JONE / CLIENT SECURITY RESEARCH

Defensive Reverse Engineering of WhatsApp Android

Message storage, Signal sessions, companion devices, and media lifecycle

SAMPLE EDITION

One complete experiment + selected chapters

JONE Security Research

Independent research | English edition | September 2026

Publication Note

Defensive Reverse Engineering of WhatsApp Android

Message Storage, Signal Sessions, Multi-Device State, and Media Lifecycle

Author: JONE Security Research. English public redacted edition V2.0. Research manuscript dated September 2, 2026; publication package prepared September 3, 2026.

This is a SAMPLE EDITION. It contains selected opening chapters, the laboratory baseline, and one complete experiment (WA-L03). Excerpts retain their original chapter and experiment numbering. The full English edition includes 19 experiment cards.

Independent research

This is an independent publication. It is not affiliated with, sponsored by, endorsed by, or an official manual of the companies or products discussed. Product names identify the subjects of analysis; their trademarks remain with their respective owners.

Evidence and scope

The findings belong to the fixed samples described in the text. Observation, inference, negative evidence, and unresolved questions retain their stated grades. Reproducing an experiment requires an authorized sample and a separate reader record; this edition makes no claim to have retested later versions.

The principal Android sample is 2.26.20.71 / 262007100. crypt15 and end-to-end encrypted backup remain open gaps. A library or symbol does not by itself prove active deployment.

AI assistance disclosure

AI tools assisted with organizing research records, drafting, translation, editing, and publication layout. AI-generated prose is not experimental evidence. Evidence claims must be read together with the fixed-build records, controls, and limitations stated in the manuscript.

Intended use

For authorized defensive research using owned samples, read-only copies, and synthetic data. The publication does not distribute real accounts, credentials, private messages, working decryption or key-recovery tools, authentication bypasses, active-send callers, or bulk extraction tooling.

Navigation Contents

Part I - Evidence Narrative and Architectural Findings

This part preserves the complete fixed-build evidence narrative: sample boundaries, key objects, fields, call chains, state machines, counterevidence, and defensive implications. It answers what the findings are and where their evidence resides. Part II rewrites those findings as experiment cards so a reader can determine how to validate them on an owned sample, what constitutes a pass, and when a claim must be downgraded.

> Reproducibility principle: reproduce methods, structures, criteria, and evidence grades-not third-party accounts, secret material, or active-call capability. Every public lab uses owned samples, read-only copies, and synthetic data.

1. Questions, Version Scope, and Evidence Grades

1.1 Questions this book actually answers

This is not a general statement that “WhatsApp uses end-to-end encryption.” It answers testable implementation questions:

- Which logical databases hold messages, identities, sessions, device state, and synchronized collections in a fixed Android build?

- Are the examined business databases SQLCipher containers? Does end-to-end encryption imply local database encryption?

- How can the fields, wire numbers, lengths, and state roles of `SessionRecord`, `Chain`, and `ChainKey` be recovered and validated?

- After two controlled text sends, how do message rows, sender-chain indices, jobs, and network log anchors change together?

- After one controlled receive, does the matching receiver chain advance by one in the isolated sample?

- Does a companion device use an independent identity and pairwise sessions? What is deleted or retained after unlinking it?

- How are text and image objects built across the obfuscated Java layer, and why are media upload and final message transmission separate stages?

- What do Curve25519, KEM, mvfst, and Fizz symbols prove-and what deployment claims do they not prove?

1.2 Exact sample scope

Exact names in this book are anchored to **2.26.20.71 / 262007100**. One complete APK sample was carried through the full workflow. Consequently:

- protocol semantics, object relationships, field combinations, and state machines tend to be more stable than obfuscated names;

- names such as `X.375`, `X.1Tt`, and `X.7wO` must not be copied into another build without revalidation;

- no statement in this edition means “all historical and future WhatsApp versions”;

- porting the analysis should begin with stable logs, superclass relations, constructor shapes, and field groups rather than old short names alone.

