

Designing and Developing an Open-Source Forensic Framework for Cross-Application Analysis of Encrypted Messaging Artifacts on Android.A Study of Signal and Session

Mustapha Ibrahim Bello¹, Akash Kumar Singh¹, Dr. Umema Ahmed¹,
Al'amin Dahiru¹, Aisha Rabiuh Ibrahim¹

¹Department of Forensic Science

Vivekananda Global University, Jaipur, India

Corresponding author: 24wbsc3fs004@vgu.ac.in

Abstract: *Background: Android messaging artifacts vary with acquisition conditions, storage structures, application versions, and examination methods. Mobile-forensics guidance emphasizes preservation, documentation, examination, and reporting [1], [2]. Objective: This study presents an open-source framework for traceable, cross-application analysis of Signal and Session artifacts on Android, separating file inventory from message interpretation. Methods: We assessed retained implementation records, schema outputs, archive-inventory reconciliation, and historical regression records. The evaluation does not include independent ground-truth validation of production parsers [3], [4]. Results: The retained Session workflow processed a working archive copy and produced 22 file-inventory records, zero message-event records, and no parsing attempt. A historical test suite reported 20 tests with no failures. The SQLite interface did not recognize the examined database files, so their formats remain undetermined. Conclusion: The evidence supports bounded file-level ingestion and reporting, not production message recovery or parser accuracy. Production parsers, controlled ground-truth experiments, and cross-version validation remain necessary.*

Keywords: Android forensics, encrypted messaging, Signal, Session, forensic artifacts, open-source tools..

I. INTRODUCTION

Communication related artefacts such as local files, media, metadata, configuration and application state can be kept in mobile devices. The evidential value of these is not only based on the possibility of extracting a file from the device, but also on the tool used to acquire the device, the information captured/recorded, and the interpretation of observations. The preservation, acquisition, examination, analysis and reporting of mobile devices are all linked but discrete steps [1], [5]. Digital evidence standards also focus on controlled identification, collection and acquisition, preservation and documentary handling of evidence [4], [6], [7], [8].

Messaging apps are more complex when they are encrypted. Applications can vary in terms of their structure of storing the files, how they interpret databases, cryptographic design, local retention and implementation of each version. However, studies of Android messaging apps, including WhatsApp, Telegram, Chat Secure and various social-messaging apps, demonstrate the need to test and interpret artifacts for each specific app, and not assume they are the same across all apps [9], [10]. The study of encrypted messengers also demonstrates how prerequisites for application-specific examination can be specialized and carefully written to be more suitable to the task at hand [2].



This study examines the design and implementation of a forensic framework for organizing and documenting Android artifacts associated with Signal and Session. It focuses on evidence provenance, file-level inventory, processing status, and structured reporting rather than claiming successful recovery of production messages. The distinction between file ingestion and message interpretation is central to the framework's design.

The paper neither states that the content of the production messages has been successfully parsed for either of these applications. Rather, it's about the supporting workflow required to maintain evidence provenance, to establish by reference if necessary, to prove the identity of the byte level file, to inventory files, to document the processing state of the file, and to report results without assuming that the file has not been processed. The primary design philosophy is the abdication of the role of ingestion from the role of interpretation. Even if an internal format of a file is not known, a file can still be identified, measured and hashed. A file-inventory record would not indicate that a message was recovered. It's similar to the way evidence is handled and the importance of identifying and evaluating the characteristics of forensic software with specific requirements [3], [4].

This paper makes four key contributions to digital forensics:

Modular framework architecture: A modular design that separates acquisition documentation, evidence integrity verification, file inventory, application-specific processing, normalization, correlation, and reporting.

Provenance-aware evidence model: An evidence model that records source identifiers, file paths, cryptographic hashes, and processing states to support traceability and auditability.

Separation of file inventory and message interpretation: A versioned schema that distinguishes file-level observations from interpreted message-event records, preventing the existence of a file from being misrepresented as successful message recovery.

Evidence-bounded evaluation: An evaluation approach that reports retained implementation and regression-test evidence while explicitly distinguishing software verification from forensic validation.

