; hybrid post-quantum key exchange offered first; HSTS; certificate pinning on desktop and mobile; device mutual TLS	Network	RFC 8446, RFC 9325	Device trust store integrity
Integrity	Tampering with messages or files	libsignal MACs; GCM tags; TLS	Endpoint, network	RFC 2104, SP 800-38D	Server-controlled ordering and dropping
Authenticity	Impersonation of users or devices	Identity keys pinned on first use; signed prekeys; signed device links; device-bound tokens; device client certificates	Endpoint, application	XEdDSA; RFC 7519	Social engineering of device approvers
Device-list integrity	A compromised or hostile server adding a device to an account to receive copies	Device lists signed with each account's list key on its own devices; append-only log refusing forks; senders verify the signed list before encrypting on every fan-out path; identity-change notices replayed to offline devices	Endpoint	Key transparency (self-hosted)	Trust on first use for a new identity, verified by safety-number comparison
Availability	Denial of service	Rate limits, adaptive limits, caps, block manager, health checks	Edge, application	Customer DDoS controls	Volumetric attacks beyond the customer's DDoS controls
Accountability	Undetected misuse	Hash-chained audit, signed with a non-exportable key; SIEM export with the anchors	Application	NIST SP 800-92	Mitigated by independent SIEM copy
Non-repudiation	Denial of administrative actions	Audit records bound to authenticated actors	Application	-	Not cryptographic non-repudiation; messages are deniable by protocol design
Forward secrecy	Later key compromise	Double Ratchet; ECDHE TLS; call key ratchet	Endpoint, network	Signal specifications	Stored local plaintext
Post-compromise recovery	Ongoing use of stolen state	DH ratchet; session reset; call re-keying	Endpoint	Signal specifications	Persistent endpoint compromise
Access control	Unauthorised access	RBAC; group roles; membership checks; policy; licence gates	Application	NIST SP 800-53 AC family	Misconfiguration by administrators

⁶⁷NIST, "Security and Privacy Controls for Information Systems and Organizations", NIST SP 800-53 Rev. 5, September 2020.

Security objective	Threat	Control	Security layer	Relevant standard	Residual risk
Sovereignty	Foreign jurisdiction and dependency	Self-hosting; no third-party SDKs; offline licence and updates	Deployment	-	Platform push for iOS in connected mode
Device trust	Rogue or lost devices	Device states, caps, approval, lost mode, wipe; hardware-backed keys with a recorded protection level; offline Android Key Attestation	Application, endpoint	OWASP MASVS	Posture on unmanaged devices
Auditability	Tampering with records	Hash chain with verifier; periodic Ed25519 anchors from the key service; exported hash links and anchors	Application	NIST SP 800-92	Collusion across separated operator duties; mitigated by independent SIEM copy
Resilience	Host failure, data loss	Encrypted backups; restore tooling; health checks; multi-host high-availability profile with an asynchronous second site and drill tooling	Operations	NIST SP 800-34	Writes since the last replicated point on a site loss (design target RPO 5 minutes); recovery time of a single-host profile
Supply-chain integrity	Malicious or tampered release	Signed manifest with sequence number and expiry; signed timestamp on connected upgrades; hashes; pinned sources; offline verification	Build, deployment	NIST SP 800-218; SLSA (intent)	Compromise upstream of the build
Administrative control	Abuse of administrator power	FIDO2 security keys for administrators; federation only to the customer's identity provider; least privilege; audit; no content keys	Application	NIST SP 800-53 AC-5, AC-6	Host-level administrators (separation of duties)

35 Defensive Attack Scenarios

The following scenarios walk through realistic attacks from the defender's perspective: how each is prevented, detected, contained and recovered from, and what risk remains. They describe defensive behaviour only.

35.1 Scenario 1: Untrusted public network interception

Stage	Description
Attack	An adversary controls the Wi-Fi or carrier network and records or manipulates traffic (ATT&CK: Adversary-in-the-Middle).
Prevention	TLS with AES-256-GCM and hybrid post-quantum key exchange where the client supports it; certificate validation against the customer CA; certificate pinning on desktop and mobile; device client certificates; E2EE inside TLS.
Detection	Certificate errors on clients; pin mismatch refusals.
Containment	Move users to trusted networks or VPN; revoke sessions if tokens may have been captured.
Recovery	Rotate token secrets if interception of tokens is suspected.
Residual risk	Metadata visible to the network; content remains end-to-end encrypted, with post-quantum protection from PQXDH and SPQR.

35.2 Scenario 2: Compromised SANKET application server

Stage	Description
Attack	An attacker gains code execution in the API container.
Prevention	Content keys absent from the server; signed device lists, so the server cannot add a device that receives copies; non-root containers; least-privilege access to secrets.
Detection	Unexpected processes or traffic (customer host monitoring); audit anomalies; clients reporting decryption failures or unexpected devices.
Containment	Isolate the host; stop the installation if necessary.
Recovery	Rebuild from a verified release; rotate all server secrets; force re-authentication; review device inventories.
Residual risk	Metadata and notice broadcasts observed until the server is isolated; content stays end-to-end encrypted.

35.3 Scenario 3: Stolen employee device

Stage	Description
Attack	A phone is stolen while locked.
Prevention	OS encryption; app passcode or biometric; SQLCipher; keys wrapped by a hardware key (Secure Enclave, StrongBox or TEE); wipe after failed unlocks.
Detection	User reports loss.
Containment	Mark lost: the device is refused by the server and receives no new messages.
Recovery	Remote wipe after confirmation; issue a replacement device; rotate identity if appropriate.

Stage	Description
Residual risk	Forensic attacks on the OS if the device was unlocked or vulnerable.

35.4 Scenario 4: Malicious system administrator

Stage	Description
Attack	An administrator tries to read communications or cover activity.
Prevention	No content keys on server; RBAC; administrator security keys; signed device lists; audit chain; separation of duties.
Detection	Audit review; SIEM copy; chain verification; identity changes shown to every peer.
Containment	Revoke the administrator; preserve evidence.
Recovery	Review and reverse changes; review device approvals made by that administrator.
Residual risk	Access to metadata within the administrator's role.

35.5 Scenario 5: Replay of captured encrypted traffic

Stage	Description
Attack	An attacker resubmits captured API requests or messages.
Prevention	TLS; libsignal duplicate detection; single-use tokens; idempotency keys; webhook timestamps.
Detection	Duplicate-message errors; refresh-token reuse events.
Containment	Device revoked automatically on refresh reuse.
Recovery	User re-authenticates.
Residual risk	Minimal.

35.6 Scenario 6: Compromised user password

Stage	Description
Attack	An attacker obtains a user's password.
Prevention	TOTP required; lockout; rate limits; device caps and approval.
Detection	Failed TOTP events; new-device events; lockouts.
Containment	Reset password; suspend account if needed.
Recovery	Re-enrol TOTP if the second factor may be exposed.
Residual risk	Social engineering of the user; administrators use phishing-resistant security keys.

35.7 Scenario 7: Unauthorised device access attempt

Stage	Description
Attack	An attacker tries to add a device to an account.
Prevention	Device caps without eviction; administrator approval mode; trusted-phone linking with TOTP and signature; a device absent from the account's signed device list receives no messages.
Detection	Pending-device notifications; device-approved events.

Stage	Description
Containment	Reject or revoke the device.
Recovery	Review account security; rotate credentials.
Residual risk	Social engineering of the user or an approver.

35.8 Scenario 8: Database theft

Stage	Description
Attack	A database copy is exfiltrated.
Prevention	E2EE content; Argon2id; encrypted secret fields; tombstoning.
Detection	Data-loss detection (customer).
Containment	Rotate field and TOTP keys if keys may also be exposed.
Recovery	Assess metadata disclosure; consider password resets.
Residual risk	Account and communication metadata in the copy.

35.9 Scenario 9: Backup media theft

Stage	Description
Attack	Backup media is lost or stolen.
Prevention	Archive encryption with a passphrase-derived key; E2EE content.
Detection	Custody records.
Containment	Treat as potential exposure of installation secrets.
Recovery	Rotate secrets and unseal material if the passphrase may be weak or exposed.
Residual risk	Offline guessing of a weak passphrase.

35.10 Scenario 10: Supply-chain attack on an update package

Stage	Description
Attack	A tampered package is introduced in transit.
Prevention	Signed manifest with sequence number and expiry; per-archive SHA-256; offline verification; signed timestamp on connected upgrades; customer staging.
Detection	Hash or signature mismatch; expired, replayed or lower-sequence manifest refused.
Containment	Reject the package.
Recovery	Obtain a fresh copy through an independent channel.
Residual risk	Compromise upstream of signing.

35.11 Scenario 11: Lost device with cached messages

Stage	Description
Attack	A device holding history is lost and may be unlocked later.

Stage	Description
Prevention	Local encryption; lock; disappearing messages where used.
Detection	Loss report.
Containment	Lost mode; remote wipe.
Recovery	Replacement device; history on lost device not recoverable by the organisation.
Residual risk	Content already viewed on an unlocked device.

35.12 Scenario 12: User removed from a confidential group

Stage	Description
Attack	A member must lose access immediately.
Prevention	Removal takes effect in the send transaction; future fan-outs exclude their devices; downloads require active membership.
Detection	Membership-change records.
Containment	Removal; device revocation if the person is leaving.
Recovery	Consider rotating any shared secrets discussed.
Residual risk	The removed user keeps whatever they already read or saved.

36 Deployment Mode Comparison

SANKET is deployed in three modes. They are not ranked: each suits a different combination of security requirement, connectivity and operating capacity. In every mode the installation is dedicated to one customer, and every mode can run either the single-host Edge profile or the multi-host high-availability profile, with an optional second site for disaster recovery (Chapter 32). The topology is chosen per installation in its environment file and role overlays; the files shipped in a release are the same for every customer.

Table 59: Deployment modes compared

Attribute	Sovereign hosted	Customer on-premise	Air-gapped
Infrastructure ownership	Dedicated installation in a sovereign data centre chosen by the customer, operated under contract	Customer's own servers and data centre	Customer's own servers in an isolated enclave
Topology	Single host, or multi-host high availability with an optional second site	Same	Same, with every host and the second site inside the enclave
Network dependency	Internet-reachable edge: one public TCP 443 (HTTPS and TURN) plus the media UDP port	Internet-reachable or private network, as the customer chooses	None outside the enclave; TURN off
PKI	ACME DNS-01 certificates or customer CA	Customer CA (recommended) or ACME	Customer internal CA
External dependencies	Optional Apple push; optional connected upgrades from Tosh Defence; no public STUN	Same, customer-controlled	None
Notifications	Android self-hosted push; iOS through Apple push	Same	Android self-hosted push; iOS without background wake-up
Updates	Offline package or optional connected upgrade, installed on approval	Same	Offline package on controlled media
Administration	Customer administrators; operator under contract for the host	Customer	Customer
Data storage	In the chosen sovereign data centre	On customer premises	Inside the enclave
Logging	Installation plus customer SIEM over TLS	Same	Internal SIEM
Identity	Administrator, invitation or SCIM provisioning; optional sign-in through the customer's own OIDC, SAML or LDAP directory	Same; SCIM and federation from the internal directory	Same; SCIM and federation from the enclave directory
Backup	Encrypted archives to customer-designated storage	Customer backup infrastructure	Offline media inside the enclave
Licence	Signed licence, verified locally	Same	Offline licence and offline renewal bundles

Attribute	Sovereign hosted	Customer on-premise	Air-gapped
Operational control	Shared between customer and contracted operator	Customer	Customer
Sovereignty level	High: jurisdiction and data stay sovereign; the operator is a party to trust	Very high	Highest available
Ideal use case	Organisations needing fast deployment without their own data centre operations	Organisations with their own infrastructure and security operations	Classified-adjacent, defence and critical environments where any external path is unacceptable

37 Feature Catalogue

Each capability is described by what it does, how it works, its security value, the options available to administrators, and its boundaries.

