JCBI
Journal of Computing & Biomedical Informatics
ISSN: 2710 - 1606
Volume 11 Issue 02
2026

Review Article
https://doi.org/10.56979/1102/2026/1432

Revisiting Android Forensics: An MCP-Based Contextual Review of Modern Mobile Investigations

Amina Naseem[1*], Ali Sufyan[1], Hamad Moiz Chodhry[2], and Atika Athar[1]

[1]Department of Information and Communication Engineering, The Islamia University of Bahawalpur, Bahawalpur, Pakistan.
[2]Department of Computer Science, Air University, Multan, Punjab, Pakistan.
[*]Corresponding Author: Amina Naseem. Email: aminanaseem101@gmail.com

Received: May 13, 2026 **Accepted:** August 20, 2026

Abstract: The strengthening of platforms, implementation of end-to-end encryption, scalability issues, and the temporary nature of modern mobile execution environments are becoming increasingly challenging in Android forensics. Modern forensic tools are mainly artifact-oriented, depending on file-system acquisition and comprehensive application analysis. Although these methods were effective in earlier Android ecosystems, they struggle to capture execution context across operating system services, runtime environments, software-enforced security services, cloud infrastructures, and user interactions. This paper reviews Android digital forensics within the Model Context Protocol (MCP) framework. Rather than proposing a new forensic framework, it redefines MCP as a perspective for organizing heterogeneous forensic evidence across multiple system dimensions, including temporal, execution, security, interaction, and external interaction contexts. Through a literature review, the paper highlights key weaknesses in existing forensic practices, particularly their limitations in addressing runtime-based threats, discontinuous evidence, and cross-layer dependencies. The relevance of MCP is further demonstrated through contemporary Android malware examples, such as file-less malware and privacy-invasive software, including spyware and stalkerware, supported by a case scenario. Finally, the paper identifies key research challenges and proposes future directions for achieving context-aware, memory-sensitive, and privacy-conscious Android forensics.

Keywords: Android Forensics; MCP; Spyware; Stalkerware; Mobile Security; Cloud Forensics

1. Introduction

Android has evolved into a highly secure and distributed computing platform that integrates hardware-backed security features, cloud-synchronized states, application sandboxes, and operating system services [1], [2],[3]. As the most widely used mobile operating system in the world, Android devices are extensively deployed across personal, business, and governmental environments. They store highly sensitive information, manage financial transactions, and maintain continuous cloud connectivity [2], [3] Consequently, Android devices are frequently central to digital forensic investigations involving cybercrime, surveillance abuse, insider threats, and privacy violations [4], [5]. System integrity and user privacy are greatly improved by ongoing architectural improvement. But it also presents significant difficulties for forensic investigations [7]. Hardware isolation mechanisms such as the Trusted Execution Environment (TEE), file-based and full-disk encryption, restricted storage access, and enhanced application lifecycle controls are key features of recent Android versions [6], [8], [9], [110],[132]. These mechanisms significantly reduce the availability, persistence, and interpretability of traditional forensic artifacts. However, most Android forensic methodologies still rely on post-mortem examination of persistent data repositories, including databases, application file systems, configuration files, and system logs [10]. These methodologies are based on assumptions formulated in earlier iterations of Android when

file system-level access and long-lived artifacts became easier to obtain. However, modernity has increasingly challenged and undermined these assumptions. Security and privacy concerns continue to change for Android apps, and runtime attacks on mobile devices are now taking advantage of more mechanisms like cloud-based C&C, accessibility services, volatile memory, inter-process communication channels, and dynamically loaded code [2], [11], [12] .At the same time, Android's platform hardening has made forensic visibility more difficult. Evidence is increasingly fragmented across system layers and over time due to encryption, scoped storage, restricted logging, and short-lived background execution [13], [14]. Consequently, investigators may obtain isolated artifacts without the contextual information needed to reconstruct relationships between operating system events, application execution, hardware-backed security operations, cloud interactions, and user behavior. This fragmentation results in significant context loss, which has become a primary and rarely examined issue in Advanced Android forensics.

However, current Android and mobile forensic techniques offer tools for artifact analysis, event reconstruction, investigation modelling and evidence interpretation [19]; they do not, however, explicitly tie together the various aspects of the Android volatile execution state, security-state transitions, persistent artifacts, inter-component interaction, and off-device/cloud evidence in a single, re-usable, contextual taxonomy. The gap this study aims to fill is not the lack of forensic reconstruction and/or contextual analysis tools, but rather the lack of an Android-specific structure to organize and interpret these diversified evidence types from a plurality of contextual perspectives. Based on this, this study investigates MCP as an interoperability mechanism and presents a taxonomy of Android forensic analysis cases from five aspects: temporal, execution, security, interaction, external-interaction contexts.

The following objectives form the foundation of this research for addressing the limitations identified in current Android forensic methodologies:

- To examine the drawbacks of artifact-based Android system forensics models in contemporary security-sensitive settings.
- To analyze the effect of changing Android architectures (e.g., encryption, runtime execution, and cloud integration) on forensic visibility.
- To apply a context-centric view of Android forensics by reframing the MCP.
- To determine the applicability of MCP to the analysis of modern threats to Android, including fileless malware, spyware, and stalkerware.
- To define the research gaps and outline the future perspectives of the context-centric mobile forensic investigations.

Compared to earlier research, where MCP is primarily applied in AI systems or context-aware computing, this paper

redefines the concept of MCP for use in Android forensic reconstruction. MCP is seen as an interoperability mechanism for the access to the relevant forensic resources and analytical tools. This study contributes conceptually an Android-specific five-dimensional contextual taxonomy that structures heterogeneous forensic evidence from temporal, execution, security, interaction and external-interaction perspectives. Additionally, this research bridges the gap between conventional artifact- based forensic techniques and modern runtime-driven Android platforms by demonstrating that MCP can support cross-layer behavior reconstruction, particularly in emerging threats such as file-less malware, spyware, and stalkerware. This study proposes a five-dimensional taxonomy of different aspects of the forensic context of mobile devices (perspectives: temporal, execution, security, interaction, and external-interaction) to organize the various types of forensic evidence from mobile devices, and not introduces a new forensic protocol, instead MCP is proposed to be used as a mechanism for interoperability between mobile devices and relevant forensic resources and tools. The contribution focuses then on the contextualization of the organization and interpretation of evidence from the Android platform, and not on the development or implementation of MCP itself.

The remainder of this article presents the following categories. Section II describes the state of Android forensics and explains important drawbacks of the traditional forensic methods. Section III proposes the MCP and its Android forensic analysis conceptualization. Section IV maps Android forensic artifacts to MCP contextual dimensions and explains cross-layer evidence correlation. Section V, however, covers the current classes of Android threats that go deeper than conventional file-based malware and contains a case

study of stalkerware research. Section VI presents several different research issues and describes potential future approaches, while Section VII provides the conclusion of this paper.

2. Review Methodology and Scope

2.1. Literature Sources and Search Scope

In this paper, a narrative review and conceptual approach are used to review the current practices in Android digital forensics and the potential of the Model Context Protocol (MCP) as a context oriented view for the forensic analysis process. The literature coverage was mainly obtained from the Google Scholar and IEEE Xplore search engines, and the main coverage period of the literature was 2020-2026. Previous publications and books were also used if they offered basic concepts or a method for forensic analysis that was relevant to the analysis. The literature search centered on the following topics related to mobile-cloud forensics: Android digital forensics, mobile-device forensics, Android forensic artifacts, Android memory forensics, mobile forensic investigation models, forensic event reconstruction, mobile/cloud forensics, and contemporary Android security challenges. The literature pertaining to the MCP was analysed separately, as this is a relatively recent protocol. Besides academic publications, the original MCP documentation/specification was analyzed along with relevant MCP implementations and projects to gain insight into the protocol architecture, context exchange, tool integration and applicability to forensic workflows. The sources were deemed relevant when discussing Android/mobile forensic investigation, forensic evidence and artifact analysis, runtime or memory evidence, event reconstruction, mobile/cloud evidence, Android security mechanisms, or MCP architecture and security. Older sources were included when they were providing basic concepts that were still pertinent to the discussion. The literature selected was then synthesized narratively in order to find common limitations in artifact-centric Android forensics and to inspire the five-dimensional contextual taxonomy that is proposed in this work. The comparative tables were based on the selection of the sources and the extraction of the forensic scope, the type of evidence to be found, the contextual abilities, and the analytical characteristics of evidential items reported or described. This review is thus not meant to be systematic or a meta-analysis, but rather a narrative and conceptual synthesis.

2.2. Source Selection Criteria

The literature that was found and identified was chosen with a purpose as it was relevant to the aims and scope of this study. Publications focusing on Android and mobile digital forensics, acquiring and analysing forensic artifacts from Android and mobile devices, volatile and runtime evidence, the Android security mechanisms, mobile and cloud digital forensics, forensic event reconstruction, current Android threats and context-aware or AI-driven digital forensic analysis were considered a priority [15]. To contribute to the discussion specific to MCP, academic research related to the MCP was complemented with the original MCP documentation and technical projects that have implemented or tested MCP, as the protocol has recently emerged and there is relatively little academic literature specifically focusing on MCP in digital forensics.

2.3. Comparative Analysis and Table Construction

The comparative tables were created by qualitative synthesis of representative studies that were selected for their relevance to the research of Android and mobile digital-forensics. The selected works cover various time periods and forensic methods such as automated application analysis, forensic surveys, automated data collection, device-level acquisition, static artifact analysis, artifact-centric analysis, tool comparison, and integrated forensic frameworks. The studies were analysed on the basis of their main forensic approach, focus area, strengths reported or described and weaknesses relevant to the aims of this review. The comparison was used to highlight common shortcomings in current methods, including the lack of forensic evidence integration, lack of runtime visibility, lack of cross-layer/contextual analysis, and reliance on manual correlation.

3. Android Forensics Landscape: Current State and Limitations

Early Android forensic investigations were carried out in a setting that was comparatively less sophisticated, with less developed encryption keys, a less complicated device storage model, and easier access to data stored in applications because Android security features were initially minimal and data was more easily accessible. Refer to survey literature in subsequent audits as well. In such a situation, the

acquisition and analysis tools based on artifacts proved to be effective and adequate for a broad study [17]. As the Android platform developed to focus on security and privacy and cloud integration, these early methods underlying presumptions gradually started to become out of pace with the modern execution environment. [18].