These contributions concern the framework's architecture, evidence representation, and evaluation approach. They do not establish successful production-message recovery, parser accuracy, complete acquisition, deleted-message recovery, or superiority over other forensic tools.

A. Research questions

RQ1: What is the possibility of a modular open source framework to structure the acquisition records, integrity verification and file inventory for Android messaging applications?

RQ2: How to maintain the provenance of the application but still keep the file-level observations and the application-specific events related to the interpretation of the messages?

RQ3: What is the evidence for the implementation (and regression) of the behaviour of the components of the processing of the framework?

RQ4: What is not validated, what is needed for production forensic claims to be supported?

II. RELATED WORK

A. Investigating the evolution of Android devices Mobile-device forensic procedures

As outlined in NIST SP 800-101 Rev. 1, range from validation, preservation, acquisition, examination, analysis and reporting. The ISO/IEC 27037 focuses on the identification, collection, acquisition and preservation of potential digital evidence [6]. SWGDE offers recommendations for mobile-device evidence collection and preservation, mobile-device analysis and overall digital-evidence collection [5], [7], [8]. These documents provide information on requirements of documentation, traceability and controlled handling. They do not certify this implementation, or suggest that software is in itself an acquisition forensically sound.

Guidance on forensic methods and analysis focuses on the use of appropriate methods and providing results in ways that enable understanding and critique of the methods used [3], [4], [11]. The framework therefore records processing



state and limitations as part of the output rather than treating a successful script exit as sufficient evidence of forensic correctness.

B. Application-specific messaging forensics

Recent work on a set of social-messaging applications for Android has revealed that there are different classes of traces left by different applications and the availability of content is different across applications and situations [9]. Following the ideas of scenario driven artifact generation and validation, Anglano and colleagues developed experimental methods to analyse WhatsApp and Telegram on Android [10], [12]. Their ChatSecure study involved controlled experiments, and they did compare the results of the emulators with the results of the real devices [13]. These methods have repeatability as a design goal, but the results of these methods cannot be directly applied to Signal or Session. Son et al. looked at encrypted instant messaging and discussed application-specific approaches to file analysis of encrypted files [2]. This literature provides motivation for specialized adapters, but the specific decryption techniques of this implementation are not the same as those used by the specialized adapters. It provides common pieces of shared evidence-handling code that can have different parsers and different validations specified and validated separately.

C. The software testing and reproducibility aspects of Forensic Science

The function of a tool is distinguished from its behaviour under a certain test procedure in forensic-tool testing. The work NIST does on computer forensics testing identifies requirements, test assertions and test data as the basis for characterizing the capabilities of the tools [3]. Unit tests can detect regressions and deal poorly with malformed inputs but are not a proof of correct interpretation of a real application's data by a parser. To validate an event, it is necessary to have individual and documented expected results, representative input values, and a consistent method for comparison [3], [14]. Controlled user actions and repeatable device states are very prevalent in application-forensics research, enabling the observation of artifacts and the linking of these to known actions [12], [13]. The current scheme is based on that principle, with current evidence based on retained file-inventory and software-test evidence.

D. The research gap and the positioning of the research work are presented as follows:

The engineering challenge is to keep the common provenance and processing semantics across applications without attempting to interpret the internal artifacts in a uniform way as this can be misleading. The framework is conceived as a complementary implementation of a workflow and schema to support application specific analysis. A full comparison with all other forensic frameworks that exist is not available based on the evidence provided and it is not the aim of this paper to provide the best recovery performance or be the first to provide a framework of this type.

III. REQUIREMENTS AND DESIGN PRINCIPLES

The design requirements derive evidence-handling guidance and application-specific research into explicit software behaviours that allow for auditability and proportionateness for outputs generated by the software based on what it does [2], [4]- [5], [13].

A. Evidence preservation

Acquired evidence should be considered a preserved source, and extraction, parsing and normalisation should be done on a copy that is not the preserved evidence, wherever possible. A derived file should be different from the source object, the processing record should refer to the input archive, output location and operation performed. The separation will minimize the possibility of a confusion or mix up with analysis derivative and will facilitate a review of the processing sequence at a later time [5], [13]. A copy of the source is required but not sufficient. It is not a sufficient proof by itself that the first acquisition was complete, that a live acquisition was a coherent point in time acquisition or that all relevant files were available for acquisition by a method. The framework thus makes a distinction between preservation and acquisition claims and does not assume that archive readability is the same as the validity of the archive's acquisition.