1.3 Five evidence grades

Grade	Meaning	Typical evidence in this book
E1: direct static	Visible in decompilation, exports, strings, or layouts	inheritance, JNI export, wire number
E2: offline structural	An authorized snapshot parses consistently	SQLite open, protobuf-style decoding
E3: controlled experiment	A single-variable action produces a repeatable diff	two sends, sender index +2
E4: primary-spec mapping	A first-party specification explains an observed role	X3DH, Double Ratchet, ML-KEM
E5: inference/open	Naming and structure suggest a use, but no dynamic closure exists	the product use of VOPRF

A claim inherits the boundary of its weakest evidence. “KEM JNI exports exist” is E1. “This build enables a specific post-quantum suite for every user” cannot be promoted from E1 without session-level E3 evidence.

1.4 Publication boundary

The public edition may include a fixed version, exact class and function names, parameter roles, wire numbers, lengths, state transitions, synthetic examples, and observation-only methods. It does not include live keys, `directPath` values, media keys, live hashes, conversation data, production endpoints, device-private extraction paths, executable extraction/decryption scripts, or callable send hooks.

2. Reproducible Experiment Design

2.1 Authorization and isolation

The work used researcher-controlled devices, test accounts, and test counterparts. Four modes were kept separate:

1. **Static analysis:** APK, DEX, ELF, strings, and exports; no business action.

2. **Offline parsing:** read-only analysis of authorized snapshot copies.

3. **Controlled experiment:** one countable variable at a time, such as sending two texts or unlinking one companion.

4. **Specification mapping:** primary documentation was used only to interpret state already observed in the product.

The observation pipeline was not converted into bulk-sending, key-recovery, or content-extraction tooling.

2.2 Baseline-action-result model

Phase	Requirement
Baseline B0	Record APK version, account/device roles, database summaries, row counts, and target state
Action A1	Perform one countable, reversible, observable action
Result B1	Capture the same evidence surfaces and a narrow log window
Difference D	Classify additions, updates, deletions, and invariants
Cross-check	Require agreement between at least two of UI, database, serialized state, and logs

When sending two texts, no device was linked or unlinked, no media upload was triggered, and no version changed. That isolation makes the chain-index difference attributable to the two successful sends.

2.3 Experiment matrix

ID	Single-variable action	Primary surfaces	Pass condition
T-DB-01	No business action; copy a snapshot	header, offline SQLite	read-only open and stable schema
T-DR-01	Send two text messages	message rows, sender chain	target index advances by two
T-DR-02	Receive one text message	message rows, receiver chain	target index advances by one
T-MD-01	Link one companion	identity, session, device rows	independent device identity/session appears
T-MD-02	Unlink that companion	device/session/sync state	access state removed; selected sync state retained
T-MEDIA-01	Send one synthetic image	upload, ImageMessage, ACK	complete reference appears after upload

2.4 Failure criteria

A result was not marked verified when:

- only a symbol or string existed, with no state or invocation evidence;

- before/after snapshots came from different versions or device roles;

- a log window contained unrelated concurrent actions;

- a current specification’s new feature was projected backward into the fixed build;

- a one-sample count was described as a universal strict 1:1 invariant.

Part II - Full Experimental Validation Manual

Laboratory Baseline

Create a separate directory for every experiment and begin with a `sample-card`, `toolchain-card`, and `experiment-log`. The sample card records app version, versionCode, ABI, SHA-256, acquisition date, and authorization scope. The toolchain card records versions of decompilers, signature parsers, database tools, and scripts. The experiment log records start/end time, unique lab id, actions, observations, anomalies, and hashes of raw evidence. No live account, device, contact, conversation, message, media, credential, key, endpoint, or exact address may enter the public manuscript.

Common Isolation Conditions

1. Use only test accounts, devices, and application artifacts that you own or are explicitly authorized to study.

2. Preserve originals read-only; analyze copies and recompute hashes before and after each lab.

3. Dynamic observation is read-only and pass-through: never alter returns, bypass checks, or act on behalf of third parties.