3.1. Artifact-Centric Forensic Practices

The investigations of Android forensics have historically been built on an artifact-focused framework, with the primary goal being the forensic analysis and retrieval of resistant computer data [17], [20]. The most common forensic procedures include file system extraction [21], recovery of application- specific data, analysis of system and application logs, and analysis of configuration files [22],[125]. These methods are based on earlier digital forensics concepts that were created for desktop and early mobile platforms, where the primary focus of evidence storage was persistent storage [16]. Artifact- based approaches offer several benefits, particularly from both academic and practical perspectives. Artifact-based approaches support the forensic acquisition and analysis of persistent application artifacts [23] .As investigative hypotheses evolve, forensic methods that were built using solid evidence through hashing are reassessed. This stability supports reliability, juridical admissibility, and consistency with traditional forensic procedures. Therefore, a significant portion of the current knowledge is in place. The Android forensic literature is concerned with methods of extracting, decoding, and reading device-stored static artifacts and application data, including backups and social media or messaging application artifacts [19], [24], [96],[111]. This assumption is becoming less trustworthy as Android systems start paying more attention to privacy, security, and the use of fewer resources [25]. Various modern forensically significant behaviors are programmed to be as lightweight as possible on persistent storage, making the analysis of static artifacts alone less effective.

These limitations result in a major challenge for existing Android forensic techniques, namely the difficulty of consistently assessing connections between dispersed and transient sources of evidence across system layers[124]. Investigators are no longer limited to using permanent evidence as the exclusive method of forensic analysis since they increasingly rely on a runtime execution environment, inter-process communication, and cloud-integrated services. Instead, there is a growing need for analysis techniques that emphasize contextual connection between system levels over time. Context-oriented techniques, which view evidence as a component of a broader system of behavior rather than as a collection of discrete data points, are the outcome of this shift. This perspective highlights the complexity of contemporary Android forensic investigations and motivates the examination of interoperability frameworks such as MCP [26].

3.2. Layered and Tool-Centric Analysis

The current Android forensic approach also includes a layered approach and a tool-centered approach to analysis. Typically, the system is broken down into several layers, such as the operating system components, network, application data, and cloud services. services, one of which is investigated using specific tools or techniques. This can be seen in Android architecture and the creation of forensic tools tailored to certain data sources [27] A study can be limited in scope by using the layered analysis approach, which allows the investigator to concentrate on a particular level or data source. This could be countered by a drawback, though, as it can be challenging to integrate data from many model layers into a logical analytical model that shows how application activity, system events, and user actions are related. As a result, researchers and current research typically manually connect data from many sources, relying primarily on their experience and intuition rather than employing systematic, rigorous models.

Academic research also shows that division. Division is also exhibited in academic research. Cross-layer synthesis is often paid attention to by research, which often looks at stand-alone topics such as limited application behavior analysis, cloud artifact recovery, and virus detection. As shown in Fig.1, Android forensic analysis is typically conducted in isolated layers. Although not all of them, the contributions to the causes are a feature of a particular subfield, which does not have structured procedures. The comparison of the outcomes on the layers limits the reconstruction of the whole Android ecosystem implementation story.

Although many Android forensic strategies have been suggested in recent years, they concentrate on various angles of evidence gathering, analysis, and interpretation[114]. The initial studies were mainly focused on traditional forensic methods and operated on fixed data containers, i.e., file systems, application

databases, and system logs. Although these techniques have been successful in conventional forensic contexts, they frequently do not have the capability of tracing dynamic runtime behavior and cross-layer interactions in the Advanced Android context. Recent works have undertaken automated analysis techniques, memory forensics, and cloud- integrated investigations, in the hope that mobile ecosystems are increasing in complexity.

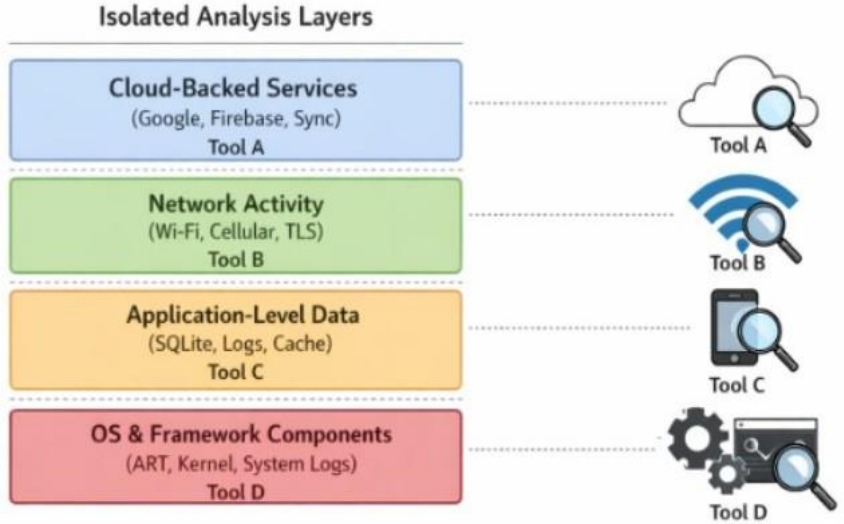


Figure 1. Layered, tool-centric Android forensic analysis.

However, these approaches lack a cohesive analytical framework and are fragmented, primarily concentrating on specific layers or categories of evidence. As Android systems continue to advance with improved security features, widespread encryption, and distributed system design, the flaws in the existing forensic procedures become increasingly apparent. Forensic event reconstruction has also been pushed further than just the examination of artifacts and a timeline in recent research. In the structured framework of Breitinger, Studiawan and Hargreaves, [28] the terminology and the process of event reconstruction and the challenges involved have been harmonised. Recently, Adila, Studiawan, and Breitinger [29] study event data processing for forensic event reconstruction, extracting sequences, process models, and investigative patterns from event data. The following studies show that temporal reconstruction and behavioral interpretation are well developed fields of digital forensic research. They are mainly interested in reconstructing and analyzing event sequences, but not in developing an Android-centric contextual taxonomy to organize the heterogeneous evidence during runtime execution as well as security state, inter-component interaction, and interactions outside the device boundary. Thus, the gap that this study proposes to fill is not the lack of forensic event-reconstruction methods but the lack of a unified structure for the Android context that can relate the different perspectives of the forensic evidence in a common analysis model.

To further understand the limitations and identify research gaps, Table I offers a comparative study of representative studies.

Table 1. Comparative Analysis of Representative Android Forensic Approaches

Reference	Approach	Focus Area	Strengths	Limitations
[30]	Automated application analysis	App-level forensic investigation	Scalable and systematic analysis	Focus restricted to application layer only
[31]	Survey of forensic techniques	Trends and challenges in mobile forensics	Comprehensive overview of existing methods	Highlights lack of unified analytical models
[32]	Automated forensic tools	Data extraction and reporting	Improve efficiency and automation	Limited contextual and cross-layer analysis

[33]	MCP landscape and security analysis	MCP architecture, security threats, and research directions	Provides a broad analysis of MCP capabilities, security considerations, and research challenges	Not specifically designed for Android forensic evidence correlation
[16]	Data acquisition methodology	Device-level evidence collection	Early standardized acquisition techniques	Limited support for modern encryption and runtime artifacts
[17]	Static artifact analysis	File system and app data extraction	Automated identification of stored evidence	Cannot capture runtime or volatile behaviors
[51]	Systematic literature review	Tools, processes, and methodologies	Broad coverage of forensic approaches	Identifies fragmentation in forensic workflows
[34]	Comparative study of tools	Tool effectiveness and methodology comparison	Evaluation of forensic tools	Limited discussion on contextual analysis

3.3. Android Platform Security Evolution

As the Android system has developed, safety concerns continued to exist, and every major version includes features meant to improve user privacy and system resilience [35] ,[36], [43].

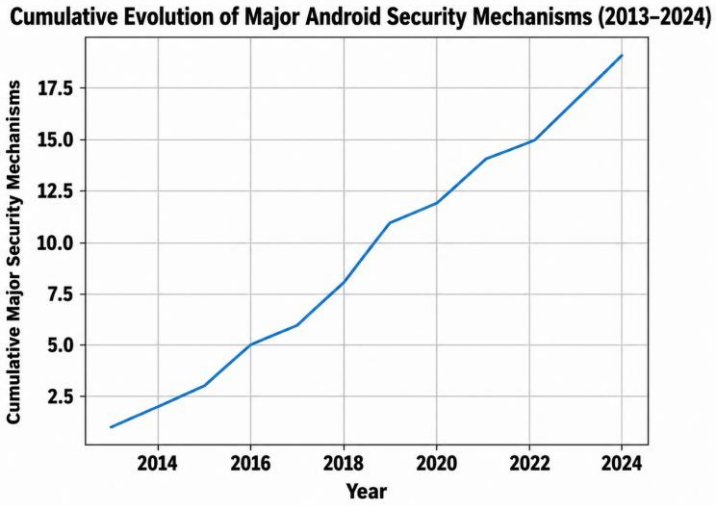


Figure 2. Major Android platform security mechanisms across releases.

Figure 2 shows how significant security features for the Android platform have been added one after the other over the course of multiple releases. The information comes from official Android release notes and security white papers that detail documented architecture improvements [1] , [37] , [38], [39]. The cumulative values are not an exhaustive record of all incremental updates or patches; rather, they are an aggregated conceptual count of important security controls.

The gradual implementation of features like runtime permissions [40] hardware-backed isolation, application sand-boxing, and required access controls is explained in the official Android security guide [1]. The development of the Android permission model and core platform security architecture is further examined by complementary research investigations [37], [88]. File-based encryption [41], scoped storage defenses per-user and per-application key isolation, required access controls, and hardware-backed security services [44] offered through trusted execution environments [45] are significant progress[116]. File-wiping applications can remove or alter forensic artifacts, creating additional challenges for the

recovery and interpretation of deleted evidence on Android devices [42] All these mechanisms will decrease the attack surface and reduce unauthorized access to sensitive data [46]. Furthermore, the level and complexity of mobile threats continue to grow, according to recent industry assessments. According to Kaspersky's 2024 mobile threat report, 33.3 million attacks involving malware, adware, or unwanted mobile software were prevented in 2024, while adware accounted for 35% of mobile threat detections [47]. Despite enhanced security measures, investigative visibility has been increasingly restricted: hardware-based isolation reduces transparency to the context of protected execution, scoped storage limits cross-application visibility, and encryption prevents access to the remaining data [48]. Also, gradual denial of system logging and debugging. interfaces and background execution decrease the availability of system-wide monitoring formerly available to investigators [49]. Even though these measures are necessary in protecting end users, they remake the forensic scene through the reduction of observable surfaces and minimizing artifact persistence [50] .Investigators are becoming more in demand Instead, it means to manipulate partial visibility, indirect indicators, or metadata. than extensive datasets. This change poses a threat to conventionality. Assumptions concerning evidence availability must be reconsidered, as well as the methods through which forensic insight can be derived in high-security environments.