B. Integrity verification

The framework should compute cryptographic digests of the evidence objects of interest, and it should output them. If there is a trusted reference digest, the observed digest should be compared to the trusted digest and the result of the comparison stored in parallel with the observed digest and the trusted digest. A digest comparison cannot conclude that two compared byte sequences are equal under selected algorithm/selected procedure, but it can conclude that they are equal under the selected algorithm and procedure.

The design should make it possible to tell whether a comparison has previously succeeded or not, but not be a straightforward Boolean field containing a result where it would be merely an indicator of a successful comparison. Only a boolean is retained, which means that the value will be recomputed independently. Non-existent, invalid or duplicated manifest values should be reported rather than assumed to be a successful validation. The algorithm to be used and how to implement it should be documented and hash results should be distinguished from parser results.

C. Provenance and traceability

Enough metadata should be kept for each artifact and derived record to be able to trace it back to the source and processing context. Fields recommended are case identifier, application/identifier, application version, device/operating system context, source archive, relative path, file size, file digest algorithm and value, acquisition time (if known), processing operation, parser status and parser schema version. It's important to have fields drawn directly from the versioned implementation schema, not from examples. Normalized outputs should be accompanied by a provenance. Exports, merges or correlations of a record should have the source reference available, so an examiner can go to the record source and repeat the exercise. This is consistent with the evidence handling standards of documentation and accountability [5], [13]. The metadata pertaining to the provenance is not trivial information to be shared with the public but should be considered prior to publication for personal or environment-specific information that is not necessary.

D. Separation of inventory and interpretation

The schema should differentiate between file characteristics that can be observed from those about what is in the file. The file_inventory record indicates the existence and attributes of a file, a message_event record is a communication event uncovered through a defined parser with evidence. Must not generate an invented message record for the existence of a database-like file(s), and an empty event output must not be construed as an indication that no messages were present. The processing state should be explicitly stated (e.g. not_attempted, attempted_failed, parsed_with_warnings or parsed_successfully) and should reflect the actual state of the processing, and be consistently applied, with states being formally defined. Machine readable rules and logs should back status labels. This will avoid reporting success on a file level as a successful message recovery.

E. Cross-application normalization

Normalization should normalize fields of which the meaning is truly comparable (e.g., a digest of the source file, processing timestamp) and leave control fields and raw values as they are, in the application. It should not be assumed that labels with the same name mean the same thing in both Signal and Session. All mapping must include the source field, target field, transformation, units, null handling and uncertainty. Original application and artifact identifiers should be maintained on a normalized record. If there is mapping that is ambiguous or not supported, the framework should leave the original value or the normalized field be blanked out, rather than assume it means 'equivalent'. This allows the possibility for cross-applied queries without obfuscating differences that are important to the forensic interpretation.



F. Reproducible processing

Software revision, version of the software dependencies, configuration, command used, identifiers of the input and output, and complete test results should be preserved along the workflow wherever possible. The results of identical inputs should be consistent if repeated processing occurs and the fields that are nondeterministic should be documented. Reproducibility is not automatically a property of the specified procedure and environment, but it is a property of the specified procedure and environment[3], [12], [13]. A reproducibility package should allow the reproduction by a reviewer of the input and version that led to a result. It should not involve revealing any sensitive information for no reason and a successful packaging/regression audit should not be called forensic validation. The framework should be able to detect errors and report them safely and clearly. For example, malformed manifests, unreadable archives, duplicate entries, missing files, and digest mismatches should produce explicit error messages or processing statuses. Invalid data should not be silently accepted or represented as valid evidence. Reports should distinguish confirmed observations, processing errors, unresolved questions, and interpretations.

IV. FRAMEWORK ARCHITECTURE***A. Architectural overview***

The logical architecture has five separate roles:

- (1) acquisition documentation,
- (2) integrity verification and inventory,
- (3) application-specific examination,
- (4) common-schema representation and normalization, and
- (5) correlation and reporting.