Table 60: Feature catalogue

Feature	What it does	How it works	Security value	Configuration options	Boundaries
Secure direct communication	One-to-one messaging between users	libsignal PQXDH and Triple Ratchet, one ciphertext per recipient device, content in a versioned envelope	Content readable only by the participants' devices	Read receipts, typing, edit, delete-for-everyone, disappearing messages	Replies, reactions, locations, contacts, link previews and forwards inside the ciphertext; sender, recipient devices, time and size band visible to the server (Chapter 20)
Secure group communication	Multi-member conversations	Pairwise libsignal fan-out to every member device	No group key to steal; removal is immediate for future messages	Creation permission, member approval (default on), size limit (default 1,024), renaming, leaving	Cost scales with members times devices
Encrypted priority broadcast	Priority messages from a dedicated broadcast-sender account to all users, a role or a custom list	Sent from the account's approved desktop, which encrypts for every recipient device with per-device libsignal sessions in a resumable job; four priority levels (normal, important, urgent, critical) with full-screen flash display on mobile; priority, deadline, sent time and expiry also inside the encrypted envelope; attachments encrypted once with a per-file key carried in each copy; recipients pin the sending device and identity	Server cannot read, downgrade, move or replay a broadcast; lock-screen text generic by priority	Audience, priority, mandatory expiry, acknowledgement deadline (1 minute to 7 days, default 30 minutes, counted from the end of the fan-out); sender accounts created, assigned to named administrators and suspended in the console	Immediate only (no scheduling)
Notice broadcast	Non-sensitive administrator announcements	Sent from the administration console; scheduled notices are sent by the platform scheduler within a minute of their time, under the same licence and rate checks as an immediate send, and never once expired, a day overdue or after their creator has left	Simple organisational notification	Audience, priority, schedule, acknowledgement deadline, retention (90 days by default)	Server-readable by design; labelled "Notice (not end-to-end encrypted)" to sender and recipients
Acknowledgement tracking	Shows who has received and acknowledged a broadcast	Per-recipient delivered and acknowledged records kept as metadata; an urgent or critical broadcast still unacknowledged at its deadline is re-notified once to each recipient who has not acknowledged and raises an administrator alert with the counts	Confirms receipt of critical notices	Deadline per broadcast	Recipients must open the app

Feature	What it does	How it works	Security value	Configuration options	Boundaries
Voice and video calls	One-to-one and group calls	Self-hosted SFU; AES-256-GCM frame encryption plus AES-256-GCM SRTP; a server-issued id per call binds its frame keys (Chapter 19)	Media not readable by the server	Voice, video and group calls toggles; duration and participant limits; per-user call rates; TURN relay	UDP media preferred; TCP 443 with TURN over TLS where UDP is blocked
Encrypted file exchange	Documents, images, video, voice notes, contacts and locations	Per-file AES-256-GCM, key inside the libsignal message	Files opaque to the server	Per-source enablement, size and type limits, download-to-device (off by default), export	Recipients control copies they have legitimately obtained
Local secure vault	Per-device index of received attachments	Encrypted under the local cache key; access logged locally	Organised access to files without a server repository	None	Per device; not synchronised
Encrypted identity backup	Optional account recovery	PBKDF2 and AES-256-GCM vault with password and recovery code	Recovery after device loss	Enable or disable by policy and licence	Identity key leaves the device in encrypted form
Message expiry	Disappearing messages	Server deletion at expiry plus device purge	Limits retained history	Allow, force, tenant default duration (30 seconds, 5 minutes, 1 hour, 8 hours, 24 hours, 7 days or 30 days), which is a ceiling when forced (the shorter of it and the user duration applies); user durations of 1 hour, 6 hours, 1 day, 7, 30, 45 or 90 days; one duration table shared by server, mobile and desktop, so every option sets the expiry it names	Recipients can capture content before expiry
File expiry	Lifecycle expiry of stored objects	Object-storage lifecycle rule (90 days)	Limits server-side retention	Retention windows	Does not remove recipients' copies
Delivery and read acknowledgements	Receipts	Server-side state	Operational assurance	Tenant and user settings	Metadata
Offline queue	Messages composed offline	Encrypted local outbox (7 days); idempotent retries with the same retry classification on mobile and desktop	No loss or duplication	None	Text messages, attachments and voice notes
Forwarding controls	Restricts forwarding	Client-side policy (forwarding off by default)	Reduces casual redistribution	Policy	Client-enforced only
Screenshot restrictions	Blocks or detects screen capture	Android secure flag and iOS protected rendering; in-chat screenshot alerts; desktop content protection	Deters capture	Configurable; detection policy	Cannot stop an external camera
Watermarking	Visible identity overlay	Desktop and mobile file viewer overlay of Sanket ID or name	Deterrence and attribution	Configurable; content choice	Visible only; forensic watermarking on the roadmap

Feature	What it does	How it works	Security value	Configuration options	Boundaries
Classification labels	Marks messages and attachments with a customer-defined label	Ranked list of up to eight labels and a default; the label travels inside the end-to-end encrypted envelope; content above the lowest rank cannot be forwarded, exported or downloaded on mobile or desktop, including the desktop secure viewer; unknown labels treated as the highest rank; a forward never lowers a label	Handling rules the server cannot read or alter	Label names and ranks; default label; copy governed by the clipboard policy	Client-enforced; conversation-level labels on the roadmap; installation separation between accredited levels (Chapter 38)
Device governance	Approve, revoke, lost mode, locate, remote wipe, duress	Server-enforced device states; confirmed wipe; device client certificates; hardware-backed keys with a recorded protection level; offline Android Key Attestation	Containment of lost or seized devices	Policy and licence gates; required protection level; device mutual TLS enforcement	Offline devices act on revocation when they reconnect
Signed device lists	Lets senders verify every account's devices	Each account's device list signed with its list key on the primary device; append-only log on the installation; senders verify before encrypting on every fan-out path	A hostile server cannot add a device that receives copies	None	Trust on first use for a new identity, verified by safety-number comparison
MFA	Second factor: TOTP for users; FIDO2 security keys for administrators	RFC 6238 with SHA-256; WebAuthn (ES256 or EdDSA, user verification required, synced passkeys refused by default)	Credential theft resistance; phishing resistance for administrators	Second-factor enforcement configurable per installation; security keys required for administrators by default	User TOTP is not phishing-resistant
Federated sign-in	Administrator sign-in through the customer's identity provider; directory check for users at enrolment	OIDC with PKCE or SP-initiated SAML 2.0 for the console; LDAP over TLS at enrolment, restore and new-device sign-in	One identity source under the customer's control	Provider settings, group-to-role map (never owner)	First factor only; public providers refused
Session management	Device-bound sessions	Short access tokens; single-use refresh; reuse detection; idle-device revocation	Theft and replay resistance	Token lifetimes (operator); idle-device periods	-
Administrator console	Users, devices, policy, broadcasts, monitors, audit, integrations	Permission-scoped roles	Governed administration	Roles	No dual control
Threat monitoring	Lockout, limits, blocks, alerts feed, optional processor	Rule-based	Detection of common attacks	Thresholds	SIEM correlation recommended
Audit and reporting	Hash-chained audit; CSV export; syslog; API; webhooks	Chapter 21	Accountability and SIEM integration	Export enablement	External anchoring through SIEM export
Directory provisioning	SCIM 2.0 Users; Integration API	Scoped credentials	Joiner-mover-leaver automation	Client scopes, expiry, CIDR allowlist	No just-in-time account creation from sign-in
Managed configuration (MDM)	Policy delivered by MDM	iOS Managed App Configuration; Android managed configurations; tighten-only	Fleet-level enforcement	Passcode, lock timeout, biometrics, wipe threshold, expected tenant	Requires an MDM

Feature	What it does	How it works	Security value	Configuration options	Boundaries
Usage analytics	First-party usage counts	A processor on the installation aggregates pseudonymous counts; no content; no third party	Operational insight without telemetry leaving the installation	On by default; organisation switch (step-up, audited); installation lock; delete all collected analytics	Audit logs and app-version records are not analytics and stay on

Roadmap capabilities include sealed sender, conversation-level classification labels, per-share file revocation, forensic watermarking and push-to-talk. Server-side call recording is deliberately excluded by design.

38 Information Classification Support

Many customers handle information under a classification scheme, whether national markings or customer-defined labels. This chapter states plainly what SANKET does and does not do in that respect.

NOTE - APPROACH IN THE CURRENT PLATFORM

SANKET supports classification at two levels. Within an installation, each message and each attachment can carry a label from a customer-defined ranked list of up to eight, carried inside the end-to-end encrypted message so that the server can neither read nor alter it, and the label drives what recipients can do with the content. Between formally accredited levels, the recommended separation remains one installation per security domain: each installation is a self-contained security domain, which is the strongest separation available and does not rely on software labels to keep levels apart.

38.1 Classification Labels

Table 61: Classification labels within an installation

Element	Behaviour	Status
Label list	Defined by the customer in the administrator console: up to eight labels, each with an identifier, a name and a rank from 0 to 7, and a default label preselected when composing. Every change is audited. Callers that have not signed in never receive the label names, since they describe the deployment	CONFIGURABLE
Where a label travels	Inside the end-to-end encrypted content envelope (Chapter 17), for the message and for each attachment. The server never sees a label, and clients no longer send a free-text classification field with file uploads	IMPLEMENTED
Handling	Content whose effective label ranks above the lowest configured rank cannot be forwarded, exported or downloaded (saved to the device or shared out of the app) on mobile or desktop, the desktop secure viewer included; the desktop main process enforces the download restriction as well as its interface. Content at the lowest rank, and unlabelled content, follows the installation's normal policy. Copying stays governed by the clipboard policy	IMPLEMENTED
Effective label	The stricter of the message label and the file label; for each, the stricter of the rank carried in the message and the rank currently configured for that label, so lowering a rank in the console never unlocks content that was sent at a higher one. A label the device does not know is treated as the highest rank (fail closed)	IMPLEMENTED
Forwarding	Where forwarding is allowed, a forward never carries a lower label than its source	IMPLEMENTED
Audit	Devices report coarse events (label applied, action blocked by a label) to the audit log under the user's Sanket ID and device, with no message identifier, conversation identifier or content (Chapter 21)	IMPLEMENTED
Parity	Mobile and desktop apply the same handling rule from the same shared policy code	IMPLEMENTED
Conversation-level labels and a per-label handling matrix	Labels attached to a whole conversation, and handling rules configured separately for each label (for example retention or audience by label), are not provided	ROADMAP

Labels are an aid to handling discipline among authorised users of one installation. They rely on the recipient's app honouring them; a recipient who can read content can still photograph the screen or retype it, which is why screenshot blocking, visible watermarks (Chapter 18) and managed devices complement them, and why labels do not replace separation between accredited levels.

38.2 Separation Between Security Domains

- **Separate installations per security domain.** Because each installation is fully independent, customers can run one installation per classification level or compartment, on networks accredited for that level. This is the strongest separation available and is how high-trust customers should segregate formally accredited levels.
- **Policy per installation.** Each installation's policy (download, export, forwarding, retention, screenshot blocking, disappearing messages) can be set to match the handling rules for the level it carries.
- **Membership-based access.** Within an installation, access to content follows conversation membership, with member approval and administrator-controlled group creation.

38.3 Policy Support Is Not Accreditation

Classification-aware handling does not by itself authorise a system to process government classified information. Formal accreditation of a system for a given classification level is a decision of the competent national authority, based on assessment of the whole system: platform, infrastructure, endpoints, personnel and procedures. SANKET is not accredited for any national classification level at the date of this document.