3.4. Runtime and Volatile-State Constraints

A major weakness of current Android-related forensic techniques is that these are limited in their ability to see inside the running program and volatile system states [15], [31],[119]. Most investigative methods focus on the static analysis of stored data, whereas execution-time activity, artifact that resides in memory, and transient inter-process interaction are given relatively little emphasis. The further development and support of volatile memory acquisition and analysis on Android is limited by the architectures of these devices, restrictive permission models, and security mechanisms provided by the platform [51], [52], [53], [54],[130].

The dynamically loaded code, in-memory decrypted content, Binder-based inter-process communication, and interactions with short-lived services become more central in Advanced Android behavior [87]. Several of these components have short- lived existence during execution and might never be written to permanent storage. Moreover, Android enforces strong process life cycle management, such as background execution constraints and memory reclamation, which can quickly kill processes and garbage volatile state. Associated evidence can be lost permanently once a process is terminated or once the state of a device is changed. Such a limited observability of a volatile execution environment greatly restricts forensic understanding of transient or file-less actions. As a result, the investigations might not be able to preserve important behavioral evidence, especially for those cases that involve memory-resident malware, runtime permission abuses, or techniques of ephemeral exploitation that were strategically meant to leave no record of the attack [54], [55].

3.5. Implications for Modern Android Threats

When you combine artifact-driven techniques, multi-layered analysis, the changing landscape of platform security and the lack of runtime awareness, this has major impacts on the investigation of current Android threats [56] . New threat categories such as spyware [56], stalkerware, privilege escalation and accessibility services abuse often [50] operate within the bounds of system restrictions, while actively decreasing persistent indicators of compromise [57], [58]. In some circumstances, individual artifacts can be seen as benign or uncertain. Understanding how permissions were utilized, services accessed, or local execution connected to remote cloud activities might be challenging without appropriate contextual correlation [18], [81]. This is particularly visible in circumstances of low-noise data extraction, hidden surveillance or misuse of trusted platform attributes [59] [20], [60].These scenarios introduce the need for investigative approaches that go beyond in-depth data extraction. But more on the understanding of the behavior, the connection between layers and maintenance Contextual connections are crucial attributes of Android threats to cope with the variations. These consequences are familiar and can provide the basis for accessing the Android forensic evidence that are designed and investigated in the context of defense and security.

Figure 3 shows the evolution of Android forensic procedures from conventional, everyday artifact-based methods to more complex context-based methods with the increasing sophistication of Android environments.

These restrictions highlights a significant weakness in current Android forensic methods: while the analysis of artifacts and layers can be useful, evidence from distributed and ephemeral sources is analysed and not necessarily associated with a common context for relating observations across the layers and across time. Such techniques for forensic event reconstruction, timeline analysis and process-mining based behavioral interpretation have been previously established in existing research [28] and [29]. The other challenge in this study is not that there are no reconstruction techniques, but that there are multiple contextual aspects of the evidence found on Android devices and it is distributed heterogeneously across them. This encourages a contextual approach, which can relate temporal, execution, security, interaction and external-interaction evidence in a shared analytical framework, and MCP is viewed as an interoperability framework for accessing relevant forensic sources and analytical tools, relevant to the Android-specific context.

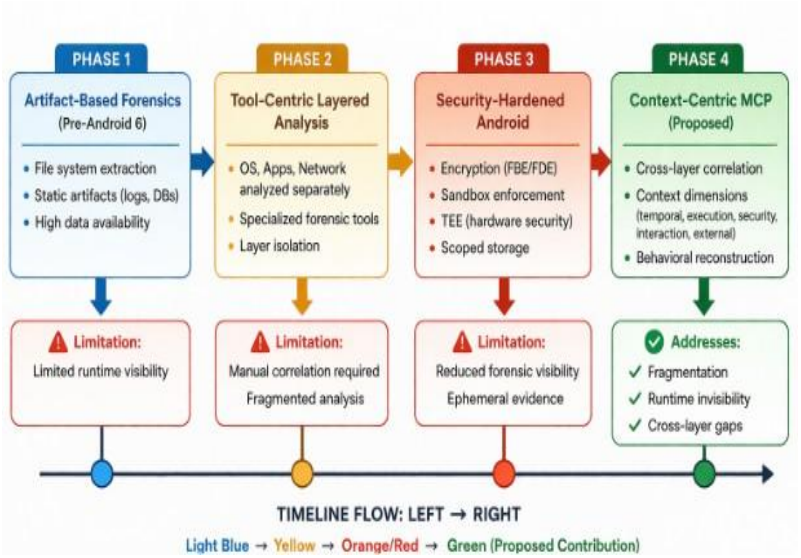


Figure 3. Evolution of Android forensics from artifact-based methods to context- centric MCP analysis.

4. Model Context Protocol: Conceptual Overview

In this section, we discuss the possibility of using the Model Context Protocol (MCP) as an interoperability tool to facilitate context-oriented Android forensic analysis [61],[113]. The proposed approach utilizes MCP to allow access to heterogeneous forensic resources and analytical tools, and the five contextual dimensions provide an Android-specific framework to organize and interpret forensic evidence.

4.1. Inspiration and Conceptual Justification

The structural limitations of the current Android forensic techniques affect how evidence is interpreted in modern situations, as was covered in Section II. The MCP [61] [84] is presented in this part as a context-oriented analytical methodology based on these observations.

Conventional forensic methods are mostly concerned with obtaining and examining permanent artifacts like files, databases, and system logs . However, much of modern Android activity occurs during execution and involves dynamic interactions among operating system services, application components, hardware-backed security mechanisms, and cloud-connected infrastructures [85]. It was originally introduced as a context-aware interaction framework for AI systems [62].. MCP puts more emphasis on keeping contextual information safe and aligning it than on handling separate data pieces. This context-dependent paradigm proposes that the execution environments are to be considered as coherent systems where system state, execution behavior, security conditions, and external interactions are correlated with time.

From this point of view the conceptual framework for exploring the heterogeneous forensic observations in the same analytical frame of work. This method can also be justified based on recent studies in memory forensics and execution-state analysis, where the in-memory behavior and runtime state are shown to be of evidentiary significance in addition to artifacts of persistent storage. In the present study, MCP is considered as an interoperability mechanism for connecting AI applications with external forensic

data, resources and analytical tools. The proposed contextual model provides the forensic organization and correlation of Android evidence and not the MCP specification itself.

4.2. Forensic Adaptation of MCP

In the context of the above-mentioned context-driven, this review examines the applicability of MCP to the field of Android digital forensics. Even though MCP was initially designed to maintain contextual integrity between distributed AI systems, the principle of structured contextual alignment in its design provides conceptual applicability to Android forensic reconstruction, where cross-system evidence correlation is required [63]. In Android systems, there are no evidentiary artifacts. The generated files, logs, database entries, and memory objects are created in certain states of execution and regulated by permission models, security mechanisms operated by hardware, and inter-process interactions. The conventional forensic approaches often interrogate these artifacts separately, with no systematic attempt to preserve the contextual circumstances in which they were produced or altered. This is a limitation of the possibility of reconstructing coherent behavioral sequences based on only isolated forensic observations [64].

In forensic terms, MCP may be reformulated as an analytical paradigm, in which the primary focus is an orderly correlation of various dimensions of the system, such as the behavior of execution, transitions of a security state, sequence in time, and interaction with the external environment. In this conceptual adaptation, context is a multidimensional description of the system's status at a specific moment in time rather than just a straightforward metadata. [18].

Instead of suggesting a technical redefinition of the original MCP specification, the present paper frames MCP as an analytical prism in the context of which the Android forensic evidence can be interpreted. This approach allows the analysis to transcend the study of single artefacts and to structured behavioral reconstruction, responding to the runtime volatility and cross-layer fragmentation challenges presented in Section III

4.3. Relationship to Existing MCP-Based Digital Forensic Research

A recent paper by Hilgert et al. [113] discusses the use of Model Context Protocol (MCP) for Digital Forensics and Incident Response. Their study theoretically analyzes MCP-based forensic workflows, such as artifact analysis, dealing with forensic data sources and tools, creation of an interpretable report, and the deployment of MCP servers in forensic environments. One of their key contributions is the notion of an inference constraint level for characterizing how the design choice of an MCP can limit the behavior of an LLM to make it easier to audit and trace the LLM in forensic workflows[113]. This is a related, but different problem that is addressed in the present study. While the focus of Hilgert et al. is on the interoperability and workflow aspect of MCP in LLM assisted digital forensics and incident response, this work addresses the organization and interpretation of heterogeneous Android forensic evidence. The suggested contribution is an Android specific contextual taxonomy consisting of temporal, execution, security, interaction, and external-interaction dimension. These dimensions are not proposed as additions to or parts of the MCP specification. Rather, MCP is viewed as a means of interoperability to make the forensic data, resources and tools available, and the proposed contextual dimensions as an analytical framework in which to structure and correlate the Android forensic observations. This difference is significant because this study is not about introducing MCP into digital forensics, or even creating a new MCP implementation. Instead, it discusses how it can be conceptually integrated into Android forensic analysis and suggests a set of structured perspectives to consider when associating persistent artifacts with an Android application's run-time state, security, inter-component dynamics, and interactions across the device boundary.

4.4. Context Dimensions in Android Forensics

4.4.1. Conceptual distinction between MCP and the proposed forensic model.

In this study, the Model Context Protocol (MCP) is employed as an interoperability mechanism that enables AI applications to interact with external data, resources, and analytical tools [112]. The five contextual dimensions *Temporal, Execution, Security, Interaction, and External Interaction* are not specified in the MCP specification but are used here to categorize the heterogeneous Android forensic evidence based on the characteristics of the modern Android systems. Therefore, the interoperability foundation is provided by MCP, and the proposed Android forensic contextual model is used to organize and interpret

evidence in these five dimensions. The five dimensions are chosen to ensure complementary analytical coverage of the main contextual features of interest for modern Android forensic evidence. Temporal Context is about when an observation is made and how it relates to other events; Execution Context is runtime state and application/system behaviour; The Security Context contains permissions, access-control decisions, and security state changes: Interaction Context describes relationships between applications, services, operating-system components and other entities within the device ecosystem. External Interaction Context describes interactions that cross the device boundary, e.g. with cloud services, remote servers, APIs and network infrastructure. Such dimensions are meant to be analytical lenses and not mutually exclusive categories. Thus, one piece of evidence can be part of several dimensions at the same time, where the interpretation depends on more than one contextual characteristic.