This decomposition allows for the ability to test shared functions without any application specific parser, and have the parser be unused and not prevent file level inventory. This is only a description of a logical design, and does not mean that every module named is actually implemented. The maturity of the modules should be determined from the source code, tests and execution logs. The capabilities should be categorized as implemented and tested, implemented but inadequately tested, planned or not implemented.

B. Documentation and evidence acquisition layer

The acquisition layer documents the evidence made available to the framework and records relevant contextual information, including the acquisition method, source identifiers, application details, and integrity results. The types of fields can vary based on the acquisition method and include device and operating system information, application identifier and version, source path, acquisition time, archive properties, manifest reference, tool/version information and integrity results. When there is a lack of metadata then that should be noted as unknown, not assumed. Examine the examined applications and archive-based handling of the project records is provided. The completeness of the acquisitions and temporal consistency of the archives must be evaluated separately from the extraction process from the archives. Readable archive and successful inventory reconciliation only confirm properties of the archive and of the comparison that has been read and reconciled. The integrity and inventory layer is responsible for maintaining evidence integrity, recording file metadata, and preserving the traceability of forensic artifacts.

C. Integrity and inventory layer

This layer records the file path, sizes, cryptographic digests that are available, and compares inventories and discrepancies between these. A good reconciliation should report on separately the following items: missing, additional, duplicate, path mismatch, size mismatch, digest mismatch, unreadable objects. If the file is found, it should be clearly distinguishable from the file which is not acquired or cannot be read. Keeping track of the contents of a file even if it is not known what it contains gives a clear idea of the contents that were available for examination. It does not provide any semantic interpretation for the file nor does it provide proof that all artifacts were acquired that are relevant for this file.



D. Application-specific adapters

Adapters are responsible for transforming the artifact identification and parsing of the application. Supported versions of applications, input structures, output contract, error structures, dependency and provenance fields should be specified for each adapter. It should record if it was attempted to be parsed, failed, completed with warning(s) or completed within any specified validation criteria. Having a module (or adapter interface) doesn't mean that a production parser actually works. For the retained Session workflow, functionality demonstrated was archive ingestion (copy only) and file-level inventory. There is no validated evidence in place for a production Session parser or an evidence that can be validated as a production Signal message parser.

E. Data is stored in the normalized form

In schema version 2.0.0 the file_inventory records are split from the message_event records. This way, the system can simulate observed files, but doesn't suggest that they have been decrypted. There are 22 file_inventory records and 0 message_event records in the retained Session output and the parser status is not_attempted. Field names, types, required/optional, enumerations, validation rules and versioning policy should be included in the schema definition which should be included with the implementation package. Backward-incompatible changes should be done in a documented manner, following a versioning policy. Future mappings of message-events to artifacts must be explicitly defined by parser rules and be traceable to source artifacts.

F. Correlation between layer and layer

Correlation attempts to link records together across various files or processing phases, and retain the source relationships and purposes of these relationships. If there are no other interpretations found then a match between file names, timestamps or metadata should be reported as a correlation candidate. Correlation cannot be used to establish a communication event. Reports should include a summary of the evidence analysed, operations conducted, versions of software/schema used and integrity results, differences in the inventory and any processing errors or unmet limitations. They need to be able to tell the difference between what they see happening and what they interpret it to be and should never make the leap into believing that an event didn't happen because they can't find an observation to back it up.

V. IMPLEMENTATION***A. Development Approach and Project Organization***

This refers to the way in which a development is approached and organized. The method and structure of a project development. The framework was developed in an iterative manner in the following phases: evidence handling, operations on the Android, inventory of artifacts, checking integrity, normalization, correlation, validation and reporting. The project organization comprises of modules related to the acquisition, Android handling, artifacts, audit, case management, core processing, correlation, evidence, filesystem operations, findings, graphical interfaces, integrity and reporting. These names are only documentation of the project organization; they do not imply that each of the functions is fulfilled. Separation of evidence-handling functions that can be reused from the creation of the application-specific evidence parser frees up common processing for later, when the format for the application's evidence can be resolved.