39 Crypto-Agility and Post-Quantum Readiness

Cryptographic algorithms age. A platform intended to protect information for years must be able to change its primitives without being rebuilt,⁶⁸ and must plan now for the arrival of cryptographically relevant quantum computers.

39.1 Crypto-Agility in the Current Architecture

Table 62: Where SANKET can change cryptography without redesign

Area	Agility mechanism
Message protocol	libsignal is a versioned library with versioned message and key formats; upgrading the library brings new primitives (as PQXDH and SPQR did) without changes to SANKET's protocol code
Call media	Call crypto is negotiated as a numbered profile; the server refuses devices and offers below the current profile, so the required profile can be raised across the whole platform in a single, server-enforced step
Transport	TLS suites and key-exchange groups are configuration at the edge and in a single outbound TLS policy module; the hybrid post-quantum group was added this way, with the suites unchanged
Symmetric primitives	All application AES-GCM goes through one strict helper, so a change of primitive or key size is made in one place and enforced by a pre-commit check
Signing	Licence and release verification use trust bundles with key identifiers, active, previous and revoked keys
Content envelope	Message content is a versioned envelope inside libsignal; a new version is switched on per tenant only when every platform's minimum app version can read it, and older clients ask to be updated instead of dropping the message
Server key custody	OpenBao Transit keys are versioned; key types can be added without application redesign
Certificates	Edge and SIEM certificates come from the customer PKI and can follow its algorithm profile

39.2 National Cryptographic Modules

Some customers require nationally approved algorithms or cryptographic modules. SANKET's non-message layers (transport, local storage, server-side secrets, signing, backups) are structured so that a provider or algorithm can be substituted in a defined set of places. The message layer is different: SANKET deliberately uses only Signal's audited libsignal implementation for end-to-end messaging, and libsignal fixes its primitives. Substituting a national cipher inside the Signal Protocol would mean departing from the audited implementation, which SANKET will not do as an unreviewed change. Any such requirement would be addressed as a separately specified, independently evaluated protocol profile, in agreement with the customer's cryptographic authority.

⁶⁸E. Barker, A. Roginsky, "Transitioning the Use of Cryptographic Algorithms and Key Lengths", NIST SP 800-131A Rev. 2, March 2019.

39.3Post-Quantum Readiness

39.3.1 Why it matters now

An adversary can record encrypted traffic today and decrypt it later if a sufficiently capable quantum computer becomes available ("harvest now, decrypt later"). Information that must stay confidential for many years is therefore already at risk from classical-only key agreement. NIST has standardised ML-KEM for key establishment and ML-DSA and SLH-DSA for signatures, and has published a transition timeline that deprecates quantum-vulnerable algorithms over the coming decade.⁶⁹⁷⁰⁷¹⁷²

39.3.2 Current security

Table 63: Post-quantum status by layer

Layer	Current algorithm	Post-quantum status
Message session establishment	PQXDH: X25519 plus Kyber-1024	Hybrid post-quantum key agreement IMPLEMENTED with single-use one-time Kyber prekeys; the last-resort Kyber prekey is used only when none remain
Message ratchet	Triple Ratchet: Double Ratchet (X25519) with SPQR (ML-KEM)	Hybrid post-quantum ratchet IMPLEMENTED through the current libsignal release
Message authentication and signed device lists	XEdDSA (Curve25519)	Classical only
Transport (TLS) at the edge	X25519MLKEM768 offered first, then X25519 and secp384r1; AES-256-GCM suites	Hybrid post-quantum key exchange IMPLEMENTED wherever the client platform's TLS stack offers it
Transport (TLS) outbound	Server runtime TLS offering X25519MLKEM768, X25519 and secp384r1	Hybrid IMPLEMENTED where the remote server supports it; classical otherwise
TURN relay hop	TLS 1.3 at the edge, TLS_AES_256_GCM_SHA384; the hybrid group is offered	Not claimed: depends on the TLS client inside each platform's WebRTC stack
Device client certificates, administrator security keys	ECDSA P-256; ES256 or EdDSA	Classical
Call media keys	Delivered over libsignal; AES-256-GCM frames	Inherits libsignal session protection; DTLS-SRTP is classical
Licence, release, timestamp, backup-manifest and audit-anchor signatures	Ed25519	Classical; hybrid Ed25519 plus ML-DSA ROADMAP (held until an audited ML-DSA implementation exists)
Local cache envelope	X25519	Classical
Symmetric encryption	AES-256	Considered adequate against known quantum attacks at 256-bit key size

⁶⁹NIST, "Module-Lattice-Based Key-Encapsulation Mechanism Standard" (ML-KEM), FIPS 203, August 2024.

⁷⁰NIST, "Module-Lattice-Based Digital Signature Standard" (ML-DSA), FIPS 204, August 2024.

⁷¹NIST, "Stateless Hash-Based Digital Signature Standard" (SLH-DSA), FIPS 205, August 2024.

⁷²NIST, "Transition to Post-Quantum Cryptography Standards", NIST IR 8547 (Initial Public Draft), November 2024.

39.3.3 Hybrid key exchange in transport

The edge offers the hybrid group X25519MLKEM768 first in TLS 1.3, followed by X25519 and secp384r1; the cipher suites are unchanged (AES-256-GCM only). The hybrid group combines X25519 with ML-KEM-768,⁷³ so a session is protected unless both components are broken: a harvested session cannot be decrypted later with a quantum computer, and a cryptographic weakness in ML-KEM cannot make it weaker than X25519 alone; the residual risk is an implementation defect in the newer code. Its adoption is a reviewed and recorded decision under SANKET's audited-library policy (Chapter 40).

Post-quantum key exchange is used wherever the client platform's TLS stack supports it. SANKET relies on each platform's own TLS stack rather than bundling a separate TLS provider, and the server offers the hybrid group on every outbound connection where the remote server supports it.

The edge records only the negotiated group name for each connection, with no client address, URI or user, so the share of post-quantum connections can be measured without logging who connected. TLS 1.3 authenticates the whole handshake transcript, including the groups the client offered, so a middlebox that tampers with the offered groups is detected.

39.3.4 Post-quantum roadmap

- Hybrid post-quantum key exchange on the remaining client paths, as their platform TLS stacks support it.
- Hybrid Ed25519 plus ML-DSA signatures for licences, release manifests and the trust bundle, both signatures over the same domain-separated message and both required. The design is complete; it is held until an ML-DSA implementation has a published independent audit or an issued CMVP certificate, in line with SANKET's rule that every cryptographic library must have one.⁷⁴

SECURITY PROPERTY - POST-QUANTUM POSITION

SANKET messaging is protected by hybrid post-quantum cryptography at both session establishment (PQXDH) and in the ongoing ratchet (SPQR), so an adversary must break both the classical and the post-quantum components. Transport key exchange is hybrid post-quantum at the edge for every client whose platform offers it, and on the server's outbound connections. Symmetric encryption uses 256-bit keys throughout. Signatures, transport for the remaining client paths and some auxiliary key exchanges remain classical and follow the roadmap above.

⁷³NIST, "Module-Lattice-Based Key-Encapsulation Mechanism Standard" (ML-KEM), FIPS 203, August 2024.

⁷⁴NIST, "Module-Lattice-Based Digital Signature Standard" (ML-DSA), FIPS 204, August 2024.

40 Cryptographic Certification Roadmap

Customers in regulated and national-security sectors may require evaluated cryptography and certified products. This chapter describes how SANKET approaches certification. No certification has been achieved at the date of this document.

40.1 Types of Requirement

- **Nationally approved cryptographic algorithms** or modules mandated by a national cryptographic authority.
- **Independent cryptographic evaluation** of the implementation and its use of primitives.
- **Product security certification** under a recognised scheme.
- **Sector-specific assurance**, such as security audits required for government deployment.

40.2 SANKET's Approach

SANKET's foundations are chosen to be evaluable: published, peer-reviewed protocols; the official libsignal implementation used without modification; standard primitives (AES-256-GCM, X25519, Ed25519, ECDSA P-256, SHA-256, HKDF, Argon2id, ML-KEM family, including the hybrid X25519MLKEM768 TLS group); and a small number of clearly defined places where cryptography is invoked. The crypto-agility mechanisms in Chapter 39 allow non-message layers to adopt approved primitives where a customer requires them.

Every cryptographic library is held to SANKET's rule of a published independent audit. Where no audited option exists, the exception is reviewed, approved and recorded, and the affected components are identified to evaluators in the controlled edition so that an evaluation can focus on them. Hybrid post-quantum signatures (Ed25519 with ML-DSA) for licences and release manifests are held until an ML-DSA implementation has a published independent audit or an issued validation certificate.⁷⁵

40.3 India-Specific Pathway

For Indian government and defence customers, Tosh Defence intends to pursue evaluation by the relevant national bodies, including security audit by a CERT-In empanelled auditor and evaluation of cryptographic implementations through the appropriate national scheme. These are CERTIFICATION TARGETS. Where national algorithms are mandated, they would be introduced as a separately evaluated protocol profile, not as an unreviewed change to the Signal Protocol implementation.

IMPORTANT - STATUS

SANKET holds no independent security certification and has not been formally accredited by any authority at the date of this document. The security reviews performed to date are internal engineering reviews. Customers should verify current certification status with Tosh Defence for the release they deploy.

⁷⁵NIST, "Module-Lattice-Based Digital Signature Standard" (ML-DSA), FIPS 204, August 2024.

41 Compliance, Assurance and Security Testing

Assurance comes from evidence that others can check. This chapter sets out the current status of formal assurance, the testing SANKET performs, and the independent assessment Tosh Defence recommends customers commission.

41.1 Assurance Status

Table 64: Assurance and certification status

Control / certification	Current status	Relevance	Target
ISO/IEC 27001 information security management	Not certified ⁷⁶	Organisational assurance for the vendor	CERTIFICATION TARGET
Secure software development practice	Internal practices (Chapter 30); not externally assessed	Supply-chain assurance	Alignment with NIST SSDF
CERT-In empanelled security audit	Planned	Required by many Indian government buyers	CERTIFICATION TARGET
STQC / national product evaluation	Planned	Indian government product assurance	CERTIFICATION TARGET
Independent vulnerability assessment and penetration test	Planned	Baseline assurance for any deployment	Recommended before production
Independent cryptographic review	Planned for SANKET's integration; libsignal and the Signal Protocol have published analyses	Assurance of correct use of primitives	CERTIFICATION TARGET
Source code review	Internal reviews performed; available to customers under agreement	Independent verification	Customer or third-party review
SBOM	Package-level CycloneDX SBOM per release image, signed with the release manifest and shipped in the offline package	Vulnerability management	IMPLEMENTED
Configuration hardening	Baseline in Chapter 42; installation gates	Deployment assurance	Customer acceptance
Customer accreditation	Customer-specific	Authority to operate	Per customer

41.2 Internal Security Reviews

SANKET has undergone several rounds of structured internal security review of the backend, mobile, desktop and administration code, with findings triaged by severity and remediated. Internal review complements, and does not replace, the independent assessment described below.

⁷⁶ISO/IEC 27001:2022, "Information security, cybersecurity and privacy protection - Information security management systems - Requirements".

