JONE / CLIENT SECURITY RESEARCH

Defensive Reverse Engineering of Douyin Android

APK integrity, local storage, IM protocols, and media pipelines

SAMPLE EDITION

One complete experiment + selected chapters

JONE Security Research

Independent research | English edition | September 2026

Publication Note

Defensive Reverse Engineering of Douyin Android

APK Integrity, Storage, Messaging, and Evidence-Based Validation

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 (DY-L02). Excerpts retain their original chapter and experiment numbering. The full English edition includes 18 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.

Douyin live-streaming is a future research plan, not a validated result. Authentication and identifier chapters preserve analysis boundaries and do not provide a reusable bypass or identifier generator.

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.

How to Read This Book and Its Boundaries

This book is not a Douyin API manual. It does not provide message injection, bulk collection, account takeover, signature forgery, authentication replay, or a one-click database decryptor. It documents defensive research performed on owned devices, owned accounts, and offline samples: how the client is layered, how functions and fields were located, which claims survived runtime or offline validation, which remain strong inferences, and which anchors persisted across builds.

The public edition retains exact sample versions, reviewable function roles, selected version-bound names, field structures, command numbers, state transitions, and validation methods. It removes live user and conversation identifiers, tickets, tokens, certificate contents, endpoints, media URLs, key material, password formulas, machine paths, and exact native addresses. All examples are synthetic and cannot address a live service.

Two adjacent Android builds define the evidence boundary. Version 38.8.0 provides the main IM-storage and text-message evidence. Version 38.9.0 provides APK-signing evidence, the image pipeline, and an independent cross-version check of the lower text path. The requested livestream directory contains research requirements but no function table, field table, runtime hit, capture, or final

report. Private-message findings are therefore not projected onto livestream behavior.

> Central rule: version boundaries, evidence grades, and refutations have the same importance as successful findings. A precise statement of what remains unknown is part of a trustworthy result.

Chapter 1 - Questions, Samples, and Evidence Grades

1.1 Research questions

The work asks six testable questions. How does the APK establish update identity? How is private-message data divided among local stores? How does local password material reach WCDB/SQLCipher? How do text and image objects become Wire requests? Which fields and states define device-bound authentication? Can public pseudonymous identifiers be reconstructed from client-side material?

These questions cross APK metadata, Java/Kotlin code, databases, protobuf, native libraries, and runtime state. Static decompilation recovers objects and call relationships; runtime observation confirms active parameters; database differencing establishes persistence semantics; offline recomputation rules out accidental success. No single technique is treated as sufficient.

1.2 Sample matrix

Sample	Primary purpose	Directly supported findings	Not covered
Android 38.8.0	IM databases, text pipeline, Wire fields	Three storage layers, SQLCipher v3 parameters, text content, command 100, 26-field request body	Main signing-lineage evidence
Android 38.9.0 / versionCode 380901	APK signing, image path, text cross-check	v1/v2/v3, proof-of-rotation, media upload, stable Wire tags	Livestream implementation

1.3 Evidence grades

Grade	Meaning	Use in this book
E8	Runtime hit, state/result update, and a second path agree	Highest-confidence behavior
E7	Static chain and runtime parameters corroborate each other	Function and field roles
E6	Static, database, or controlled-variable evidence is strong	A missing link remains explicit
E5	Two builds preserve structure or tags	Cross-version stability
E3	Sample statistics or database/UI comparison	Never generalized to all users
H	Candidate mechanism or cipher family	Hypothesis only
R	Earlier claim disproved by later evidence	Retained in the correction ledger

Core claims map to DY-01 through DY-32 in Appendix A. “Verified,” “strong inference,” and “hypothesis” are deliberately non-interchangeable terms.

Chapter 2 - Establishing a Trusted APK Sample

2.1 Sample identity

The 38.9.0 sample uses package `com.ss.android.ugc.aweme` and versionCode 380901. The workflow records the original file hash, package metadata, signature-verification output, and tool versions before decompilation. A read-only working copy is then created. Without this step, a modified or re-aligned APK can invalidate every later claim about classes, certificates, or behavior.

2.2 Three signing schemes

Offline inspection confirms JAR v1 and APK Signature Schemes v2 and v3. The v1 material lives in traditional archive metadata, whereas v2/v3 live in the APK Signing Block. Their coexistence provides compatibility across Android verification generations; it does not mean three unrelated publishers signed the application.