B. Session archive ingestion

The Session workflow that is retained works with a working copy of acquired TAR archive. It pulls files, matches it against archived files and converts file metadata into the standard schema. The Project records list an archive of 22 inventoried regular files, which were reportedly readable and included the relative paths and the SHA-256 values of the files within the archive. This means that this is bounded file-level ingestion for the archive(s) being examined. It does



not show that all relevant application files were acquired, that the files are not a snapshot of a point in time nor that the contents of any databases were interpreted.

C. Common schema implementation

Version 2.0.0 of the retained schema is now in place and the file_inventory has been moved out of message_event. A record of a file and its observed metadata is a file_inventory record, while a message_event record is used for only information regarding the communication produced by an appropriate parser. The Session output consists of 22 file inventory records, 0 message event records and parsing was not attempted. The outcome is meaningful as it truly reflects the extent of processing. It does not enable an inventory-only workflow to be viewed as a message-recovery pipeline. Any constraints and definitions of the fields in the fields section should be included in the versioned schema and be part of the release package. The implementation of Integrity and reconciliation

D. Historical regression cases

The historical regression cases include malformed hash and size fields, missing/archive or duplicate manifest entries, and unreadable TAR archives or a mismatch between a Boolean comparison field and an acquisition digest. These cases illustrate the importance of having manifest semantics, aside from file hashing. A boolean that is established once the two values match cannot be used instead of the actual digest. The implementation should preserve values of references, observed values and results of comparison. Unmatched data should be identified as such and a summary of the reconciliation should not indicate an entire acquisition was verified just because some of the portions matched. The paper presents the reported results as historical data in the expectation that they can be repeated independently.

E. Database characterization

In the case of the Signal and Session database files, the standard SQLite interface that was used during checks was not able to recognize the examined files, during the characterization. This observation is only for the interface and inputs that were used. It doesn't create encryption, a proprietary format, a wrapper, a transformed database or anything else that could provide another specific method of storage. The framework therefore doesn't report database records that have been recovered or deduce the meaning of database semantics from failures to open sqlite. Before it's possible to draw conclusions at the message level, additional non-destructive format investigation is necessary as well as a documented parser contract and validation against independent expected outcomes.

VI. EVALUATION METHODOLOGY

A. Evaluation scope

The evaluation is split into two parts: implementation verification and forensic validation. Software verification: Do a defined piece of software act as expected on given inputs? Another question in forensic validation is if the method yields correct interpretations under controlled conditions where there is a known ground truth [4], [8], [11], [14]. The existing assessment relies only on implementation records that are retained and evidence of file level reconciliation and historical regression.

B. File-inventory reconciliation

The retained Session inventory was compared to files which were extracted to a separate working directory. To determine if there was an extracted file with the same digest as each expected file, both relative paths and the SHA-256 were used to verify this. According to project records 22 regular-file paths and digests were extracted and there was a corresponding number of preserved ones. This helps to identify files being compared in the retained comparison. It does not guarantee completeness of the acquisition, temporal consistency of the interpretation of the message nor the validity of the interpretation. In the final reproduction package, the exact archive, manifest and command as well as complete output should be provided.



C. Regression testing

The retained suite features synthetic cases to handle archive operations, manifest reconciliation, missing or duplicate rows, malformed hashes and sizes, unreadable TAR files, database-header observations and paths with spaces. 20 tests were run and no failures were reported, errors or exit code were 0. This test was not replicated separately for the preparation of this manuscript. It is not a new experiment, but is reported as historical test evidence. A full passing of the unit-test suite would not predict the precision, recall, completeness of message-recovery, and false-positive rates or comparative performance of the parser.

D. Ground-truth validation protocol

The controlled scenarios ought to be defined for a production validation experiment before execution. When conducting a research on each scenario, the researchers should record their observations of the sender and receiver, the versions of the applications (if appropriate), the test action, what they expected to see happen, the process used to acquire the data, and the hashes of the archive and file, plus the mapping of the expected event and the output from the parser. Ground truth should be collected independently of the parser output, and should not be evaluated in a circular fashion. Once a production parser is in place, there are measures that can be taken such as event-level precision and recall, false positive counts, false negative counts, attachment recovery, interpretation of the timestamp, repeatability and version resilience. Only valid experiments, in which there is a known denominator and a ground truth, should be used to calculate metrics. They are not to be deduced from the number of units a student passes in a unit test.