41.3 Testing Performed

Table 65: Security-relevant testing in the engineering process

Activity	Current practice
Unit and route tests	Automated tests for authentication, device lifecycle, key directory, messaging, files, calls, integrations and policy enforcement
Local end-to-end suite	Checks against a running installation covering integrations, audit export and device-loss controls
Release freeze and rollback tests	Freeze and rollback attempts refused across the offline, connected and pre-flight upgrade paths
Upgrade compatibility	A contract test replays the previous release's client calls against the new server; automatic rollback is tested against upgrade failures
Media-server proof tests	Real DTLS-SRTP handshakes in both roles proving that only AEAD_AES_256_GCM is accepted; built into the image build
TLS configuration probes	Handshake probes confirming that only AES-256-GCM suites are accepted at the edge, on TURN over TLS and on outbound connections, and that the hybrid post-quantum group is offered first
Call-key interoperability	Equality of the client HKDF ratchet with the media frame-cryptor ratchet over many steps
Cryptographic policy checks	Pre-commit and lint checks for AES-256-only construction and the pinned patched WebRTC artefact
Container scanning	Image vulnerability scanning and container benchmark workflows
Device test matrices	Defined test matrices for calls (including calls over TCP 443 only on iOS, Android and desktop), network conditions, MDM and device-loss controls

41.4 Recommended Testing Programme

- **Threat modelling** of the customer's specific deployment, building on Chapter 5.
- **Code review** and **SAST** of the release, and **software composition analysis** of its dependencies.
- **DAST and API testing** of the edge and APIs, including authorisation tests across roles and memberships.
- **Mobile application security testing** against OWASP MASVS on both platforms.⁷⁷
- **Container and infrastructure assessment** against a container benchmark and NIST SP 800-190.^{78,79}
- **Penetration and red-team testing**, including insider and lost-device scenarios.
- **Configuration review** against the hardening baseline.
- **Restore testing and disaster-recovery exercises**, including a second-site failover drill that records the measured recovery point and recovery time.

41.5 Cryptographic Testing

Table 66: Cryptographic test areas

Area	What to test
Test vectors and known-answer tests	AES-256-GCM, HKDF, HMAC and Ed25519 helpers against published vectors; libsignal's own test suite

⁷⁷OWASP Foundation, "Mobile Application Security Verification Standard (MASVS)" v2, mas.owasp.org.

⁷⁸M. Souppaya, J. Morello, K. Scarfone, "Application Container Security Guide", NIST SP 800-190, September 2017.

⁷⁹Center for Internet Security, "CIS Docker Benchmark", cisecurity.org.

Area	What to test
Interoperability	Sessions between iOS, Android and desktop in all combinations; call key ratchet equality
Nonce uniqueness	Fresh random IV per encryption; one key per file
Randomness	Platform CSPRNG installed before any cryptographic library loads on mobile
Key lifecycle	Prekey rotation, replenishment, consumption; identity rotation; wipe and sign-out key deletion
Corrupted ciphertext	Modified messages, files and frames rejected
Replay	Duplicate messages, reused refresh tokens, reused TOTP challenge tokens, replayed call keys
Revoked keys and devices	Revoked devices excluded from fan-out and refused by the server; revoked device certificates refused at the edge
Device-list integrity	A device absent from the signed device list receives no copy on any fan-out path; a list that does not extend the previous one is refused
Security keys	Wrong origin, missing user verification, synced passkeys and sign-count regression refused
Expired certificates	Clients refuse expired or mismatched edge certificates; certificate pins enforced
Group membership changes	Removed members receive no new copies; call re-keying on join and leave
Downgrade	Server refuses lower call profiles; edge refuses weaker TLS suites; older or expired release manifests refused

42 Secure Deployment Baseline

A self-hosted platform is only as strong as the host it runs on. The baseline below separates what the SANKET installation provides from what the customer should apply.

42.1 Provided by the Platform

- One public TCP port (443, carrying HTTPS and TURN over TLS) plus the media UDP port; ICE over TCP optional; UDP TURN closed by default; API bound to loopback; data services without host ports.
- Edge on a digest-pinned nginx image whose OpenSSL supports ML-KEM, offering hybrid X25519MLKEM768 key exchange first, then X25519 and secp384r1, with AES-256-GCM suites only. The edge logs only the negotiated key-exchange group per connection, with no client address, URI or user, so the share of post-quantum connections can be measured.
- TURN terminated by the edge with TLS 1.3 and an AES-256-GCM suite only; the media server holds no certificate key, and a patch restricts its TURN relay to the node's own media address, so it cannot be used as an open relay.
- Device mutual TLS enforced per tenant setting; the installation gate refuses an edge configured for it without the customer CA and the device-certificate revocation list in place.
- No container-runtime socket in the API in the standard installation.
- Secrets service on an internal-only network with a least-privilege runtime token.
- Hardening overlay for the Edge profile: no-new-privileges on every service, per-service CPU and memory limits, log rotation and removal of the API host port.
- Every base and third-party image pinned by version and content digest, in every container build and every deployed service default, so a re-pushed or altered upstream tag cannot change what runs; a release check refuses any reference pinned by tag alone, and the installation check warns about an image override that is not pinned.
- Every SANKET-built container runs as a fixed non-root user; any component that must start with higher privilege drops it at start-up and is documented.
- In the Edge hardening overlay SANKET-built containers drop all Linux capabilities and run on a read-only root filesystem, with small in-memory scratch areas only for the paths the service writes; any documented exception keeps only the minimum the component needs.
- Generated secrets at installation; rejection of weak placeholder secrets in production.
- Installation gates that check, among other things, that every SANKET container runs non-root with no capabilities and a read-only root filesystem, that no placeholder secrets remain, that image references match the manifest and that the release verification key is present.
- Upgrade execution from the console off by default.

42.2 Operator Hardening Points

The single-host profile favours compatibility with the widest range of customer environments. Further hardening, such as hardened variants of third-party infrastructure images, seccomp or AppArmor profiles and the customer's own container benchmark, is applied by the operator to its own standard; inter-service traffic stays on isolated container networks inside the host, and an air-gapped or mirrored deployment stages the pinned third-party images in its internal registry.

42.3 Customer Hardening Baseline

Table 67: Recommended host and deployment hardening

Area	Recommendation
Operating system	Minimal, supported server OS; automatic security updates through an internal mirror
Boot integrity	UEFI Secure Boot; TPM 2.0
Disk encryption	Full-disk encryption of all installation volumes, keyed to TPM with a recovery key under dual custody
Mandatory access control	SELinux or AppArmor in enforcing mode
Host firewall	Allow only TCP 443 and the media UDP port inbound (and the ICE-over-TCP port if enabled); deny all outbound except explicitly approved destinations (SIEM, webhooks, the customer's identity provider, optional Apple push)
Administrator sign-in	Keep the security-key requirement for every administrator; register more than one key per administrator; review administrator sign-in audit events
Device mutual TLS	Require device mutual TLS for every device
Usage analytics	For high-sensitivity customers, set the installation-level analytics lock at provisioning: analytics is then forced off, clients collect nothing, ingestion is refused and any queued events are cleared. Other organisations can switch analytics off in the console (a step-up confirmation and a high-severity audit event) and delete collected analytics
Inter-host links (high availability)	Carry replication between hosts and sites on a dedicated network segment
Administration	Key-based SSH from a bastion only; MFA on the bastion; no password authentication; separate administrator identities
Management interface	Isolated management network; administrator console reachable only from it
Containers	Mirror the pinned image digests to the internal registry and keep any image override pinned by digest; prefer non-root variants of third-party images
Secrets	Restrict the environment file to the service account; keep OpenBao unseal shares and backup passphrase with separate custodians
Egress	Set the media server node address or an internal STUN server; leave Apple push disabled where iOS wake-ups are not required
Host auditing	auditd or equivalent; forward host and container logs to the internal SIEM
Time	Internal authoritative time source
Backups	Scheduled encrypted backups to separate media; periodic restore tests
Immutable infrastructure	Rebuild hosts from known-good images rather than patching in place where practical
Monitoring	Certificate expiry, disk, memory and service health alerts

43 External Dependency Analysis

The table lists every category of external service a communications platform commonly depends on, and how SANKET handles it in connected and air-gapped deployments.

Table 68: External dependency analysis

Dependency	Connected deployment	Air-gapped deployment	Customer-controlled alternative
Content delivery network	None. Applications and assets are served by the installation or bundled	None	-
Analytics	None third-party. First-party, content-free, pseudonymous rollups computed inside the installation, on by default	Same	Switch analytics off in the console, or set the installation-level lock at provisioning to force it off
Crash reporting	None	None	-
Fonts	Bundled with the applications	Same	-
Push notifications (Android)	Self-hosted ntfy in the installation; the release app carries no Firebase messaging library	Same	-
Push notifications (iOS)	Apple push service (generic payloads), if enabled	Not available; no background wake-up	None available from Apple; foreground delivery
Email	In-installation relay; direct delivery or customer relay; off by default	Internal mail relay or disabled	Customer mail system
SMS	Off by default; customer-selected provider (Indian or international) for verification only	Disabled	Customer SMS gateway through a supported provider, or none
Certificate authority	ACME DNS-01 or customer CA	Customer internal CA	Customer PKI
DNS	Public or private DNS for installation host names	Internal DNS	Customer DNS
NTP	Customer time source	Internal time source	Customer NTP
STUN (media server)	None: the node address is set explicitly at installation and STUN discovery is off	Not used	Installer and security gate refuse a missing node address or STUN discovery
Object storage	MinIO in the installation	Same	-
Package repositories	Offline package or optional connected download from Tosh Defence	Offline package only	Customer staging repository
Third-party base images	Pulled by the operator from the vendor registry or public registries	Pre-staged by the customer	Customer internal registry
Monitoring	Built-in monitors; optional customer metrics and tracing collectors	Same	Customer monitoring
Logging	Installation logs; syslog to customer SIEM	Internal SIEM	Customer SIEM
Licensing	Local verification; optional signed status report to Tosh Defence	Local verification; offline renewal; no status report	-

Dependency	Connected deployment	Air-gapped deployment	Customer-controlled alternative
Identity provider	Optional: the customer's own OIDC, SAML or LDAP directory; never a foreign provider (public identity providers are refused). Optional SCIM from the customer directory	Same, inside the enclave	Customer directory
Security-key verification (WebAuthn)	Verified inside the installation on platform cryptography, with no external attestation or metadata service	Same	-
Device attestation (Android)	Verified offline against vendor roots and a revocation snapshot shipped in each release; no runtime call to Google	Same	-
Key management	OpenBao in the installation	Same	Customer HSM where policy requires (evaluation)
Backup destinations	Operator-chosen: local or network-attached storage, S3-compatible object storage or SFTP, all customer-operated; there is no foreign cloud adapter	Offline media	Customer storage

43.1 Security-Relevant Libraries and Runtimes

SANKET's rule is that every cryptographic library has at least one published independent security audit. Any exception is reviewed, approved and recorded, with how its exposure is limited, and is detailed for evaluators in the controlled edition.

Table 69: Security-relevant libraries and runtimes

Component	Role	Audit position and limits
libsignal (official)	Signal Protocol for messages, call keys, broadcasts and device-list signatures, through the official Swift, Java and TypeScript bindings (current audited release)	Signal's official implementation; the only end-to-end protocol code in SANKET (Chapter 10)
OpenSSL in the edge	TLS termination in the edge, offering hybrid X25519MLKEM768 key exchange	A widely deployed TLS library; the hybrid group combines ML-KEM with X25519, so a weakness in either alone does not break the key exchange ⁸⁰
Server runtime	Node.js on a supported LTS line for the API, workers and the vendor super-administration API, digest-pinned; outbound TLS offers the hybrid group too	Platform cryptography used through the runtime's bundled OpenSSL
Desktop runtime	Electron on a supported release line	Kept on a line that receives security updates; all cryptography runs in the main process
Android release build	Push through the self-hosted relay	Carries no Firebase messaging library

⁸⁰NIST, "Module-Lattice-Based Key-Encapsulation Mechanism Standard" (ML-KEM), FIPS 203, August 2024.

KEY POINT - ARCHITECTURAL STATEMENT

SANKET is designed with no mandatory foreign runtime dependency for sovereign deployment. An air-gapped installation operates with no external service at all. A connected installation can be configured with none, at the cost of iOS background wake-ups. Build-time tooling used by Tosh Defence (source hosting, public package and image registries, mobile build services) is a supply-chain consideration addressed by the signed release process in Chapter 30, not a runtime dependency of the customer's installation.