4. Network-facing tests use minimal synthetic messages; structural work that can be completed offline remains offline.

5. Every causal claim requires at least one negative control; correlation without a control is not promoted to causation.

6. For a new build, remap tags, fields, states, and outcome anchors before using any old obfuscated name or address.

Record Format

Each card has fourteen elements: research question; sample/build; environment and prerequisites; static anchors; inputs; structure; outputs and side effects; analysis path; positive procedure; expected and observed evidence; negative controls; failures and corrections; version/evidence/publication boundary; and defensive meaning with pass/fail/inconclusive criteria. “Observed result” in this book belongs to the author's fixed-build evidence; readers must record their own result separately rather than overwrite it.

Experiment 03 | WA-L03 | Offline Stop, Timeline Correlation, and UI Reconciliation

1. Research Question

Can database rows, UI state, and experimental actions be matched one-to-one while excluding background synchronization?

2. Sample, Build, and Evidence Scope

Lab id: `WA-L03`; evidence mapping: `WA-05`; fixed sample: `WhatsApp Android 2.26.20.71`. This lab makes claims only for the listed sample.

3. Environment, Authorization, and Prerequisites

Follow the laboratory baseline in this part. Keep originals read-only, pin tool versions, restrict dynamic observation to owned accounts with pass-through behavior, and publish synthetic values only.

4. Static Locators and Evidence Anchors

Experiment ID, monotonic and wall clocks, direction, type, state, database key/time fields, UI capture time, and network state.

5. Inputs, Provenance, and Constraints

Three self-created texts under online, airplane-mode, and restored-network conditions; each uses a unique phrase represented only by a hash publicly.

6. Data, Protocol, or Object Structure

The event table contains action_time, db_first_seen, ui_state, network_state, row_hash, and interpretation; clock tolerance is declared in advance.

7. Outputs, State Changes, and Side Effects

Each action explains one set of new/changed rows; offline state is not mistaken for server confirmation; the recovered update returns to the same logical message.

8. Analysis Path and Call Chain

Baseline snapshot → unique action → immediate stop → consistent copy → database diff → UI/timeline review.

9. Positive Validation Procedure

1) Synchronize lab clocks; 2) establish a blank baseline; 3) create one online message; 4) stop and copy; 5) repeat in airplane mode; 6) restore network and wait for one state transition; 7) diff new/changed rows each time; 8) reconcile by direction, time, and hash.

10. Author Observation and Reader Record

Expected/observed pattern in the author's fixed build: Each action explains one set of new/changed rows; offline state is not mistaken for server confirmation; the recovered update returns to the same logical message. Readers create a separate result column containing actual values, timing window, raw-evidence hash, and any divergence.

11. Negative and Control Experiments

A blank window must add no target row; repeat with background sync disabled; changing the unique phrase must change its content hash.

12. Failed Hypotheses, Anomalies, and Corrections

Second-level time or row count alone can misassociate records; direction, type, hash, and state must be combined.

13. Version, Evidence Grade, and Publication Boundary

Evidence E2+E3; publication B. No third-party messaging and no bodies or phone numbers.

14. Defensive Meaning and Pass/Fail Criteria

Pass: the three-condition timeline repeats with no orphan rows. Fail: multiple actions share one window. Inconclusive: background activity was not isolated.

About the Full Edition

The full edition of **Defensive Reverse Engineering of WhatsApp Android**, by JONE Security Research, contains the complete evidence narrative and 19 experiment cards. Each experiment includes scope, inputs, structures, positive validation, controls, failed hypotheses, version boundaries, and pass/fail criteria.

Find the full edition on Leanpub by its exact title and author. This sample preserves original chapter and experiment identifiers; the navigation contents above list only the included excerpts.

The principal Android sample is 2.26.20.71 / 262007100. crypt15 and end-to-end encrypted backup remain open gaps. A library or symbol does not by itself prove active deployment.