VII. RESULTS

A. Summary of retained implementation evidence

TABLE I. SUMMARY OF RETAINED IMPLEMENTATION EVIDENCE.

Evaluation item	Retained result	Permitted interpretation
Session archive	TAR reported readable	Supports archive-level processing
File inventory	22 regular-file paths	Supports file-level inventory
Path/hash reconciliation	22 paths and SHA-256 values reported matching	Supports identity of compared files
Schema	Version 2.0.0	Supports versioned representation
Session output	22 file_inventory; 0 message_event	No message recovery established
Parser status	not_attempted	Message parsing was not performed
Regression suite	20 tests; zero reported failures/errors	Supports tested software behaviour only
Database characterization	SQLite interface did not recognize examined files	Underlying format remains undetermined

B. File-level ingestion

The Session ingestion workflow showed that the preserved archive was able to be processed into a file inventory and that the information regarding the paths and SHA-256 values was reported as matching the preserved information. This is a limited success rate, for an archived file at a limited level (file-level processing as opposed to a claim of success across application versions, devices or acquisition methods).

C. The parser state and its output in the form of schema

The output includes 22 file-inventory records and there are no message-event records. This shows that it's possible to have workflows that perform file-level observation, but do not imply that the message parsing process took place. The lack of message-event records does NOT mean that the application wasn't sending any messages; it merely indicates that these message-event records were not created by this workflow.



D. Regression results

Historical regression record 20 tests - no failure / error. The tests focus upon specific software functions such as error handling, reconciliation etc. The results provide information on the progress of implementation but are not evidence of the accuracy of signal or session content as these results are not based on independently verified expected content of production Signal or Session data.

E. Claims not established

None of the evidence retained supports:

- (i) successful recovery of production Signal messages;
- (ii) successful recovery of production Session messages;
- (iii) parser precision, recall or accuracy;
- (iv) message and attachment completeness;
- (v) deleted-message recovery performance;
- (vi) general acquisition completeness and temporal consistency; or
- (vii) false-positive rates for interpreted events; or
- (viii) superiority over commercial or open-source tools.

There has been a need for further implementation and evidence of effectiveness for each claim.

VIII. DISCUSSION***A. Recognizing the concept of inventory***

One of the main advantages of this work is the important separation of observations on the level of files from interpretations on the level of messages. The issues of whether or not the file exists, what bytes are in the file and what the content of the file means are separate evidential questions. A system can record a file and its digest without having to know what it is, and it can report that it did not attempt to parse the file, even if there is no event list found in the file. This distinction helps to conserve reporting and permits the reviews to see state visibility. It does also offer an incremental development architecture: inventory and provenance can be tested prior to the availability of a production parser.

B. Provenance as a "transversality requirement"

There may be differences between Signal and Session in internal storage and semantics. A common workflow should ensure that the identity and source for the application is maintained and not ignored as one is replaced with the other. As long as it can be traced back to the source and has been assigned meanings to its fields, a normalized record is helpful. The existing inventory implementation is a first step that is bounded towards this. For the parser contracts, explicit error state, and field mappings, message-event normalization will need to be done. The framework should not throw away application specific data if it is not possible to prove semantic equivalence.

C. Integrity - necessary but not sufficient

When comparing two evidence objects, hash verification can be used to determine whether there are any changes between them. It does not determine if acquisition was able to gather all pertinent evidence and if a live collection was internally coherent or if a parser correctly interpreted the data [5], [13] and [4], [14]. The integrity outcomes, acquisition completeness and parser validity can be reported separately, as dimensions in the schema and in the report.

D. Software verification is not equal to Forensic validation

Selected software functions were tested with evidence that they behaved as expected under the conditions that were tested, retained in the regression suite. When it comes to forensics, there is a need for independently established expected outcomes, realistic controlled inputs, defined comparison procedures and documented failures and



reproducible results[3], [4].The framework should be termed “partial software validation” until these conditions are satisfied and not validated message-recovery system.