Interaction Context describes the relationships between applications, services, and operating-system components in the Android environment. External Interaction Context describes interactions that cross the device boundary, such as cloud services, remote servers, APIs, and network infrastructure. This separation is maintained as external interactions introduce distributed evidence, different provenance and acquisition requirements, and network or jurisdictional considerations.

4.4.2. Evidence Entities and Events

In the context of this conceptual model, a forensic artifact is an original evidence source, which in this instance can be a system log, an application database, a memory image, network record, or another forensic data source. An evidence entity is an observation that is relevant and extracted from or identified from one of these sources; an event is an occurrence of a system and is represented by one or more evidence entities, including service interaction, network communication, application execution, permission change, and more. These definitions, which are conceptual, are used to provide a common ground for the organization of the heterogeneous Android forensic evidence, without introducing a new engine for the extraction and processing of evidence and events.

4.4.3. Conceptual Representation of Forensic Context

Conceptually, an evidence entity can be represented as a structured data record with attributes of evidence identifier, artifact or source, timestamp, event or observation, source component, associated contextual dimension(s), related entity or entities, provenance information, and uncertainty or confidence indication. An observation, for instance, like an access to location service, could be described along with the originating Android system log, the time of the access, the application or component that was used, the security dimensions that applied to the access and the interaction dimensions that were applicable, etc., and also with the source provenance. Proposed as a conceptual model for the organization of forensic context, it is not a database schema or a software data structure that is in use in the current study.

Table 2. Conceptual Representation of an Evidence Entity

Field	Example
Evidence ID	E001
Artifact/Source	Android system log
Timestamp	10:31:04
Event/Observation	Location service accessed
Source Component	Chrome
Context Dimension(s)	Security, Interaction
Related Entity	Location service
Provenance	Source log / acquisition information
Confidence	High / Medium / Low

4.4.4. Proposed Contextual Dimensions

Temporal Dimension: The temporal dimension records chronological links between system events, as well as process life cycle, permission calls, service bindings, and external interactions. It is necessary to have the correct sequence of events to determine causality. In MCP, temporal correspondence is the foundation of the heterogeneous evidence correspondence between runtime and persistent layers.

Execution Dimension: When the system is operational, the behaviors of the systems and applications are indicated by the execution dimension. These include dynamically loaded code, in-memory objects,

thread activity, process state changes, and inter-process communication based on binders [68], [75]. A realistic cognitive reconstruction requires the runtime state to be included in the behavioral analysis because advanced Android threats frequently operate in unstable execution environments [65], [66].

Security Dimension: The security dimension includes key store operations, encryption states, UID/GID assignments, SE-Linux enforcement domains, permission models, and inter- actions with trusted execution environments (TEEs) [67], [68],[131]. These mechanisms influence the creation, accessibility, and security of artifacts. Instead of just being viewed solely as obstacles to data acquisition, security-state transitions can be in MCP as contextual signals that guide forensic interpretation.

Interaction Dimension: The interaction dimension defines the relationship between components of a systems, such as system service calls, content provider access, broadcast receivers, accessibility services, and application-to-application communication. Such interaction routes can tend to expose the behavioral purpose which cannot be directly observed by simply looking at the artifacts in the form of a static object [69] [70].

External Interaction Dimension: A significant amount of the functionality of advanced Android smartphones is dependent on network infrastructures and cloud services [71], [72],[133]. Also, the variety of data sources from applications, system logs, network and cloud data sources result in data integration issues [107], [121]. As Android apps often communicate with remote servers via network requests and APIs, which could leak sensitive data flows and behavioral patterns to servers as part of the security analysis [73]. TLS sessions, sync events, remote command-and-control communications, and account- linked activity are all included in the external dimension. Reassembling distributed behaviors that were performed outside of the device itself becomes possible when local device state can relate to external interactions.

To review Android forensic evidence, all of these dimensions are arranged in a systematic environment. These dimensions are used together to conceptually structure and organize the heterogeneous forensic observations of Android.

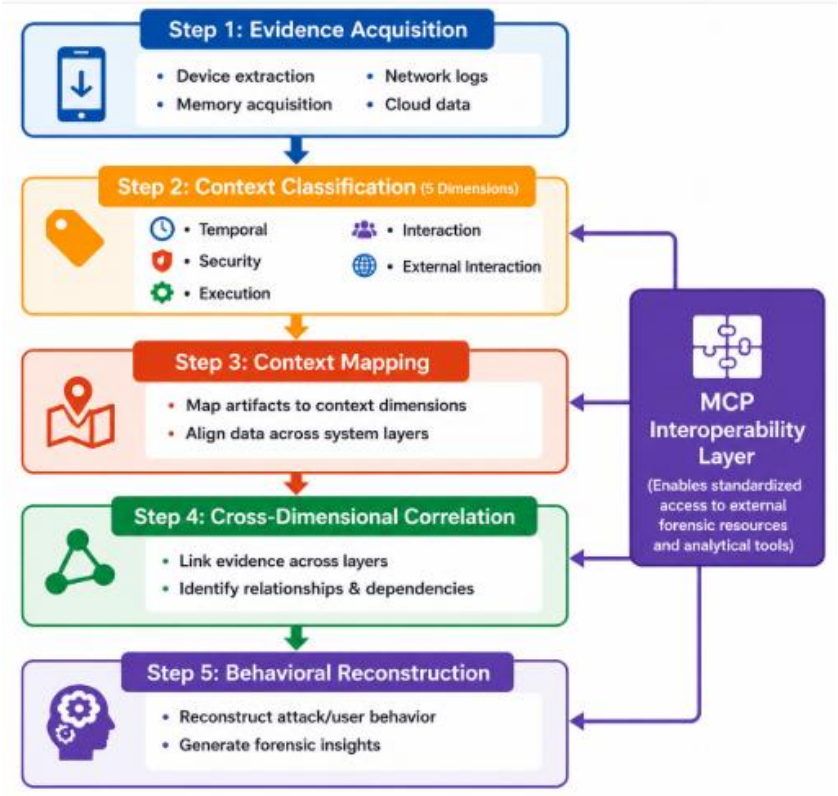


Figure 4. MCP-based investigation workflow for Android forensics.

Figure 4 provides the MCP-based investigation process of Android forensics. It starts with evidence sourcing via multi- faceted sources, then the artifacts are classified into different contextual dimensions. The MCP context layer supports balanced mapping and cross-dimensional correlation of evidence, which

allows one to reconstruct behavior correctly and increase the effectiveness of the entire process of forensic analysis.

4.4.5. Mapping Evidence Entities to Contextual Dimensions

One or more contextual dimensions can be linked to an evidence entity, depending on the nature of the observation the evidence entity represents. The mapping depends on the forensic significance and the attributes of the evidence and not just the type of artifact alone. When the evidence available provides direct or inferable information related to a dimension, then the entity is associated with the dimension. As a result, evidence entities can take part in multiple context types, such as temporal, execution, security, interaction, and external-interaction context. This multi-dimensional mapping enables evidence from a single artifact to also support multiple parts of the forensic interpretation without having to be categorized and assigned to one type.

Table 3. Conceptual Mapping of Evidence Entities to Contextual Dimensions

Evidence Observation	Temporal	Execution	Security	Interaction	External Interaction
Application started at 10:30	✓	✓	—	—	—
Permission granted at 10:31	✓	—	✓	—	—
Application accessed a location service	✓	✓	✓	✓	—
Application contacted a cloud API	✓	✓	—	✓	✓

The following examples show that contextual assignment does not have to be exclusive. For example, an application-to-cloud communication may contain temporal information, show the activity when the application is running, express an interaction with other service, or be an external interaction. It outlines the proposed conceptual mapping logic; the table is not the result of an automated, implemented classifier.

The available evidence entities are conceptually aligned on a common investigative timeline, based on their temporal metadata (and existing timestamps). If sources provide different time references, normalization of the time reference should be undertaken to interpret temporal relationships. If timestamps are not available, they should be assumed to be approximate or possibly inaccurate; they should not be assigned an exact time [126]. Thus, the temporal relationships can be interpreted as a precise time, as sequential events or as a time interval, depending on the level of reliability and precision of the available evidence.

4.4.6. Conceptual Evidence Correlation and Interpretation

The proposed model, at a conceptual level, considers the correlation of evidence not as a resolution mechanism, but a process that preserves evidence. If multiple observations from different sources contradict each other, the competing evidence entities are not silently dropped or resolved but are retained and taken into account based on their source, time, acquisition conditions, and confidence. Evidential provenance and chain of evidence information associated with each evidence entity allowing for the ability to trace back to its original sources. Confidence and uncertainty can be shown with an observation, to indicate that they are well supported or incomplete, approximate, or possibly not reliable. A cross dimensional relationship has two or more evidence entities that have a defensible temporal, causal, functional or contextual relationship that can be supported by the attributes of the evidence entities and the evidence's provenance. The various observations across the five dimensions are then collapsed conceptually into a behavioral reconstruction, which must include unresolved contradictions and uncertainty, but also must be a coherent sequence or set of possible explanations. They are considered as conceptual mechanisms for context-related forensic interpretation and not applied or tested in the present review. These mechanisms are offered as conceptual concepts for contextual forensic interpretation; their implementation and experimental evaluation are beyond the scope of the present review.

4.5. Core Principles of Forensic MCP

To enhance the comprehension of contextual information in Android forensic investigations, several conceptual principles direct the forensic reinterpretation of the MCP [74]. These principles transform the

context-driven philosophy of MCP into analytical policies applicable in complicated and security-enhanced mobile contexts.

Principle 1: Context as the Primary Analytical Unit. Traditional Android forensic techniques often concentrate on the extraction of individual artifacts such as files, logs, and databases. .In the forensic MCP view, the artifacts are perceived as something encompassed in a broader context. Evidence gains significance by its relationship to system conditions, execution behavior, and surrounding events. Investigators can create behavioral accounts using context analysis as the unit of analysis, but not a set of separate data points.

Principle 2: Cross-Layer Evidence Correlation. Android has several connected layers of operation, such as operating system functions, application frameworks, runtime environments,

security elements that are supported by hardware, and cloud- connected frameworks. MCP promotes organized correlation between these layers to reduce the artifact fragmentation. It is important to evaluate the evidence gathered from one subsystem in relation to the actions occurring at other layers.

Principle 3: Temporal Continuity of Events. Maintaining the chronological relationships between the system events is essential for accurate forensic reconstruction. The temporal synchronization of the artifacts and runtime states on a uniform investigation timeline is the main emphasis of MCP. The investigators can ascertain how the system operates, user behaviors, and interactions with the external environment change over time according to this concept, which supports causal reasoning.