44 Privacy Architecture

SANKET is designed to collect what it needs to deliver communications securely, to keep it inside the customer's boundary, and to delete it on a defined schedule. Customers remain the controllers of personal data in their installations; this chapter describes the technical measures, not legal compliance.

44.1 Principles Applied

- **Data minimisation:** content is end-to-end encrypted, including reactions, encrypted broadcasts and classification labels; phone and email are optional; notifications carry no content by default; the user directory is not enumerable.
- **Purpose limitation:** metadata is used for delivery, security and accountability; usage analytics are first-party, content-free counters computed within the installation, which the organisation can switch off and an installation can lock off, after which nothing is collected.
- **Administrator visibility** is limited by role and never extends to content.
- **Deletion:** delivered ciphertext is tombstoned; disappearing messages expire; retention windows prune metadata; account self-deletion (optional) revokes devices and removes the account from use.
- **Access records:** administrator actions and security events are audited.

44.2 Retention Schedule

Table 70: Default retention of server-held data

Data	Default retention	Mechanism
Message ciphertext and its metadata (reactions included)	72 hours after delivery to all copies	Scheduled tombstoning job
Encrypted broadcast ciphertext	Until each device confirms it has stored its copy, otherwise until the broadcast expires	Deleted per device on confirmation; expiry
Message stubs	90 days	Hard deletion
Disappearing messages	At expiry	Scheduled purge job
Message partitions	90-day hot window (configurable)	Partition drop
Broadcast records (notice content; encrypted broadcast metadata), recipients and notifications	90 days (configurable)	Partition drop
Call logs	365 days (configurable), rolled up into daily statistics	Batched deletion
Security alerts	365 days (configurable)	Partition drop
Notification delivery metrics	90 days	Partition drop
Attachment objects	90 days	Object-storage lifecycle
Vault backup versions	Current plus non-current versions for 30 days	Object-storage versioning

Data	Default retention	Mechanism
Analytics rollups	400 days (configurable, minimum 30 days); none collected while analytics is off	Purge job; "Delete all collected analytics" on demand
Webhook delivery records	30 days	Worker deletion
Audit log	Indefinite unless pruning is explicitly enabled (default setting 365 days)	Chapter 21
Backups	14 days, minimum seven archives	Backup script

Shrinking a retention window is itself an audited administrative action.

44.3Personal Data Inventory

Personal data held by the server comprises the Sanket ID, display name, optional email and phone, profile fields, device information (platform, model, app version, hardware protection level and the fingerprint of the device's client certificate, which itself carries an opaque random name), each account's signed device list, IP addresses and user agents in audit records, group memberships and names, and communication metadata (Chapter 20). Where federated sign-in is configured, it also holds the mapping from the customer's identity provider or directory to the administrator account or Sanket ID; accounts are never created from the provider. Email and phone are stored in clear.

The server does not hold the emoji of any reaction (where the large-group count option is on, it learns only who reacted to which message), the title, body or attachments of an encrypted broadcast (only its audience, priority, deadline, expiry and receipts; notice broadcasts remain readable), or the classification label of any message or file. Usage analytics are pseudonymous counts, collected only while the organisation's switch is on and never on an installation locked against them; the organisation can delete everything collected at any time. Audit records and the app version each device reports for update enforcement are not analytics and are kept regardless. End users can review their own security activity. Export of user data by users is off by default.

45 Security and Usability Trade-offs

Every strong security property costs something. Stating the costs openly allows customers to choose policy deliberately.

Table 71: Principal trade-offs and how customers choose

Trade-off	What is gained	What is lost	Customer choice
End-to-end encryption vs server-side search	Server cannot read content	No server-side full-text search or content supervision	Fixed: search is on-device only
Metadata minimisation vs auditability	Less data to leak	Less forensic detail	Retention windows; audit export
Message expiry vs record retention	Less exposure on lost devices	Records disappear	Allow, force or disable disappearing messages
Air gap vs mobile push	No external path	No background wake-up on iOS	Deployment mode
Strong device trust vs friction	Rogue devices blocked	Approval delays; caps	Approval mode; device caps
Confidentiality vs data-loss prevention	Content opaque to servers	No server-side DLP inspection	Endpoint DLP on managed devices; download and export policy
Offline access vs immediate revocation	Work continues offline	A device offline at revocation keeps its local data until it reconnects or is wiped	Lost mode, wipe, disappearing messages
Screenshot blocking vs usability	Deters capture	Users cannot capture legitimately	Configurable; detection alerts
Identity vault vs keys-never-leave-device	Recovery after loss	Encrypted identity key on server	Enable or disable vault
TOTP SHA-256 vs authenticator choice	Stronger algorithm	Some popular apps incompatible	Fixed: SHA-256 only
Pairwise group fan-out vs efficiency	Per-device forward secrecy; simple removal	Bandwidth grows with group size	Group size limit
Encrypted broadcasts vs scheduling and convenience	The server cannot read, downgrade, move or replay a broadcast	No scheduling; sending only from an approved desktop; fan-out time grows with recipient devices	Encrypted broadcasts for sensitive content; notice broadcasts for scheduled, non-sensitive announcements
Encrypted reactions vs server-side counts	The server never learns who reacted with which emoji	Counts are computed on each device	For groups of 500 or more active members, an option (off by default) lets the server count reactions without the emoji
Security keys vs administrator convenience	Phishing-resistant administrator sign-in	Keys to issue, carry and replace; synced passkeys refused by default	Configurable per installation; required on new installations
Classification handling vs workflow	Higher-ranked content cannot be forwarded, exported or downloaded	Users cannot move that content out of the app, even legitimately	Label names and ranks; default label

Trade-off	What is gained	What is lost	Customer choice
Device mutual TLS vs operational simplicity	Only enrolled devices complete a TLS connection to the API	A certificate lifecycle and revocation list to operate	Configurable per tenant
Asynchronous second site vs zero data loss	Writes are not tied to the inter-site link	Writes since the last replicated point can be lost when a site fails	Single-host or high-availability profile; drills record the measured figures
Usage analytics vs data minimisation	First-party operational insight	Pseudonymous usage counts held on the installation	Switch off; installation lock at provisioning; delete collected analytics

An external camera can always photograph a screen. No screenshot control changes that, and SANKET does not suggest otherwise.

46 What SANKET Does Not Claim

Precise claims are more useful than broad ones. The statements below define the edges of SANKET's security properties.

1. End-to-end encryption cannot protect plaintext on a fully compromised authorised endpoint.
2. Air-gapping does not eliminate insider risk; it removes external paths, not trusted people.
3. Screenshot restrictions cannot prevent an external camera.
4. Revocation cannot erase plaintext that was already legitimately obtained or exported.
5. Encryption does not replace endpoint security, device management or user training.
6. The server is not blind to metadata: it knows who communicates with whom and when, including the sender of every message, and it can read notice broadcasts, which are labelled as not end-to-end encrypted (Chapter 20). Sealed sender is not implemented.
7. SANKET is not "zero knowledge" in the cryptographic sense.
8. Post-quantum protection covers messaging (PQXDH and SPQR) and, wherever the client platform's TLS stack supports it, hybrid key exchange at the edge. Other transport paths and all signatures (identity keys, licences, release manifests, device certificates) remain classical and follow the roadmap in Chapter 39.
9. SANKET holds no independent security certification at the date of this document. Certification requires independent assessment.
10. Software classification labels do not confer accreditation to process government classified information. Their handling rules are enforced by SANKET's own clients; separation between accredited levels is by installation.
11. Signed device lists stop the server adding a device to an account, but the first identity key seen for a new contact is trusted on first use; safety-number comparison is how a substituted identity is detected.
12. Infrastructure security, host hardening, physical security and key custody remain the customer's responsibility in a self-hosted deployment.
13. High availability across sites is asynchronous: writes since the last replicated point can be lost when a site fails. Recovery figures are design targets that each deployment's drill measures, not guarantees.
14. Key protection on each device is bounded by the key store its platform provides.
15. A file already delivered cannot be revoked per share, and roadmap capabilities such as push-to-talk, forensic watermarking and conversation-level classification labels are not claimed.
16. No responsible platform can guarantee absolute immunity from compromise.

47 Why Air-Gapped SANKET Is Structurally Different

Most security improvements reduce the probability of an attack succeeding. An isolated deployment does something different: it removes whole categories of attack path, because the parties and systems on those paths are no longer involved.

47.1 Trust Removed

Table 72: Categories of external trust removed by an air-gapped deployment

Category removed	Attack paths that disappear
Foreign cloud provider	Provider insiders, provider compromise, cross-tenant attacks, provider-side key access
External legal jurisdiction	Compelled disclosure or modification by a foreign authority
External analytics and telemetry	Leakage of usage patterns and device data to third parties
Internet identity systems	Account takeover through a third-party identity provider
Public runtime APIs	Service disruption or data exfiltration through API dependencies
Public storage and CDNs	Content or code substitution by a third-party host
External key management	Key access by a cloud KMS operator
External key directories and attestation services	Device or key substitution by a third-party key transparency service; dependence on a platform vendor's online attestation service (signed device lists are kept by the installation, and Android attestation is verified offline)
External logging	Disclosure of security events to a log provider
External licence validation	Remote disablement through a licensing service
Internet adversaries	Remote exploitation, scanning, volumetric denial of service from the internet

47.2 Control Retained

What remains is inside the customer's control: servers, keys, identities, network, storage, audit records, updates, certificates, backups, administrators, policies and monitoring. Every remaining risk is a risk the customer can see, govern and test.

SECURITY PROPERTY - THE STRUCTURAL ARGUMENT

In a conventional service, the customer's security depends on the security of every external party in the delivery chain, most of which the customer cannot inspect. In an air-gapped SANKET installation, combined with end-to-end encryption that keeps content from the customer's own servers, security depends on the customer's own people, endpoints and infrastructure, and on the integrity of the software it chose to install, which it can verify. That is a smaller, inspectable trust base.

The remaining risks are real: insiders, endpoint compromise, the integrity of the release chain, and the operational discipline of the customer's own teams. Chapters 27 to 30 address them.

48 Comparative Architecture

The comparison below is between architectural categories, not named products. Individual products within a category vary; the columns describe typical properties of the category. For the SANKET column, the properties are those of the described release. Protocol properties such as forward secrecy and post-compromise security differ materially between products in every category and should be checked against published analyses rather than assumed.