E. Relationship to existing approaches

Current guidelines provide general guidelines for mobile evidence preservation and analysis and controlled experiments can be used to identify application-specific artifacts[2], [13].The framework supports these methods, by structuring acquisition records, inventory, integrity checks and common-schema outputs.It does not prove to be as effective as special equipment or to result in better recovery.To make a defensible comparative analysis, the evidence sets need to be the same, the versions of the tools and methods used are important, the ground truth needs to be controlled, the measures of outcomes need to be predetermined and unsupported cases need to be handled transparently.Those are not conditions that have been shown to result in those demonstrated results.

F. Threats to validity

The correctness of file-inventory and correctness of message-recovery are not the same, thus limiting the extent of construct validity.Resolution of acquisition and schema related questions are factors that limit the internal validity of controlled experiments, such as a restart comparison, which is still under review.The retained Session inventory only served as one baseline (and the other versions of the inventory, the different devices, and the different methods of acquisition not represented).The reproducibility is limited up to the time the exact release, dependencies, commands and complete logs are provided.These restrictions should be tackled with further evidence and not by using inferred metrics.Even if a production parser and independent ground-truth is available, no parser-accuracy or comparative-performance result should be reported until now.

IX. LIMITATIONS

The demonstrated Session result is only for one retained archive, and a file-level inventory, and does not constitute full acquisition.

The formats of the Signal database files and Session database files examined have yet to be determined. It's not a certainty that they will not open with a standard SQLite interface if they fail. If they fail to open with a standard SQLite interface, this is not a proof that they were encrypted.

A validated Production Signal parser and Session parser are not available. A session was not attempted to be parsed.

Defined software cases and synthetic fixtures are the only ones covered in retained regression tests, which are not designed to provide end-to-end forensic accuracy or recovery completeness.

Experiments with controlled text messages and/or the restart of applications have unresolved issues of validation. No exact message recovery is made available from the available logs and the restart comparison will need to be reviewed for possible schema and acquisition mismatches.

Records of implementation and testing should be verified and validated with the exact source code and the release artifacts prior to publishing historical implementation.

There should be no comparative superiority claim unless there are equivalent experimental data and pre-determined performance measurements.

X. FUTURE WORK

A. Production parser contracts

Explicit contracts that describe each application specific parser supported application version(s), input structures, output, provenance requirements, warnings and error states. An independent tracking system should be used to monitor parser implementation.



B. Database-format investigation

Explore the real file formats of the databases that have been looked at, with documented, non-destructive techniques. It's not possible to make a conclusion about encryption or storage architecture based on the result of an SQLite-open attempt.

C. Controlled ground truth

Designed Signal and Session experiments with independently documented sender and receiver events and versions of the application, acquisition records and file hashes. Ensure a transparent relationship between the events that are expected and any records created by the framework.

D. End-to-end validation

After production parsers, evaluate the event level precision, recall, false positive, false negative, and recovery and repeatability of the parsers compared to the ground truth. Report the resilience of the report and acquisition reliability independently of event level accuracy.

E. Reproducibility package

Display the exact source revision, license, dependency list, environment information, schema definition, test command, full output, archive and manifest identifiers, hashes, reconciliation results and representative schema records. Check package for areas that require special handling and extraneous identifiers prior to publishing.

F. Broader evaluation

Review other versions of applications and/or controlled evidence sets. When comparing to existing tools, be sure that input evidence, versions, and expected outcomes and measures are similar and documented. Isolate common business functionalities from business-specific functions.

XI. CONCLUSION

In this paper, we introduced the design and implementation of an open source framework that will help to perform cross application forensic analysis of encrypted messaging artifacts on Android. It provides an architecture to handle evidence, verify integrity, inventory files, process evidence as required by the applications, represent evidence under a common schema, normalize evidence, correlate and report evidence.

The implementation evidence retained states that there is a Session file-level ingestion workflow, a versioned schema with 22 File-inventory records and 0 Message-event records, and a historical regression suite with 20 tests with no failures or errors. Such results lend support to a limited role for these in evidence processing and software verification.