Principle 4: Integration of Volatile and Persistent Evidence. Modern Android behavior is frequently observed in volatile execution contexts, including memory processes, dynamically loaded code, and transient inter-process interactions. Both volatile runtime state and persistent artifact can be detected by MCP as complementary sources of forensic evidence. Reconstructing complex tasks that are not observable with static artifacts is improved by incorporating the sources into the work.

Principle 5: Security-State Awareness. The security mechanisms of Android (enforcement via encryption, limited storage, permission control, and secure execution) affect the generation of artifacts and access to them. In the MCP, the framework, these mechanisms do not restrict evidence collection but provide contextual cues that govern system behavior.

Principle 6: Multidimensional Evidence Interpretation.

The proposed model views multidimensional evidence interpretation as the integration of observations over the contextual dimensions defined in Section 4.3.4. These principal stresses that forensic relevance can arise from the relationships among observations, not from an isolated artifact. These concepts offer a basis for the possible use of the MCP in Android forensics. This model facilitates a more comprehensive analysis of forensic evidence inside the mobile ecosystem by highlighting contextual continuity and systematic connection.

4.6. Conceptual Architecture of Forensic MCP

The forensic MCP will be structured within a model framework to systematically organize Android forensic evidence. MCP focuses on **forensic** reconstruction through contextual composition instead of as objects that are data sources, [75]The protocol integrates several aspects of system activity to preserve the relationships between events, system state and execution. This is based on a major challenge in traditional artifact-based forensics, which is that evidence from different subsystems is often separated and examined independently from the broader operational context of the data.

The major context dimensions that were outlined earlier in this section surround the MCP Context Model. The time dimension provides a chronological sequence of system activity, which permits the investigator to build proper timelines of application **activity**, system transitions, and system communications with networks. The execution dimension captures runtime process activity, memory conditions, dynamically loaded code [53], and inter-process communication patterns, all of which are often characteristics of contemporary Android applications [60], [76], [83]. Permission states [59], enforcing encryption, and hardware- supported security are examples of security dimensions that impact a system's behavior and access to artifacts [77] . In real terms, these contextual factors are complementary analytical perspectives rather than sources of proof. The observations in one dimension can be explained by the data collected in another. The proposed Android forensic contextual model organizes the evidence into the five dimensions of context, while also connecting observations from various parts of the system and from the

outside environment. MCP can help bring these observations to life and provide the forensic resources and tools needed to obtain and analyze such observations. Furthermore, the model highlights how runtime activities, security limitations, and external interactions all contribute to forensic visibility of Android systems and direct investigators toward more context-sensitive analytical techniques.



Figure 5. The forensic MCP framework's conceptual design highlights the contextual aspects of the MCP context model.

The proposed Android forensic contextual model classifies heterogeneous evidence in five complementary dimensions: temporal, execution, **security**, interaction, and external interaction. The model is intended to represent relationships between evidences from different layers of Android system and external sources. MCP is the interoperability mechanism that allows an AI-assisted forensic environment to access the relevant forensic resources and analytical tools. The proposed contextual model provides the structure for organizing and relating the resulting evidence.

5. Mapping Android Forensic Artifacts to the Proposed Contextual Dimensions

The section provides a crosswalk between the most common Android forensic artifacts and this work's five contextual dimensions. The mapping shows the different ways evidence can be heterogeneous and organized in a way that allows for temporal, execution, security, interaction, and external-interaction characteristics; MCP offers an interoperability mechanism for accessing the evidence and analysis tools from which such evidence may be derived.

5.1. Android Forensic Evidence Sources

The MCP is presented as a context-driven method [29], [61] for analyzing Android forensic data in the conceptual framework presented in Section 3. An analyst must examine how the Android forensic artifacts align with the contextual characteristics outlined in MCP to demonstrate the analytical utility of this framework. Today's Android smartphones generate a wide range of evidence due to network communication, runtime states, operating system activity, and application execution. All these components are useful in forensic investigations since they represent the device's operational activity [78], [79].

Determining persistence artifacts in the file system, ap- plication storage, and system logs have long been a part of Android forensics. Application databases, configuration files, user-generated content, and system event logs are examples of such artifacts [115],[117],[127]. In addition to these long-term sources, Android systems generate temporary artifacts in real-time due to factors like memory allocations, dynamically loaded binary code, and process states. When displaying an application's behavior that cannot be recorded in long-term storage, such runtime artifacts might be extremely important [80]. Moreover, Android devices are also closely connected with the external infrastructures like cloud services, remote servers and application programming interfaces. This leads to numerous user actions and application activities that leave evidence that extends outside the immediate environment of the local device. Network

logs, such as network traffic, synchronization, and remote service access, ess can be of significant interest to forensic investigations. Location-based Android applications can also generate application-level evidence through interactions with mapping and location services [5].

5.2. Mapping Artifacts to MCP Context Dimensions

Artifacts are produced by execution environments, which are influenced by system rights, runtime conditions, and interactions among programs and services.

[81], [82]. In turn, successful forensic reconstruction involves the procedure of matching the artifacts through different contextual lenses [83] .In a modern Android system, application behavior frequently spans several levels of the operating environment, including the operating system framework, runtime execution environment, and external communication channels [84] . Consequently, an individual artifact can be a component of a larger behavioral sequence, and a single analysis is therefore inadequate in comprehending complicated device behavior [85], [86].

By categorizing forensic artifacts according to the contextual factors mentioned above in this study, the MCP framework resolves this challenge. Investigators can connect artifacts to the system conditions and execution scenarios in which they were created through this systematic mapping approach. By connecting artifacts with temporal, execution, security, interaction, and external interaction settings, MCP helps investigators find connections between occurrences that might otherwise appear unrelated. This mapping allows more detailed analysis of the activity of the Android system and correlation of evidence across layers. Moreover, the contextual alignment offered by MCP assists investigators in following behavioral chains between system components to enhance the reconstruct ability of application workflows, anomalous activity, and the roles played by various subsystems in reported forensic evidence.

Table 4. Expanded Mapping of Android Forensic Artifacts to MCP Context Dimensions

Reference	MCP Context Dimension	Representative Artifacts	Tools Used	Forensic Value / Challenges
[28]	Temporal Context	Logs, time stamps, usage history	Autopsy, Cellebrite	Allows reconstruction of timelines; however, timestamps may be lost or altered
[80]	Execution Context	Memory artifacts, process states, IPC	Volatility, LIME	Reveals runtime behavior; access limitations and volatility make it difficult to obtain
[37] [89]	Security Context	Permissions, encryption state, TEE data	Cellebrite, Oxygen	Provides insight into access control; restricted by secure hardware and encryption
[69]	Interaction Context	IPC logs, service calls, API usage	Autopsy, Andriller	Demonstrates communication between components; difficult to understand and correlate
[71], [72]	External Interaction Context	Network logs, cloud sync data	Wireshark, NetworkMiner	Records external communications; cloud collaboration and legal access may be necessary

Table 4 offers the mapping of Android forensic artifacts to the contextual dimensions of the MCP framework. The dimensions represent a different facet of the behavior of the system, and the structured

analysis of the evidence and the correlation of it across dimensions allow a better interpretation of the evidence by the forensic system.

5.3. Cross-Dimensional Evidence Correlation

The MCP model allows sharing of context in a structured way and provides coordination mechanisms, whereby the systems can combine information from multiple sources and still have a consistent contextual knowledge about various tasks [92]. Temporal artifacts enable the researcher to recreate the chronological order of occurrences; execution artifacts disclose the behavior of applications and processes when they run. Security artifacts give us an idea about permission states and platform protection mechanisms that determine how systems behave.

The way that system components connect via service calls and inter-process communication mechanisms is described by interaction artifacts. Artifacts of external interaction make the scope of investigation of the device broader as they record a record of network communication and activity on the cloud.

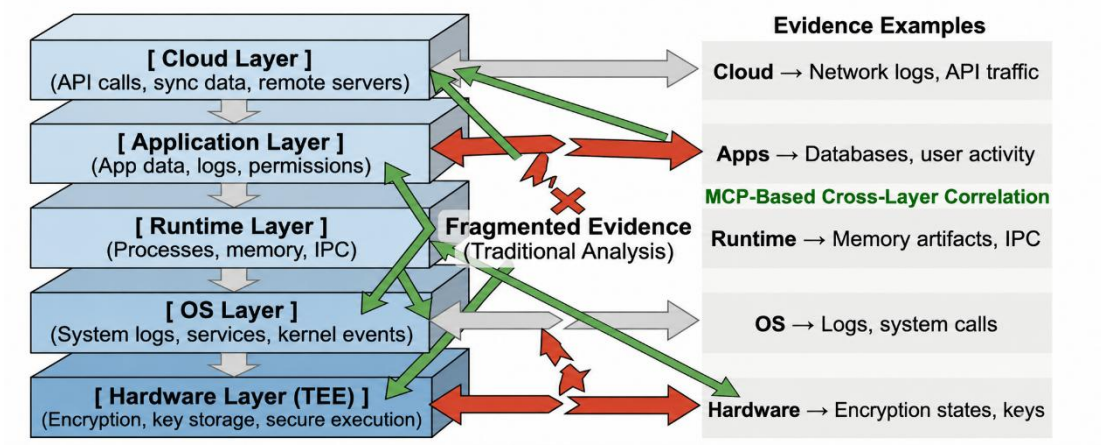


Figure 6. Cross-layer evidence fragmentation and MCP-based correlation.

Figure 6 shows how forensic evidence is often dispersed over several layers (posts) in contemporary Android systems, including applications, system services, runtime environments, and cloud components. These objects are examined independently by traditional forensic investigation techniques, which produce fractured or incomplete explanations. By enabling the structured cross-layer correlation of evidence, which will consider the temporal, execution, security, interaction, and external context dimensions to reconstruct consistent and comprehensive forensic information, the MCP-based solution will overcome this constraint.

Coupled with these artifacts, one can see a multidimensional picture of the Android system's behavior. As an example, a suspicious network connection detected in the external interaction context can be related to application activity detected in the execution context and permission use registered in the security context. Put together with timestamps in the temporal context, these artifacts can be used to help investigators reconstruct complicated behavioral patterns.

5.4. Positioning MCP inside Existing Android Forensic Approaches

As highlighted in Section II a majority of Android forensic research is founded on artifact-based investigative strategies. The approaches primarily focus on the extraction and investigation of persistent artifacts, including application databases such as SQLite databases [22] file system contents, system logs, configuration files, and network logs.