Table 73: Architectural categories compared

Property	A. Consumer cloud messenger	B. Enterprise cloud collaboration platform	C. Self-hosted encrypted platform (typical)	D. Sovereign air-gapped SANKET deployment
Deployment ownership	Provider	Provider	Customer	Customer
Internet requirement	Required	Required	Usually optional	None
External telemetry	Common	Common	Varies	None
Third-party analytics	Common	Varies	Varies	None
Content encryption	Often end-to-end	Often provider-readable	Varies	End-to-end (libsignal, including reactions, encrypted broadcasts and classification labels; per-file AES-256-GCM)
Administrator authentication	Provider-managed	Single sign-on, often with FIDO2	Varies	FIDO2 security keys verified by the installation; federation only to the customer's identity provider
Device-list integrity	Varies; some publish key transparency	Provider directory	Usually server-asserted	Device lists signed by each account's devices, verified by senders against the installation's own log
Private PKI	No	Limited	Usually	Yes
Air-gap support	No	No	Varies	Yes
Customer key control	No	Sometimes (customer-managed keys)	Usually	Content keys on endpoints; server keys in customer OpenBao
Customer-controlled logs	No	Partial	Yes	Yes, hash-chained, SIEM export
Offline updates	No	No	Varies	Yes, signed offline packages
Jurisdiction exposure	Provider's jurisdiction	Provider's jurisdiction	Customer's	Customer's
Organisational policy controls	Minimal	Extensive	Varies	Device governance, retention, feature and access policy

Property	A. Consumer cloud messenger	B. Enterprise cloud collaboration platform	C. Self-hosted encrypted platform (typical)	D. Sovereign air-gapped SANKET deployment
Crypto-agility	Provider decides	Provider decides	Varies	Profile-negotiated calls; versioned protocol and content envelope; hybrid post-quantum TLS key exchange offered first; single crypto helpers
High availability	Provider-managed	Provider-managed	Varies	Multi-host profile with an asynchronous second site, or single host
Certification flexibility	Provider decides	Provider certifications	Varies	Customer-specific evaluation possible; no certification yet

49 SANKET within the Tosh Defence Ecosystem

Tosh Defence Private Limited builds sovereign software for high-trust environments, designed to be deployed on customer-controlled infrastructure. SANKET secures communications. Other Tosh Defence products address adjacent needs, including MAYA (cyber deception), DRISHYA (infrastructure visibility), SAJAG (predictive maintenance intelligence), RecordsKeeper AI (secure records intelligence) and TOSH AI (sovereign AI reasoning). Each is a separate product with its own architecture; nothing in this whitepaper describes or depends on them, and SANKET does not share data with them.

50 Technical FAQ

Can Tosh Defence read customer messages?

No. Tosh Defence does not operate customer installations and has no access path into them. Message content is end-to-end encrypted with keys that exist only on users' devices.

Can customer administrators read communication content?

No. Administrators govern identities, devices and policy; the server holds no content keys. Administrators with database access can read server-visible metadata as described in Chapter 20, and notice broadcasts, which are labelled as not end-to-end encrypted.

What happens if a SANKET server is compromised?

The attacker gains metadata and the ability to disrupt service. Signed device lists prevent it from adding a device to an account to receive copies, and safety-number comparison verifies identities. Past message and file content remains confidential (Chapter 28).

What happens if an endpoint is stolen?

If locked, the data is protected by operating-system encryption, the app lock and encrypted local storage. The device can be placed in lost mode, revoked and remotely wiped. If the attacker obtains an unlocked device, they can see what the user could see until the device is locked or revoked.

Does SANKET require public internet?

No. It runs on private or air-gapped networks.

Can SANKET operate completely air-gapped?

Yes, with internal CA, DNS and time, self-hosted push for Android and offline releases and licences. iOS devices work without background wake-ups in that mode.

Does SANKET require foreign cloud services?

No. The media server never uses public STUN; its address is set at installation, and the TURN relay runs on the installation. Android Key Attestation is verified offline against vendor roots shipped in each release. The remaining optional connected-mode paths (Apple push for iOS and the optional status report) can each be disabled.

Where are keys stored?

End-user keys on the device, protected by the platform key store (on iOS under a key derived from a Secure Enclave key, on Android under a StrongBox or TEE key); server keys inside the installation; only public licence and release keys come from Tosh Defence. See Chapter 11.

Does SANKET support hardware-backed keys?

Yes. On iOS the protocol store, device secrets, the device certificate key and the account list key are encrypted with AES-256-GCM under a key derived from a Secure Enclave key; on Android they are protected by a StrongBox key where the phone has one and the TEE otherwise. libsignal identity keys stay in libsignal format, wrapped by the hardware key, because the Secure Enclave cannot hold Curve25519 keys. Android Key Attestation is verified offline; the protection level is recorded per device and policy can require one. Administrators sign in with FIDO2 security keys.

Can SANKET operate with private PKI?

Yes. The edge certificate and every device's client certificate can be issued under the customer's CA, and the SIEM connection pins the customer's CA.

Can SANKET integrate with LDAP or Active Directory?

Yes. User lifecycle is driven from the directory through SCIM 2.0 provisioning. An installation can also require the user's directory password, over LDAPS or StartTLS, when a device is enrolled or restored or a new device signs in; this is an extra check beside TOTP and device approval, never a replacement. Everyday sign-in stays local, so field devices work without the directory.

Does SANKET support single sign-on?

For the administration console, yes: through the customer's own OpenID Connect or SAML 2.0 identity provider. The provider is a first factor only, the local security key is still required, public identity providers are refused and no account is created on first sign-in. End users sign in locally.

Can retention be configured?

Yes, within limits: message hot window, broadcasts, call logs, alerts, disappearing messages and more (Chapter 44).

Can files be revoked?

Removing a member or revoking a device prevents future access. Files already obtained by an authorised recipient cannot be recalled by any platform.

How are backups protected?

Each component is encrypted with its own AES-256-GCM key. The recovery artefact's key is wrapped under a passphrase-derived key (scrypt) and every other key by the installation's OpenBao; a manifest signed with a non-exportable Ed25519 key is verified, with every artefact, before a restore changes anything. Scheduled database backups are signed and verified the same way, and content inside is already end-to-end encrypted. Hold the passphrase and the OpenBao unseal material separately from the backup media (Chapter 32).

How are air-gapped updates performed?

With a signed offline package, verified with the release public key, staged and installed by the customer with automatic rollback (Chapter 30).

What appears in audit logs?

Security and administrative events with actor, action, target, device, address, result and time. Never content, ciphertext or keys.

Is SANKET independently certified?

Not at the date of this document. Certification targets are described in Chapters 40 and 41.

Is SANKET post-quantum secure?

SANKET messaging uses hybrid post-quantum cryptography for both session establishment (PQXDH with Kyber-1024) and the ongoing ratchet (SPQR). The edge offers the hybrid X25519MLKEM768 TLS key exchange first, and it is negotiated wherever the client platform's TLS stack supports it. Symmetric encryption is 256-bit throughout. Signatures remain classical (Chapter 39).

What is protected if the server is compromised?

Message bodies, reactions, encrypted broadcasts, classification labels, file contents, call media and private keys.

Are reactions end-to-end encrypted?

Yes. A reaction is a small libsignal message; the server stores only opaque per-device copies and each device computes the counts. For groups of 500 or more active members a tenant option, off by default, lets the server keep a count per message: it then learns who reacted to which message, never the emoji.

Are broadcasts end-to-end encrypted?

Encrypted broadcasts are. They are sent from a dedicated broadcast-sender account's approved desktop, which encrypts a copy for every recipient device with libsinal; priority, deadline and expiry are inside the encryption too, so the server cannot downgrade or replay them. Notice broadcasts, for non-sensitive scheduled announcements, are readable by the server and labelled "Notice (not end-to-end encrypted)".

Does SANKET support classification labels?

Yes. The customer defines up to eight ranked labels. A label travels inside the end-to-end encrypted message, so the server can neither read nor change it, and content above the lowest rank cannot be forwarded, exported or downloaded. Separation between formally accredited levels remains by installation (Chapter 38).

What is exposed if the endpoint is compromised?

Everything that endpoint can decrypt and display while compromised.

How does device revocation work?

The server marks the device revoked; its next request or refresh fails; it is excluded from future fan-outs; its client certificate is revoked; a remote wipe can also be requested.

What happens when a user leaves the organisation?

Suspend or delete the account (SCIM delete suspends), which revokes access; revoke or wipe the user's devices; remove them from groups. Their Sanket ID is not reused.

Can SANKET support multiple sites?

Yes. The high-availability profile runs on several hosts with Docker Compose and replicates the databases, object storage and secrets service asynchronously to a second site, with drill tooling to fence, promote, switch and fail back. The reference topology is designed for a recovery point of 5 minutes and a recovery time of 1 hour; each deployment's disaster-recovery drill records its measured figures. Separate installations and a standby restored from backups remain available (Chapter 32).

Is SANKET highly available?

With the multi-host profile, yes: two edge hosts, more than one API, WebSocket and media node, replicated PostgreSQL with fenced failover, Redis with Sentinel, distributed MinIO and a three-node OpenBao cluster. The single-host profile remains available for small and disconnected sites. Kubernetes is on the roadmap.

Can SANKET support multiple departments or business units?

Within an installation, through groups and administrator roles; for strong separation, through separate installations. Hierarchical units are a roadmap item.

Can SANKET integrate with existing internal systems?

Yes: Integration API with scoped credentials, SCIM 2.0, syslog to SIEM and signed webhooks, documented in OpenAPI.

What metadata does SANKET retain?

See the inventory in Chapter 20 and the retention schedule in Chapter 44.

Can all telemetry remain local?

Yes. Product applications contain no third-party analytics or crash reporting; first-party usage counts stay in the installation, and the organisation can switch them off or the installation can be locked with them off; the status report to Tosh Defence is not sent under an offline licence.

How does licensing work in a disconnected deployment?

The licence is a signed file verified locally. Renewals are signed offline bundles bound to the tenant with an increasing counter.

Appendix A - Cryptographic Algorithm Reference

Table 74: Algorithms and libraries used in the described release

Purpose	Algorithm / construction	Library or component	Reference
Message key agreement	PQXDH: X25519 + Kyber-1024, HKDF	libsignal (current release)	PQXDH rev. 3; RFC 7748
Message encryption	Triple Ratchet (Double Ratchet + SPQR); AES-256-CBC + HMAC-SHA256 per message, inside a versioned content envelope	libsignal (current release)	Double Ratchet spec; SPQR
Identity signatures	XEdDSA over Curve25519	libsignal	XEdDSA spec
File encryption	AES-256-GCM, 96-bit IV, 128-bit tag, random key per file	noble ciphers library (mobile); Node.js crypto (desktop)	FIPS 197; SP 800-38D
Call frames	AES-256-GCM with HKDF-SHA256 key ratchet	Patched WebRTC SDK (mobile); livekit-client (desktop)	RFC 5869
Call transport	DTLS-SRTP, AEAD_AES_256_GCM	SANKET LiveKit build (pion)	RFC 5764; RFC 7714
TLS	TLS 1.3 AES-256-GCM-SHA384; TLS 1.2 ECDHE AES-256-GCM; TURN over TLS on TCP 443: TLS 1.3 TLS_AES_256_GCM_SHA384 only	nginx; Node.js	RFC 8446; RFC 5289
TLS key exchange	Hybrid X25519MLKEM768 (X25519 + ML-KEM-768) offered first, then X25519 and secp384r1	nginx with a current OpenSSL release; Node.js LTS	FIPS 203; RFC 7748
Device mutual TLS	ECDSA P-256 key generated on the device; X.509 certificate issued under the customer CA with an opaque name; CRL at the edge	Platform crypto (device); OpenBao PKI (issuing); nginx (verification)	RFC 5280
Local database	SQLCipher (AES-256, HMAC-SHA256 page authentication)	SQLCipher	SQLCipher design
Local key envelope	X25519 ECDH + HKDF-SHA256 + AES-256-GCM	noble curves and ciphers; Node.js crypto	RFC 7748; RFC 5869
Push payloads	AES-256-GCM, fixed-size padding	Node.js crypto	SP 800-38D
Locate	HPKE (DHKEM X25519, HKDF-SHA256, AES-256-GCM)	libsignal (phone); noble (browser)	RFC 9180
Hardware key protection	iOS: Secure Enclave P-256 ECDH + HKDF-SHA256, deriving an AES-256-GCM key; Android: Keystore key in StrongBox or the TEE	Platform crypto	RFC 5869; SP 800-38D
Android Key Attestation	X.509 attestation chain verified offline to vendor roots shipped in the release, with a per-release revocation snapshot	SANKET API	RFC 5280
Signed device lists	XEdDSA signatures by the account list key over each list version, each list chained to the hash of the previous one	libsignal	XEdDSA spec