The results are not meant to prove the capability of acquiring messages, their accuracy when parsing, or the capability of the parser to prove the effectiveness of the acquisition. Formats of the databases are not yet specified and the production parsing and controlled-ground-truth validation are yet to be done. The main contribution is a foundation for implementing provenance-aware processing, which explicitly includes the various stages of processing and the processing limitations. The framework needs to be developed further and validated, independently, before it can be considered a valid system for recovering messages.

XII. ACKNOWLEDGMENT

I would like to express sincere gratitude to the Department of Forensic Science, Vivekananda Global University, Jaipur, Rajasthan, India, for providing the academic environment in which this research was undertaken. I also appreciate the guidance and encouragement received from my supervisor and colleagues during the research process, which contributed to the completion of this work.



REFERENCES

- [1] R. Ayers, S. Brothers, and W. Jansen, "Guidelines on Mobile Device Forensics," National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA, 2014. doi: 10.6028/NIST.SP.800-101r1.
- [2] J. Son, Y. W. Kim, D. B. Oh, and K. Kim, "Forensic analysis of instant messengers: Decrypt Signal, Wickr, and Threema," *Forensic Science International: Digital Investigation*, vol. 40, 2022, doi: 10.1016/j.fsidi.2022.301347.
- [3] B. Guttman, J. R. Lyle, and R. Ayers, "Ten years of computer forensic tool testing," *Digital Evidence and Electronic Signature Law Review*, vol. 8, 2011, [Online]. Available: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=909329
- [4] I. O. for S. (ISO) and I. E. C. (IEC), "Information Technology—Security Techniques—Incident Investigation Principles and Processes," ISO, Geneva, Switzerland, 2015. [Online]. Available: <https://www.iso.org/standard/44407.html>
- [5] S. W. G. on D. E. (SWGDE), "Best Practices for Digital Evidence Collection," 2025. [Online]. Available: <https://www.swgde.org/documents/published-complete-listing/18-f-002-2-0/>
- [6] I. O. for S. (ISO) and I. E. C. (IEC), "Information Technology—Security Techniques—Guidelines for Identification, Collection, Acquisition and Preservation of Digital Evidence," ISO, Geneva, Switzerland, 2012. [Online]. Available: <https://www.iso.org/standard/44381.html>
- [7] S. W. G. on D. E. (SWGDE), "Best Practices for Mobile Device Evidence Collection, Preservation, Handling, and Acquisition," 2025. [Online]. Available: <https://www.swgde.org/documents/published-complete-listing/18-f-003-2/>
- [8] I. O. for S. (ISO) and I. E. C. (IEC), "Information Technology—Security Techniques—Guidelines for the Analysis and Interpretation of Digital Evidence," ISO, Geneva, Switzerland, 2015. [Online]. Available: <https://www.iso.org/standard/44406.html>
- [9] D. Walnycky, I. Baggili, A. Marrington, J. Moore, and F. Breitingner, "Network and device forensic analysis of Android social-messaging applications," *Digit. Investig.*, vol. 14, no. suppl. 1, pp. S77–S84, 2015, doi: 10.1016/j.diin.2015.05.009.
- [10] C. Anglano, M. Canonico, and M. Guazzone, "Forensic analysis of Telegram Messenger on Android smartphones," *Digit. Investig.*, vol. 23, pp. 31–49, 2017, doi: 10.1016/j.diin.2017.09.002.
- [11] K. Kent, S. Chevalier, T. Grance, and H. Dang, "Guide to Integrating Forensic Techniques into Incident Response," National Institute of Standards and Technology, Gaithersburg, MD, USA, 2006. doi: 10.6028/NIST.SP.800-86.
- [12] C. Anglano, "Forensic analysis of WhatsApp Messenger on Android smartphones," *Digit. Investig.*, vol. 11, no. 3, pp. 201–213, 2014, doi: 10.1016/j.diin.2014.04.003.
- [13] C. Anglano, M. Canonico, and M. Guazzone, "Forensic analysis of the ChatSecure instant messaging application on Android smartphones," *Digit. Investig.*, vol. 19, pp. 44–59, 2016, doi: 10.1016/j.diin.2016.10.001.
- [14] N. I. of S. and T. (NIST), "Secure Hash Standard (SHS)," 2015. doi: 10.6028/NIST.FIPS.180-4.