Several well-known mobile forensic investigation models, notably FORMOBILE and models based on the NIST mobile forensic guidelines, provide predetermined protocols for gathering, preserving, analyzing, and sharing evidence [45], [75]. A comparison is made in this section in reference to NIST SP 800-101 Rev. 1, Guidelines on Mobile Device Forensics, the appropriate NIST source for comparison when considering mobile device evidence acquisition, preservation, and examination. Does not apply to enterprise mobile device security management as covered in NIST SP 800-124 Rev. 2, Guidelines for Managing the Security of Mobile Devices in the Enterprise, which covers enterprise mobile device security management rather than forensic examination and is not the same comparison as presented here in Table 5, the "NIST Mobile Model". These models focus on methodological consistency and integrity of evidence, and they have been

significant towards standardization of mobile forensic investigation [93] . Practically, tool-centric analysis pipelines based on artifact extraction on specialized forensic software are also used in most Android investigations.

Although these methods offer useful tools of extraction and interpretation of forensic artifacts, they frequently examine each source of evidence separately. As a result, cross-layer analysis is usually performed by the investigators in an attempt to link evidence created by different parts of the system such as application activity and other operating system events, network events [94], [95].his inefficiency is further exacerbated in the current Android environment, where system events could occur across the execution states of run-time, hardware-backed security features, and third-party cloud services.

The suggested MCP model enhances the current state of Android forensics by offering a contextual paradigm. MCP does not regard artifacts as separate evidence sources but instead organizes forensic observations over numerous perspectives such as chronological ordering, execution state, security state changes, system interactions, and external communications. The multidimensional design provides a way to structure the correlation of heterogeneous evidence sources across layers of the Android system.

Table 5. Comparison of MCP Framework with Traditional Mobile Forensic Models

Feature	Description	NIST SP 800-101 Rev. 1 [45]	FORMOBILE	Artifact-Centric Analysis	MCP Framework
Evidence Acquisition	Mobile Device Digital Evidence Guidelines for Validation, Preservation, Acquisition, Examination, Analysis and Reporting	Provides guidance for validation, preservation and acquisition of digital evidence from mobile devices.	Standard forensic acquisition workflow	Supported through traditional extraction methods	Supports multi-layer and contextual acquisition
Artifact Analysis	System, application, and user-generated artifacts are examined for forensic insights.	Supports review and analysis of seized mobile-device evidence.	Fully supported	Core focus of the model	Enhanced with contextual correlation
Timeline Reconstruction	Reconstruction of user/system activity sequence based on collected evidence.	Allows for review and analysis of evidence for investigative interpretation.	Well supported process	Manually performed correlation	Automated and context-aware reconstruction
Cross-layer Evidence Correlation	Linking artifacts across applications, OS, network, and system layers.	Limited explicit support for cross-layer correlation	Limited support	Requires manual correlation	Structured and automated correlation mechanism
Runtime Behavior Analysis	Investigation of live system execution and dynamic behavior.	Limited explicit support for runtime behavior analysis	Limited support	Partially logs supported via	Fully supported via execution context modeling

context-aware Forensic Model	Utilizing contextual connections between artifacts to aid with interpretation.	No explicit context-aware forensic model	Not supported	Not supported	Core design principle of MCP framework
Automation Support	Level of automated analysis and processing of evidence.	Partially automated workflows	Semi-automated tools used	Mostly manual analysis	Highly automated pipeline
Scalability	The capacity to manage extensive forensic datasets and investigations.	Moderate scalability	Moderate scalability	Scalability issues brought on by manual work	High scalability with a hierarchical context model
Cloud Integration	The ability to integrate cloud-based sources of forensic evidence.	Limited guidance for cloud-based evidence	Partially supported	Restricted cloud artifact support	Cloud-aware design that is fully integrated
Modern Threat Suitability	The capacity to manage sophisticated malware, dynamic code loading, and runtime threats.	Limited adaptability to contemporary threats	Moderate adaption capacity for	Handling advanced threats in part	Strong adaptability to modifying threats

Table 5 highlights variations in evidence collection, artifact analysis, timeline reconstruction, cross-layer correlation, and support for contemporary threats in a comparison of mobile forensic frameworks, such as the NIST Mobile Model, FOR- MOBILE, Artifact-Centric Analysis, and the MCP [33].

6. Threat Coverage Beyond File-Based Malware

Modern Android threats have expanded beyond the traditional file-based malware to include runtime systems, cloud- based systems, and legal system services [98], [53]. These threats tend to reduce sustained artifacts and thus, they cannot be detected using the conventional artifact-based techniques of forensics. In that regard, this section focuses on examining these threat categories using the MCP analytical lens, demonstrating how its view of the context-based perspective can assist with better forensic interpretation of contemporary Android behaviors.

6.1. Runtime and File-less Threats

A growing type of threats to Android operates largely in volatile run times, not long-term storage [97], [106]. Other methods, such as dynamic code loading (DCL) [53], permit the applications to download executable payloads across remote servers and execute them in memory [99], [100].

These payloads are not written to disk, and the conventional artifact-based methods of forensics may be incapable of detecting them. Alternatively, evidence is supposed to be gathered based on runtime evidence like abnormal process behavior, dynamic loading of classes, and network communication.

MCP-wise, the threats imply the importance of the correlation of various contextual dimensions. The temporal dimension gives a list of all the orders of execution, the execution dimension illustrates the action being performed, and the external interaction dimension is communication with other remote servers. Individually, one may not consider such signals to be dangerous, but when put together they can provide large forensic usefulness to file-less attack habits.

6.2. Privacy-Invasive Applications: Spyware and Stalkerware

Real-World Context: Privacy-invasive Android apps are well recognized in the literature and in experimental studies. The initial spy software like *Tapsnake* proved to be capable of secret GPS tracking, sending the position information to remote servers [102], [103]. Further developed spyware families like *Hermit* have been seen gathering call logs, messages, es and location information by utilizing valid Android API [101], [104],[123]. Large- scale attacks such as Dark Caracal also showcase how mobile spyware is being used for surveillance, to track activities and steal sensitive information such as communication data, audio recordings, and location data [105]. These case studies show us that the advanced Android threats tend to be within the bounds of the law of the system, and that, therefore, it is hard to detect them by individual characteristics. Applicationsfor stalkerware and spyware usually take advantage of Android permissions such as ACCESS_FINE_LOCATION, READ_CONTACTS, and RECORD_AUDIO, in place with accessibility features and background services [55], [109],[118]. These actions, on the individual level, might seem harmless, but once taken together, they show traces of secret surveillance.

In the MCP view, this kind of behavior can be viewed in a variety of contextual dimensions. The security dimension can capture permission usage, the interaction dimension can capture system service access, the execution dimension can capture the persistent background activity, and the external interaction dimension can disclose the data transmission. The correlation of these dimensions will give a superior explanation of privacy-invasive behavior.

6.3. Implications for context-aware Forensics

The flaws in artifact-centric forensic procedures are further illustrated by runtime-based and privacy-invading attacks. Current Android threats spread their activity into several layers of the system, among which are a runtime process, system service, and external infrastructures [46].

Consequently, the study of artifacts in isolation frequently presents an incomplete or vague interpretation of those artifacts. The categories are an additional illustration of the need to consider Android observations in the context of how they are executed, secured, interacted with, time bound and externalized. This contextual interpretation can aid investigators in differentiating isolated observations from coordinated behaviors.

6.4. Case Study: Stalkerware Investigation Using MCP

As an example of the analytical feasibility of MCP in Android forensics, one can take a case representative based on behavioral trends of the spyware found in the real world, *Tapsnake*, Hermit, and campaigns, e.g., Dark Caracal.

Scenario: On an Android device, the software functions as a hidden application, gathering location data and sending it to a distant server without the user's knowledge.

Analysis: These artifacts might seem harmless when taken separately. But through the multidimensional approach of MCP, they constitute a logical trend of surveillance conduct. Temporal anomalies are signs of covert execution. The context of execution is used to demonstrate persistent activity, the sensitive dimension of permissions usage, and the external interaction shows possible data exfiltration.

By building the proposed multidimensional Android forensic contextual model, the observations can be arranged along the dimensions of the contexts and can be analyzed using the other dimensions. MCP can help facilitate access to the sources of these observations and the analytic tools available to obtain and analyze these observations.

6.5. Integration with Existing Forensic Tools

MCP can be well incorporated into the current Android forensic tools to increase the ability to perform the analysis. Conventional forensic tools like Cellebrite, Autopsy, and Oxygen mainly target the extraction of artifacts from devices such as file system data, application databases, logs, and network traces. Though they are effective in acquiring data, they do not usually have systems of systematic cross-layer correlation and context analysis.

The MCP integration brings about a context-based analytical layer working on top of the extracted artifacts. The MCP framework allows logical mapping and association of heterogeneous data origins by arranging evidence in various contextual dimensions, such as temporal, execution, security, interaction,

and external interaction. This will enable a more in-depth interpretation of forensic evidence, especially in new Android settings with distributed artefacts and run-to-run behavior.

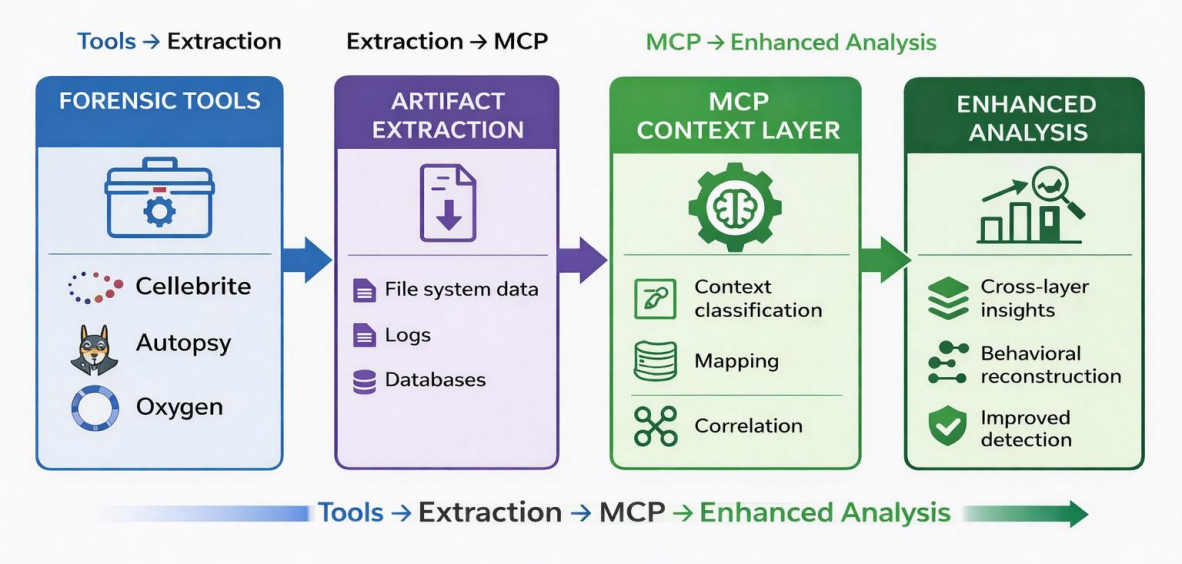


Figure 7. Integration of MCP with existing Android forensic tools.