Purpose	Algorithm / construction	Library or component	Reference
Passwords	Argon2id (64 MiB, t=3, p=4)	argon2	RFC 9106
Second factor	TOTP, HMAC-SHA256, 6 digits, 30 s	SANKET implementation	RFC 6238
Administrator security keys	WebAuthn assertions with ES256 (ECDSA P-256, SHA-256) or EdDSA (Ed25519); user verification required	Node.js crypto	W3C WebAuthn Level 2; RFC 8032
Federated sign-in	OIDC: authorisation code with PKCE S256, ID token signature verified with asymmetric algorithms only (no none or HMAC). SAML 2.0: assertion signature by the pinned certificate, RSA-SHA256 or stronger	SANKET API	OpenID Connect Core 1.0; SAML 2.0
Passcode and vault	PBKDF2-HMAC-SHA256, 600,000 iterations; AES-256-GCM	Platform crypto	SP 800-132 (PBKDF2)
Session tokens	JWT HS256 (HMAC-SHA256)	SANKET API	RFC 7519; RFC 8725
Webhooks	HMAC-SHA256 over timestamp and body	Node.js crypto	RFC 2104
Integration secrets, invite codes	SHA-256 hashes; HMAC-SHA256 for activation codes	Node.js crypto	FIPS 180-4
Licences, releases (with sequence number and expiry), release timestamps, status reports, backup manifests, audit anchors	Ed25519 (release timestamps under a key separate from the release key)	OpenBao Transit (signing); noble curves / Node.js (verification)	RFC 8032
Audit chain	SHA-256 hash chain, head signed periodically (Ed25519 anchor)	SANKET API; OpenBao Transit	FIPS 180-4; RFC 8032
Backups (released)	AES-256-GCM per artefact; recovery key under scrypt; data keys wrapped by OpenBao Transit; Ed25519-signed SHA-256 manifest	Node.js crypto; OpenBao Transit	SP 800-38D; RFC 7914; RFC 8032
TOTP secrets at rest	AES-256-GCM, key derived with HKDF-SHA256 from a dedicated installation secret, bound to the account	Node.js crypto	SP 800-38D; RFC 5869

Libraries are chosen under SANKET's audited-library policy (Chapter 40). libsignal is used without modification.

Appendix B - Key Hierarchy

The complete key inventory, with origin, storage, purpose, rotation and exposure consequence for each key, is in Chapter 11 (Key Management Architecture) and its figure. In summary:

Tier	Keys	Custodian
Endpoint content tier	libsignal identity, prekeys and sessions; per-file keys; call sender keys; local cache DEK and envelope key; account list key; device certificate key; hardware wrapping key (Secure Enclave, StrongBox or TEE)	User device
Administrator tier	FIDO2 security-key private keys (only public keys registered on the installation)	Administrator
Installation tier	Edge TLS key; device certificate issuing key (OpenBao PKI); token secrets; field and TOTP keys; OpenBao Transit keys; push payload keys; identity-provider secrets; backup passphrase	Customer operator
Customer PKI tier	Root and issuing CA keys	Customer PKI administrator
Vendor tier	Licence, release and release-timestamp signing keys (public halves only in the installation)	Tosh Defence

Appendix C - Security Control Matrix

The full security control matrix, mapping objectives to threats, controls, layers, standards and residual risk, is in Chapter 34. The status of each principal capability is in Chapter 2.

Appendix D - Threat Model Summary

Table 75: Threat actors by STRIDE category

STRIDE category	Principal actors	Principal controls
Spoofing	Credential thief, rogue device, man-in-the-middle, compromised server	TOTP; administrator security keys; device states and caps; signed device links and device lists; device client certificates; identity pinning; certificate validation
Tampering	Network adversary, compromised server, supply-chain attacker	libsignal MACs; GCM tags; TLS; signed releases with sequence and expiry; labels and broadcast priority inside the encryption
Repudiation	Malicious insider, privileged administrator	Hash-chained audit; SIEM export
Information disclosure	Eavesdropper, database or storage thief, metadata analyst, endpoint malware	E2EE; local encryption; retention; self-hosting
Denial of service	DoS attacker, compromised network device	Rate limits; caps; block manager; customer DDoS protection
Elevation of privilege	External attacker, lateral mover, malicious update	RBAC; least-privilege tokens; network isolation; signed updates

The full matrix of twenty-three actors is in Chapter 5.

Appendix E - Deployment Reference Architectures

Architecture	Status	Chapter
Single-site, single-host (Edge profile)	IMPLEMENTED	25
DMZ-segmented	DEPLOYMENT-DEPENDENT	25
Air-gapped security enclave	IMPLEMENTED	23
Highly available on-premise (multi-host Docker Compose)	IMPLEMENTED	25, 32
Multi-site with asynchronous replication to a second site	IMPLEMENTED	25, 32
Kubernetes	ROADMAP	25
Disaster recovery by backup and standby host	IMPLEMENTED	32

Appendix F - Air-Gap Deployment Checklist

- ☐ Enclave network has no route to the internet, verified by testing egress from the installation host.
- ☐ Internal CA issues the edge certificate; clients trust the internal CA.
- ☐ Internal DNS resolves all installation host names.
- ☐ Internal authoritative time source configured on servers and managed devices.
- ☐ Licence issued for air-gapped deployment and offline operation; trust bundle installed.
- ☐ Release public key obtained out of band and configured; verification succeeds.
- ☐ Third-party base images pre-staged from a verified source.
- ☐ Media server node address set explicitly or internal STUN configured; TURN relay enabled only if needed inside the enclave.
- ☐ Android attestation verified against the roots and revocation snapshot shipped in the release; no online attestation check.
- ☐ Identity provider and LDAP directory, if used, hosted inside the enclave.
- ☐ Device client-certificate revocation list published to the edge.
- ☐ Apple push disabled; iOS behaviour without background wake-up accepted by users.
- ☐ SMS disabled; email relay internal or disabled.
- ☐ Android devices registered with the self-hosted push relay.
- ☐ Desktop clients built with the installation's host allowlist.
- ☐ Syslog configured to the internal SIEM with the internal CA pinned.
- ☐ Controlled media transfer procedure documented, with scanning and custody records.
- ☐ Encrypted backups written to separate media inside the enclave; restore tested.
- ☐ OpenBao unseal shares and backup passphrase held by separate custodians.

Appendix G - Security Hardening Checklist

- ☐ Hardened, minimal host OS with Secure Boot and TPM; full-disk encryption on all volumes.
- ☐ SELinux or AppArmor enforcing; host firewall allowing only HTTPS and media ports inbound.
- ☐ Default-deny egress with explicit exceptions.
- ☐ Bastion-only, key-based SSH with MFA; administrator console restricted to the management network.
- ☐ All images pinned by digest; Edge hardening overlay applied; installation gates pass.
- ☐ Release public key configured; upgrades verified before load.
- ☐ Tenant policy: TOTP required; invitation-only or administrator-only registration; administrator device approval; member approval for groups.
- ☐ Security keys required for every administrator; synced passkeys refused.
- ☐ Device mutual TLS required; required hardware protection level and device attestation enforced on managed fleets.
- ☐ Minimum app versions set and enforced per platform.
- ☐ Classification labels defined; encrypted broadcasts used for sensitive announcements and notices only for non-sensitive content; large-group reaction counts left off.
- ☐ Usage analytics switched off, or locked off at provisioning, where policy requires.
- ☐ Tenant policy: notification previews and sender names off; download to device off; forwarding off; link previews off.
- ☐ App passcode required and wipe-after-failures set; screenshot blocking enabled.
- ☐ Vault disabled if identity keys must never leave devices.
- ☐ Audit export to SIEM enabled; audit chain verified on a schedule.
- ☐ Retention windows set to the minimum consistent with policy.
- ☐ Managed devices enrolled in MDM with SANKET managed configuration.
- ☐ Certificate pinning configured on clients.

Appendix H - Network and Protocol Categories

Category	Protocol	Direction	Notes
Client application traffic	HTTPS / WSS (TLS 1.3 or 1.2)	Inbound to edge	All API, WebSocket, console and signalling traffic
Real-time media	DTLS-SRTP over UDP; TCP fallback	Inbound to media server	AES-256-GCM SRTP
Media relay	TURN over UDP, or TURN over TLS on TCP 443 (TLS 1.3 AES-256-GCM only), routed by server name	Inbound	On by default for internet-facing installations, off for air-gapped ones
Android wake-ups	HTTPS to self-hosted push relay	Device to edge	Sealed payloads
iOS wake-ups (optional)	HTTPS / HTTP/2 to Apple push	Outbound	Connected deployments only
Security events	Syslog over TLS	Outbound to SIEM	RFC 5424 / 5425
Integration events	HTTPS webhooks	Outbound	HMAC-SHA256 signed
Provisioning	HTTPS (SCIM 2.0, Integration API)	Inbound	Scoped credentials
Directory and identity provider (optional)	LDAPS or StartTLS; HTTPS (OIDC, SAML 2.0)	Outbound to the customer's directory and identity provider	Customer-hosted only; public providers refused
Second-site replication (high-availability profile)	Database streaming replication and WAL archive; object-storage site replication; encrypted secrets snapshots	Site to site	Asynchronous; customer-operated link
Administration	SSH via bastion	Management network	Customer-operated
Backups	Customer-chosen transport	Outbound to backup zone	Encrypted archives

Appendix I - Glossary

Term	Definition
ABAC	Attribute-based access control: decisions based on attributes of user, resource and context
Account list key	A per-account libsignal key pair, generated on the primary device and protected by its hardware key, that signs the account's device list
AEAD	Authenticated encryption with associated data: encryption that also guarantees integrity
AES-GCM	AES in Galois/Counter Mode, an AEAD mode; SANKET uses 256-bit keys
Air gap	Physical or logical isolation of a network from the internet and other untrusted networks
Broadcast-sender account	A dedicated SANKET account with a Sanket ID, assigned to named administrators, whose approved desktop sends end-to-end encrypted broadcasts
Content envelope	The versioned structure inside every libsignal message that carries a message, reaction or broadcast with its message id and optional classification label
Crypto-agility	The ability to change cryptographic algorithms and parameters without redesigning the system
DAST	Dynamic application security testing of a running system
Double Ratchet	Signal's algorithm deriving a new key for every message, combining a symmetric ratchet and a Diffie-Hellman ratchet
E2EE	End-to-end encryption: only the communicating endpoints hold the keys
FIDO2	FIDO Alliance standards for phishing-resistant authentication with hardware or platform authenticators
HKDF	HMAC-based extract-and-expand key derivation function (RFC 5869)
HSM	Hardware security module: tamper-resistant hardware for key storage and operations
Key Attestation	Android mechanism by which the hardware key store certifies, in an X.509 chain to a vendor root, that a key was generated in secure hardware
MFA	Multi-factor authentication
ML-KEM	Module-lattice key encapsulation mechanism (FIPS 203), the standardised form of Kyber
mTLS	Mutual TLS: both sides of a TLS connection present certificates
Notice broadcast	A broadcast whose content the server can read, for non-sensitive announcements; labelled as not end-to-end encrypted
PCS	Post-compromise security: recovery of confidentiality after a compromise ends
PFS	Perfect forward secrecy: past sessions stay confidential if long-term or current keys are later compromised
PKI	Public key infrastructure: CAs, certificates and revocation
PQC	Post-quantum cryptography: algorithms believed secure against quantum computers
PQXDH	Signal's post-quantum extension of X3DH, adding a KEM to the initial key agreement
PROXY protocol	A header by which a proxy passes the original client address to the server behind it
RBAC	Role-based access control
SAST	Static application security testing of source code
SBOM	Software bill of materials: inventory of components in a release
Secure Enclave	Apple's hardware security subsystem, whose keys cannot be exported from it