Android's official `apksigner` documentation treats verification as validation across the Android releases supported by the APK and provides a separate certificate-lineage operation. Two independent parsing paths were used so that inspecting `META-INF` alone could not hide v2/v3 evidence.

2.3 Certificate rotation

The sample contains two self-signed RSA-2048 certificates connected by a v3 proof-of-rotation from the older to the newer signer. Older platforms can retain the v1-compatible identity while newer platforms understand the lineage. Full fingerprints are not necessary for the public proof; algorithm, certificate count, lineage direction, and cross-parser agreement are sufficient.

Observation	Evidence	Interpretation
v1, v2, and v3 structures validate	Two parsing paths	Multi-generation compatibility
Two self-signed RSA-2048 certificates	X.509 properties	Publisher-managed trust roots
A v3 rotation record exists	Proof-of-rotation lineage	Continuous update identity
v1 remains associated with the older signer	Legacy path	Not a signature conflict

2.4 Defensive implication

Enterprise allow-listing should not hard-code only the newest certificate fingerprint. A stronger artifact ledger records the authorized lineage, minimum platform behavior, package name, version, and file hash. Rotation is expected; an unauthorized break in the lineage is the anomaly.

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 02 | DY-L02 | Recover the Three-Layer Storage Topology with Minimal-Action Diffs

1. Research Question

Are key-value state, person-to-person messages, and feature/search data stored in separate layers rather than one database?

2. Sample, Build, and Evidence Scope

Lab id: DY-L02; evidence mapping: DY-04; fixed sample: Douyin Android 38.8.0. 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

Keva references and file magic; WCDB/SQLCipher open chain; Room entities/DAOs; FTS4 schema; before/after file mtime, size, and page diffs.

5. Inputs, Provenance, and Constraints

A clean test account, offline snapshot directory, and three minimal actions: toggle one setting, send one synthetic text, and perform one local search. Execute only one action per run.

6. Data, Protocol, or Object Structure

The diff ledger records file role, container type, changed pages/keys, write time, caller, and UI result. Keep unknown files labeled unknown; never infer semantics from names alone.

7. Outputs, State Changes, and Side Effects

Settings primarily affect the key-value layer, person-to-person messages affect encrypted IM stores, and search/feature actions affect Room/FTS objects. One action may touch several layers, but primary responsibilities remain separable.

8. Analysis Path and Call Chain

Single user action → filesystem diff → container identification → code reader/writer → table/key diff → UI confirmation.

9. Positive Validation Procedure

1) Force-stop and capture a baseline; 2) launch and perform one action; 3) stop immediately and capture again; 4) filter changed files by hash/size; 5) identify magic/schema; 6) trace writers; 7) repeat three rounds; 8) remove one-off noise from the fact table.

10. Author Observation and Reader Record

Expected/observed pattern in the author's fixed build: Settings primarily affect the key-value layer, person-to-person messages affect encrypted IM stores, and search/feature actions affect Room/FTS objects. One action may touch several layers, but primary responsibilities remain separable. Readers create a separate result column containing actual values, timing window, raw-evidence hash, and any divergence.

11. Negative and Control Experiments

Control A: launch with no action to measure background noise. Control B: infer from filenames without callers and mark insufficient. Control C: inspect one database only and fail to explain all UI state.

12. Failed Hypotheses, Anomalies, and Corrections

Background work contaminates mtimes and table names may be indexes only. Correct with short windows, three repetitions, caller tracing, and UI alignment.

13. Version, Evidence Grade, and Publication Boundary

Do not publish live directories, account bucket names, or message bodies. Fixed-build evidence E1+E3; publication B.

14. Defensive Meaning and Pass/Fail Criteria

Pass: each action has a repeatable primary layer with matching callers. Fail: filename/mtime only. Inconclusive: background noise cannot be isolated. Use the result for layered forensics, logout cleanup, and sensitive-index minimization.

About the Full Edition

The full edition of **Defensive Reverse Engineering of Douyin Android**, by JONE Security Research, contains the complete evidence narrative and 18 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.

Douyin live-streaming is a future research plan, not a validated result. Authentication and identifier chapters preserve analysis boundaries and do not provide a reusable bypass or identifier generator.