Figure 7 shows MCP can be combined with the current Android forensic tools. Classical tools are used to pull out arti- facts on devices and then process them using the MCP context layer. The integration allows a systematized classification of context, cross-layer correlation, and improved forensic analysis through converting individual artifacts into context-sensitive knowledge.

All in all, this integration shows that MCP is not an alternative to the current forensic tools but rather an addition that offers another analysis layer which increases interpretability, facilitates behavioral reconstruction, and enhances the efficiency of present Android forensic investigations.

7. Research Challenges and Future Directions

Although Android forensics keeps advancing, the contemporary mobile setting offers various technical and analytical issues that constrain the efficacy of conventional investigation techniques. Applying more advanced analytical insights, such as context-driven techniques like MCP, is necessary because Android systems are becoming more complicated and threat models are changing.

7.1. Technical Challenges

It is one of the main problems of Android forensics because it is difficult to obtain reliable runtime evidence. A lot of the current threats, like fileless malware and just-in-time (JIT) compiled code, are present in the volatile memory and might not leave any traces[128],[129]. This information is hard to collect due to process isolation, sand-boxing, and restricted system access.

The other factor that makes it hard is the adoption of encryption and other hardware-supported security features. Data sources such as file encryption, key store, and trusted execution environments (TEE) can potentially deny access to critical information, even if the device is accessible to forensic investigators. While this is good for the security of the user, it makes it difficult to see and extract the evidence.

Also, the variety of data sources from applications, system logs, network and cloud data sources result in data integration issues [107]. Current forensic tools do not correlate analysis of these sources, and it is difficult to get a full picture of system operations [108].

7.2. Analytical Challenges

Along with technical constraints, there are major difficulties with the interpretation of forensic evidence. Modern Android settings tend to break evidence into various layers of the system and over different periods of time. Consequently, the analysis of individual artifacts might seem incomplete or misleading.

The existing forensic techniques are often based on manual evidence correlation, which is highly subjective and does not necessarily perform well when complex investigations are concerned. Besides, it is

still hard to differentiate between benign and malicious behavior, especially when applications access privileged permissions and system services to execute privacy-invasive operations .

In that regard, theories like MCP provide a theoretical ground for how to structure evidence at various dimensions. Nonetheless, such multidimensional analysis is not yet practically implemented, especially in the areas of automation and tools assistance.

7.3. Legal and Privacy Considerations

The data used in modern forensic investigations is becoming increasingly tied to the device, with the inclusion of cloud services and remote servers. This creates the problem of jurisdiction since the appropriate evidence can be dispersed in various legal systems and geographic locations [90] [91].

There are also some serious privacy issues with context- driven analysis. A thorough forensic investigation might include looking at private user data like location history, behavioral patterns, and private messages. It's important to make sure that this kind of analysis doesn't break any laws or ethical principles.

7.4. Future Research Directions

Future studies should concentrate on creating tools and methods that could support context-based forensic investigations. This includes improving techniques for capturing run-time evidence, helping us to combine different types of data, and automatic correlation of cross-layer evidence [122].

Machine learning and graph methods can potentially under- exploit the analysis of complex relationships among forensic artifacts. These methods can be used to facilitate more efficient forensic investigations.

Similarly, it is needed to explore the practical usage of the MCP-based models in forensic investigations. The interaction between theoretical models and investigative tools will help in building mobile forensic capabilities.

Finally, future research should focus on combining device and cloud evidence, and also develop privacy-aware forensic approaches that will balance between investigative demands and users' privacy.

7.5. Limitations of This Study

There are a few limitations that are mentioned in this study. To begin with, the suggested MCP usage is only conceptual and lacks full-scale implementation in a real-world forensic tool or pipeline. Consequently, the empirical validity of the practical performance of the framework and its scalability is still to be established.

Second, the discussion given in this publication is found mostly on literature review and theoretical mapping of forensic artifacts to contextual dimensions. There has been no experimental assessment or quantitative benchmarking done on the effectiveness of MCP in comparison to the current forensic techniques.

Third, the case study presented in this paper is not demonstrative and is not based on a large-scale or non-homogeneous dataset of real-life Android forensic cases. This restricts the external validity of the results to other device models, Android versions, and threat scenarios.

Nevertheless, these shortcomings do not imply that the study does not offer a background view of context-driven forensic analysis and offer potential avenues of future research, especially in the creation of automated tools and empirical validation of MCP-based methods.

8. Conclusion

Android digital forensics has changed over time, and the constraints of the classic artifact- based approaches to investigations have become apparent in the environment of current mobile features. Due to the ongoing deployment of more powerful security systems, runtime enabled models of system execution, and cloud-integrated architectures, forensic evidence is becoming progressively more fragmented and distributed across many layers of the system and over time. These changes disrupt traditional methodology based mainly on the persistence of artifacts and techniques of isolated analysis.

This paper has studied the use of MCP to support a context-based approach to Android forensic analysis. The proposed Android forensic contextual model arranges heterogeneous sources of evidence by

linking observations in the temporal, execution, security, interaction and external-interaction contexts, and the MCP offers an interoperability mechanism to access the relevant forensic resources and analysis tools.

The discussion of modern threat categories, such as runtime- based and privacy-invasive programs, including spyware and stalkerware, supports the appropriateness of context-based methods for the current investigations. The given case study also illustrates how MCP can facilitate the interpretation of distributed forensic evidence by revealing coordinated behaviors that may otherwise be hidden during the examination of individual artifacts.

Irrespective of its potential, there are a few challenges. The effective application of the context-driven analysis presupposes the development of runtime evidence acquisition, cross- layer data processing and automated correlation methods. Moreover, legal and privacy issues should be considered very closely because the use of such sensitive and distributed sources of data as the basis of forensic investigation is becoming more widespread.

The brainstorm of future research is to devise the tools and methodology to put context-centric models like MCP into practice in a real-world forensic process. These kinds of efforts help make Android forensic investigations perform better in more sophisticated settings by bridging the gap between abstract models and their real-world use.

References

1. Android Open Source Project, "Android Security -- Overview," February 2026. [Online]. Available: https://source.android.com/docs/security/overview.
2. C. Ferreira, M. Lopes, L. Correia, C. Wanzeller, F. Sa, P. Martins and M. Abbasi, "Database Performance on Android Devices, A Comparative Analysis," in Marketing and Smart Technologies: Proceedings of ICMarkTech 2022, Singapore, Springer, 2023, p. 261–273.
3. S. Kumar and S. K. Shukla, "The State of Android Security," in Cyber Security in India, Springer, 2020, p. 17–22.
4. B. Pribadi, S. Rosdiana and S. Arifin, "Digital forensics on facebook messenger application in an android smartphone based on nist sp 800-101 r1 to reveal digital crime cases," in Proceedings of the International Conference on Computer Science and Computational Intelligence (ICCSCI 2022), 2022.
5. T. Moses, A. L. Bolaji and I. W. Andrew, "An Android Location-Based Crime Reporting System Using the Google Map API," University of Pitesti Scientific Bulletin: Electronics and Computers Science, vol. 20, no. 1, 2020.
6. M. Tiwari and S. Thapaliya, "Technical challenges of encryption in mobile forensics," International Journal of Multidisciplinary and Innovative Research, vol. 2, no. 6, p. 178–183, June 2025.
7. B. Fakiha, "Unlocking digital evidence: Recent challenges and strategies in mobile device forensic analysis," Journal of Information Systems and Information Security, 2024.
8. A. F. A. Kadir, A. H. Lashkari and M. D. Firoozjaei, "Understanding Cybersecurity on Smartphones: Challenges, Strategies, and Trends," 2024.
9. D. Podlesnykh, "User acquisition using android privacy sandbox," Theseus Repository, 2025.
10. K. Kaushik, M. Ouaissa and A. Chaudhary, Advanced Techniques and Applications of Cybersecurity and Forensics, Boca Raton, FL: CRC Press, 2024.
11. M. Zhang, S. Wang, G. Zheng, Y. Zhao and H. Wang, "Demystifying cookie sharing risks in webview-based mobile app-in-app ecosystems," arXiv preprint arXiv:2510.24141, 2025.
12. K. V. K. Raju, D. Akula, S. S. Areti, B. K. Barre and S. S. K. R. Bhupathiraju, Android malware detection using dynamic analysis, 1 ed., Boca Raton, FL: CRC Press, 2025, pp. 1-8.
13. M. Shukla, A. Sharma and S. Mandal, "Challenges in android forensics: Challenges and complexities in mobile forensics—a comprehensive investigation," in Recent Advances in Computing Sciences, 1 ed., Boca Raton, FL, CRC Press, 2025, pp. 1-4.
14. K. Chitrashri, K. C. Gull, V. SushilaDevi, V. B. Sansktruti and P. K. Korlahalli, "A review on mobile forensic needs, challenges, approaches, process model, tools and standard," in ICT Systems and Sustainability, vol. 1650, Cham, Springer, 2026.
15. J. Bellizzi, "Responding to stealthy attacks on android using timely-captured memory dumps," University of Malta, 2023.
16. T. Vidas, C. Zhang and N. Christin, "Toward a General Collection Methodology for Android Devices," Digital Investigation, vol. 8, no. S1, July 2011.
17. C. C.-C. Cheng, C. Shi, N. Z. Gong and Y. Guan, "EviHunter: Identifying Digital Evidence in the Permanent Storage of Android Devices via Static Analysis," arXiv preprint arXiv:1808.06137, 2018.
18. T. K. Hazra, S. Ghosh and A. Kumar, "Challenges in Android Forensics," in Proceedings of the International Conference on Computing and Communication Systems (ICCCS 2017), 2017.
19. R. Sheth, K. Kaushik, C. Parekha and N. Chayal, Android and iOS Mobile Forensics: Leveraging Blockchain, Machine Learning, and Deep Learning for Digital Investigations, Boca Raton, FL: CRC Press, 2025.
20. J. Park, J. Park, H. Kim, S. Kang and J. Kim, "A Comprehensive Artifact Analysis of Google Applications on Android and iOS Platforms," Forensic Science International: Digital Investigation, vol. 36, p. 302029, 2025.
21. S. Yoo, Y. Park and D. Jeong, "Preserving Evidence Integrity in Full File System Extractions: Assessing Hash Inconsistencies and Potential Solutions," SSRN Electronic Journal, June 2025.
22. E. Daraghmi, Z. Qaroush, M. Hamdi and O. Cheikhrouhou, "Forensic Operations for Recognizing SQLite Content (FORC): An Automated Forensic Tool for Efficient SQLite Evidence Extraction on Android Devices," Applied Sciences, vol. 13, no. 19, p. 10736, 2023.
23. A. Boztas, J. De Jong and C. Hadjigeorghiou, "Argus: A New Approach for Forensic Analysis of Apps on Mobile Devices," Forensic Science International: Digital Investigation, vol. 36, p. 301938, 2025.
24. R. Sheth, K. Kaushik, C. Parekha and N. Chayal, "Examination of Social Media and Messaging Applications," in Android and iOS Mobile Forensics, Apress, 2026.