Term	Definition
SIEM	Security information and event management system
Signed device list	An account's list of devices, signed with its account list key and kept in an append-only log, which senders verify before encrypting
SPQR	Sparse Post-Quantum Ratchet, Signal's ML-KEM-based ratchet run alongside the Double Ratchet
StrongBox	An Android hardware key store in a separate secure element, stronger than the TEE
TEE	Trusted execution environment: an isolated area of the main processor that protects keys
TLS	Transport Layer Security
TPM	Trusted Platform Module: hardware root of trust for keys and measured boot
TURN	Traversal Using Relays around NAT: a relay that carries media when a direct path is blocked
WebAuthn	W3C API for public-key credentials used by FIDO2
WORM	Write once, read many: storage that prevents modification after writing
X25519	Diffie-Hellman function over Curve25519 (RFC 7748)
X3DH	Extended Triple Diffie-Hellman, Signal's original asynchronous key agreement
Zero trust	Security model in which no implicit trust is granted by network location or ownership (NIST SP 800-207)

Appendix J - Standards and References

- [1] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3", IETF RFC 8446, August 2018.
- [2] T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", IETF RFC 5246, August 2008.
- [3] Y. Sheffer, P. Saint-Andre, T. Fossati, "Recommendations for Secure Use of TLS and DTLS", IETF RFC 9325 (BCP 195), November 2022.
- [4] E. Rescorla, "TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)", IETF RFC 5289, August 2008.
- [5] A. Langley, M. Hamburg, S. Turner, "Elliptic Curves for Security" (X25519, X448), IETF RFC 7748, January 2016.
- [6] S. Josefsson, I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", IETF RFC 8032, January 2017.
- [7] H. Krawczyk, P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", IETF RFC 5869, May 2010.
- [8] H. Krawczyk, M. Bellare, R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", IETF RFC 2104, February 1997.
- [9] NIST, "Advanced Encryption Standard (AES)", FIPS PUB 197 (updated 2023).
- [10] M. Dworkin, "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", NIST SP 800-38D, November 2007.
- [11] E. Barker, "Recommendation for Key Management: Part 1 - General", NIST SP 800-57 Part 1 Rev. 5, May 2020.
- [12] E. Barker, J. Kelsey, "Recommendation for Random Number Generation Using Deterministic Random Bit Generators", NIST SP 800-90A Rev. 1, June 2015.
- [13] E. Barker, A. Roginsky, "Transitioning the Use of Cryptographic Algorithms and Key Lengths", NIST SP 800-131A Rev. 2, March 2019.
- [14] S. Rose, O. Borchert, S. Mitchell, S. Connelly, "Zero Trust Architecture", NIST SP 800-207, August 2020.
- [15] P. Grassi et al., "Digital Identity Guidelines: Authentication and Lifecycle Management", NIST SP 800-63B.
- [16] K. Kent, M. Souppaya, "Guide to Computer Security Log Management", NIST SP 800-92, September 2006.
- [17] M. Souppaya, K. Scarfone, D. Dodson, "Secure Software Development Framework (SSDF) Version 1.1", NIST SP 800-218, February 2022.
- [18] M. Souppaya, J. Morello, K. Scarfone, "Application Container Security Guide", NIST SP 800-190, September 2017.
- [19] M. Swanson et al., "Contingency Planning Guide for Federal Information Systems", NIST SP 800-34 Rev. 1, May 2010.
- [20] M. Marlinspike, T. Perrin, "The X3DH Key Agreement Protocol", Signal specification, Revision 1, November 2016. signal.org/docs/specifications/x3dh/
- [21] E. Kret, R. Schmidt, "The PQXDH Key Agreement Protocol", Signal specification, Revision 3, May 2023 (updated January 2024). signal.org/docs/specifications/pqxdh/
- [22] T. Perrin, M. Marlinspike, "The Double Ratchet Algorithm", Signal specification. signal.org/docs/specifications/doubleratchet/
- [23] T. Perrin, "The XEdDSA and VEdDSA Signature Schemes", Signal specification, Revision 1, October 2016. signal.org/docs/specifications/xeddsa/

- [24] M. Marlinspike, T. Perrin, "The Sesame Algorithm: Session Management for Asynchronous Message Encryption", Signal specification, April 2017. signal.org/docs/specifications/sesame/
- [25] Signal, "Signal Protocol and Post-Quantum Ratchets" (Sparse Post-Quantum Ratchet, SPQR), signal.org/blog/spqr/, October 2025.
- [26] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, D. Stebila, "A Formal Security Analysis of the Signal Messaging Protocol", IEEE EuroS&P 2017; extended version in Journal of Cryptology 33, 2020.
- [27] K. Bhargavan, C. Jacomme, F. Kiefer, R. Schmidt, "Formal verification of the PQXDH Post-Quantum key agreement protocol for end-to-end secure messaging", USENIX Security 2024.
- [28] Signal Messenger LLC, "libsignal" (Rust implementation with Java, Swift and TypeScript bindings), AGPL-3.0, github.com/signalapp/libsignal.
- [29] Signal Messenger LLC, libsignal source, group cipher (Sender Key) implementation, [github.com/signalapp/libsignal \(rust/protocol/src/group_cipher.rs\)](https://github.com/signalapp/libsignal/tree/master/rust/protocol/src/group_cipher.rs).
- [30] D. M'Raihi et al., "TOTP: Time-Based One-Time Password Algorithm", IETF RFC 6238, May 2011.
- [31] M. Jones, J. Bradley, N. Sakimura, "JSON Web Token (JWT)", IETF RFC 7519, May 2015.
- [32] Y. Sheffer, D. Hardt, M. Jones, "JSON Web Token Best Current Practices", IETF RFC 8725, February 2020.
- [33] T. Lodderstedt et al., "Best Current Practice for OAuth 2.0 Security" (refresh token rotation), IETF RFC 9700, January 2025.
- [34] A. Biryukov, D. Dinu, D. Khovratovich, S. Josefsson, "Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications", IETF RFC 9106, September 2021.
- [35] OWASP Foundation, "Password Storage Cheat Sheet", cheatsheetseries.owasp.org.
- [36] OWASP Foundation, "Mobile Application Security Verification Standard (MASVS)" v2, mas.owasp.org.
- [37] OWASP Foundation, "Pinning Cheat Sheet", cheatsheetseries.owasp.org.
- [38] W3C, "Web Authentication: An API for accessing Public Key Credentials, Level 2", W3C Recommendation, April 2021.
- [39] FIDO Alliance, "Client to Authenticator Protocol (CTAP) 2.1", fidoalliance.org.
- [40] NIST, "Module-Lattice-Based Key-Encapsulation Mechanism Standard" (ML-KEM), FIPS 203, August 2024.
- [41] NIST, "Module-Lattice-Based Digital Signature Standard" (ML-DSA), FIPS 204, August 2024.
- [42] NIST, "Stateless Hash-Based Digital Signature Standard" (SLH-DSA), FIPS 205, August 2024.
- [43] NIST, "Transition to Post-Quantum Cryptography Standards", NIST IR 8547 (Initial Public Draft), November 2024.
- [44] D. Cooper et al., "Internet X.509 Public Key Infrastructure Certificate and CRL Profile", IETF RFC 5280, May 2008.
- [45] S. Santesson et al., "X.509 Internet PKI Online Certificate Status Protocol - OCSP", IETF RFC 6960, June 2013.
- [46] J. Hodges, C. Jackson, A. Barth, "HTTP Strict Transport Security (HSTS)", IETF RFC 6797, November 2012.
- [47] R. Gerhards, "The Syslog Protocol", IETF RFC 5424, March 2009.
- [48] F. Miao, Y. Ma, J. Salowey, "Transport Layer Security (TLS) Transport Mapping for Syslog", IETF RFC 5425, March 2009.
- [49] P. Hunt et al., "System for Cross-domain Identity Management: Core Schema", IETF RFC 7643, September 2015.
- [50] P. Hunt et al., "System for Cross-domain Identity Management: Protocol", IETF RFC 7644, September 2015.
- [51] M. Baugher et al., "The Secure Real-time Transport Protocol (SRTP)", IETF RFC 3711, March 2004.
- [52] D. McGrew, E. Rescorla, "DTLS Extension to Establish Keys for SRTP", IETF RFC 5764, May 2010.

- [53] D. McGrew, K. Igoe, "AES-GCM Authenticated Encryption in SRTP", IETF RFC 7714, December 2015.
- [54] E. Omara et al., "Secure Frame (SFrame): Lightweight Authenticated Encryption for Real-Time Media", IETF RFC 9605, August 2024.
- [55] The MITRE Corporation, "MITRE ATT&CK Enterprise and Mobile Matrices", attack.mitre.org.
- [56] A. Shostack, "Threat Modeling: Designing for Security", Wiley, 2014 (STRIDE methodology).
- [57] ISO/IEC 27001:2022, "Information security, cybersecurity and privacy protection - Information security management systems - Requirements".
- [58] OpenSSF, "Supply-chain Levels for Software Artifacts (SLSA)" v1.0, slsa.dev.
- [59] OWASP Foundation, "CycloneDX Bill of Materials Standard" (ECMA-424); and The Linux Foundation, "SPDX Specification" (ISO/IEC 5962:2021).
- [60] Zetetic LLC, "SQLCipher Design" (AES-256 page encryption with HMAC page authentication), zetetic.net/sqlcipher/design/.
- [61] Apple Inc., "Apple Platform Security Guide" (Secure Enclave, Keychain data protection), support.apple.com/guide/security.
- [62] Android Open Source Project, "Android Keystore system" and "Hardware-backed Keystore", developer.android.com and source.android.com.
- [63] Electron project, "safeStorage" API documentation, electronjs.org/docs/latest/api/safe-storage.
- [64] LiveKit project, "LiveKit open source WebRTC SFU" and "End-to-end encryption" documentation, github.com/livekit/livekit and docs.livekit.io.
- [65] Signal, "Technology preview: Sealed sender for Signal", signal.org/blog/sealed-sender/, October 2018.
- [66] R. Barnes, K. Bhargavan, B. Lipp, C. Wood, "Hybrid Public Key Encryption", IETF RFC 9180, February 2022.
- [67] NIST, "Security and Privacy Controls for Information Systems and Organizations", NIST SP 800-53 Rev. 5, September 2020.
- [68] Center for Internet Security, "CIS Docker Benchmark", cisecurity.org.
- [69] OpenBao project (Linux Foundation), "OpenBao: secrets management, encryption as a service and privileged access management", openbao.org.
- [70] MinIO, Inc., "Server-Side Encryption" documentation, min.io/docs.
- [71] ntfy project, "ntfy - push notifications made easy" (self-hostable pub-sub notification server), docs.ntfy.sh.
- [72] T. Reddy, A. Johnston, P. Matthews, J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", IETF RFC 8656, February 2020.
- [73] W. Tareau, "The PROXY protocol, Versions 1 & 2", HAProxy Technologies specification.
- [74] K. Kwiatkowski, P. Kampanakis, B. Westerbaan, D. Stebila, "Post-quantum hybrid ECDHE-MLKEM Key Agreement for TLSv1.3", IETF Internet-Draft [draft-ietf-tls-ecdhe-mlkem](https://datatracker.ietf.org/draft-ietf-tls-ecdhe-mlkem).
- [75] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, C. Mortimore, "OpenID Connect Core 1.0", OpenID Foundation.
- [76] N. Sakimura, J. Bradley, N. Agarwal, "Proof Key for Code Exchange by OAuth Public Clients", IETF RFC 7636, September 2015.
- [77] OASIS, "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard, March 2005.
- [78] Android Open Source Project, "Verifying hardware-backed key pairs with Key Attestation", Android developer documentation.

[79] J. Samuel, N. Mathewson, J. Capps, R. Dingleline, "Survivable Key Compromise in Software Update Systems", ACM CCS 2010; and "The Update Framework Specification".

Product documentation cited for SANKET components refers to the upstream open-source projects named; their inclusion does not imply endorsement by those projects.