25. P. Kumar, R. Narayan and E. Rasid, "Android Forensics: A Literature Review of Methodologies and Tool Efficacy," International Journal of Advanced Networking and Applications, vol. 17, no. 4, p. 7045–7054, 2026.
26. A. Almuqren, H. Alsuwaelim, M. M. H. Rahman and A. A. Ibrahim, "A Systematic Literature Review on Digital Forensic Investigation on Android Devices," Procedia Computer Science, vol. 235, p. 1332–1352, 2024.
27. X. Zhang, C. Z. Liu, K.-K. R. Choo and J. A. Alvarado, "A Design Science Approach to Developing an Integrated Mobile App Forensic Framework," Computers & Security, vol. 109, p. 102226, 2021.
28. F. Breitinger, H. Studiawan and C. Hargreaves, "SoK: Timeline Based Event Reconstruction for Digital Forensics: Terminology, Methodology, and Current Challenges," Forensic Science International: Digital Investigation, vol. 53, p. 301932, 2025.
29. R. Adila, H. Studiawan and F. Breitinger, "Forensic event reconstruction with process mining," IEEE Access, 2026.
30. X. Lin, T. Chen, T. Zhu, K. Yang and F. Wei, "Automated forensic analysis of mobile applications on Android devices," Digital Investigation, vol. 26, no. Supplement, 2018.
31. K. Barmpatsalou, T. Cruz, E. Monteiro and P. Simões, "Current and future trends in mobile device forensics: A survey," ACM Computing Surveys, vol. 51, no. 3, p. 1–31, 2018.
32. J. Grover, "Android Forensics: Automated Data Collection and Reporting from a Mobile Device," Digital Investigation, vol. 10, no. S1, p. S12–S20, 2013.
33. X. Hou, Y. Zhao, S. Wang and H. Wang, "Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions," ACM Transactions on Software Engineering and Methodology, 2026.
34. P. Kumar, E. Rashid and R. Narayan, "A Comparative Study of Mobile Forensic Tools for Android Devices," in Advances in Intelligent Systems Research / Atlantis Press Proceedings, Atlantis Press, 2025.
35. A. Diallo, "Security of Android Applications in Developing Regions," Université du Luxembourg, Luxembourg, 2026.
36. Android Open Source Project, "Android security features," 2026. [Online]. Available: https://source.android.com/docs/security/features.
37. R. Mayrhofer, J. V. Stoep, C. Brubaker, D. Hackborn, B. Bonne, G. S. Tuncay, R. P. Jover and M. A. Specter, "The android platform security model," arXiv preprint arXiv:1904.05572, 2019.
38. Google Developers, "Android Security and Privacy," [Online]. Available: https://developers.google.com/android/security. [Accessed 2026].
39. Android Developers, "Privacy changes in Android 11 — storage," 2026. [Online]. Available: https://developer.android.com/about/versions/11/privacy/storage.
40. A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin and D. Wagner, "Android permissions: User attention, comprehension, and behavior," 2012.
41. Android Open Source Project, "File-based encryption," 2026. [Online]. Available: https://source.android.com/docs/security/features/encryption/file-based.
42. D. B. Oh, S. Lim, S. Lee, Y. Jo, G. Choi, B. Kim and H. K. Kim, "Forensic analysis and evaluation of file-wiping applications in android os," Journal of Forensic Sciences, vol. 71, no. 1, p. 338–352, 2026.
43. H. Deng, G. S. Tuncay, A. Acar, E. Luques, H. Öz, A. Aris and S. Uluagac, "Unveiling the global landscape of android security updates," IEEE Transactions on Dependable and Secure Computing, p. 1–18, 2026.
44. Android Open Source Project, "Trusty — Android Trusted Execution Environment," 2026. [Online]. Available: https://source.android.com/docs/security/features/trusty.
45. S. B. W. J. Richard Ayers, "Guidelines on Mobile Device Forensics," NIST SP 800-101 Rev. 1, Gaithersburg, MD, 2014.
46. P. Faruki, A. Bharmal, V. Laxmi, V. Ganmoor, M. S. Gaur, M. Conti and M. Rajarajan, "Android Security: A Survey of Issues, Malware Penetration, and Defenses," IEEE Communications Surveys & Tutorials, vol. 17, p. 998–1022, 2015.
47. Kaspersky, "Mobile malware evolution 2024: Kaspersky security bulletin," 2025.
48. A. Fukami, R. Stoykova and Z. Geradts, "A new model for forensic data extraction from encrypted mobile devices," Forensic Science International: Digital Investigation, vol. 37, p. 301189, 2021.
49. C. C.-C. Cheng, C. Shi, N. Z. Gong and Y. Guan, "LogExtractor: Extracting Digital Evidence from Android Log Messages via String and Taint Analysis," Forensic Science International: Digital Investigation, vol. 37, no. Supplement, p. 301193, 2021.
50. J. L. George, "Identifying threats in android: Detection techniques for malicious accessibility service exploits," International Journal for Multidisciplinary Research, vol. 7, no. 1, 2025.

51. B. Patel and P. S. Mann, "A survey on mobile digital forensic: Taxonomy, tools, and challenges," Security and Privacy (SPY), vol. 8, no. 2, p. e470, 2024.
52. A. M. M. Soares and R. T. de Sousa Junior, "Forensic analysis of android runtime (ART) application heap objects in emulated and real devices," in Information Systems Security and Privacy, vol. 867, Springer, 2018, p. 130–147.
53. Z. Qu, S. Alam, Y. Chen, X. Zhou, W. Hong and R. Riley, "DyDroid: Measuring Dynamic Code Loading and Its Security Implications in Android Applications," in 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2017.
54. K. Krinkin, V. Dopira, O. Kochneva, S. Petrov and M. Kopylov, "Android memory inspection techniques and tools," in Proceedings of the Conference of Open Innovations Association FRUCT, Yaroslavl, Russia, 2020.
55. M. Jadhav, W. Bao and V. Bindschaedler, "Thwart me if you can: An empirical analysis of android platform armoring against stalkerware," arXiv preprint arXiv:2508.02454v1, 2025.
56. C. Anglano, "Review of Mobile Surveillanceware: Capabilities, Countermeasures, and Research Challenges," Electronics, vol. 14, no. 14, p. 2763, 2025.
57. Y. Leguesse, M. Vella, C. Colombo and J. Hernandez-Castro, "Reducing the forensic footprint with android accessibility attacks," Security and Trust Management, 2020.
58. S. Surakanti, S. Goundar and J. Dwight, "Countering anti-forensic tactics in cybercrime investigations – a systematic literature review," International Journal of Information Security, vol. 24, p. 210, 2025.
59. A. Bartel, J. Klein, M. Monperrus and Y. L. Traon, "Static analysis for extracting permission checks of a large scale framework: The challenges and solutions for analyzing android," arXiv preprint arXiv:1408.3976, 2014.
60. Q. Duan and Z. Lu, "AI agent communications in the future internet—paving a path toward the agentic web," Future Internet, vol. 18, no. 3, p. 171, 2026.
61. Model Context Protocol Project, "Introduction — getting started with the model context protocol," 2026. [Online]. Available: https://modelcontextprotocol.io/docs/getting-started/intro.
62. A. Singh, A. Ehtesham, S. Kumar and T. T. Khoei, "A survey of the model context protocol (MCP): Standardizing context to enhance large language models (LLMs)," Preprints, 2025.
63. N. Elenkov, Android Security Internals: An In-Depth Guide to Android's Security Architecture, San Francisco, CA, 2014.
64. S. Bhandari and V. Jusas, "An abstraction based approach for reconstruction of timeline in digital forensics," Symmetry, vol. 12, p. 104, 2020.
65. A. Kumar, K. Puja, A. Kumar, S. Kashyap, Anupama and L. Kumari, "Intelligent hybrid android malware detection through static, dynamic, and machine learning analysis," International Journal of Advanced Digital Systems and Multidisciplinary Studies (IJADSMS), vol. 2, no. 6, p. 131–140, 2026.
66. N. Asiago Ondara, "Introduction to security," in Beginning Linux SysAdmin, Berkeley, CA, Apress, 2025.
67. R. Sheth, K. Kaushik, C. Parekha and N. Chayal, "Android Forensics Fundamentals," in Android and iOS Mobile Forensics, Apress, 2025.
68. G. Meng, R. Feng, G. Bai and S. Li, "DroidEcho: An In-Depth Dissection of Malicious Behaviors in Android Applications," Cybersecurity, vol. 1, no. 4, 2018.
69. S. Bhandari, W. B. Jaballah, V. Jain, V. Laxmi, A. Zemmari, M. S. Gaur, M. Mosbah and M. Conti, "Android Malware Detection Using Machine Learning Techniques: A Survey," Computers & Security, vol. 72, p. 57–76, 2017.
70. G. Kirubavathi, B. P. Mayuri and S. Pranathasree, "Ensemble machine learning for proactive android ransomware detection using network traffic," Scientific Reports, 2026.
71. J. Lu, J. Chen, S. Cai, Y. Liu and S. Wang, "A novel android malware classification approach based on multi-scale feature fusion for encrypted traffic," Computers & Security, 2026.
72. S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. L. Traon, D. Octeau and P. McDaniel, "FlowDroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps," in Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), 2014.
73. I. Al-Oqily, B. Khalaf and M. Al-Fayez, "Deconstructing android messaging clones: A hybrid reverse engineering and risk assessment framework," Cluster Computing, vol. 29, p. 156, 2026.
74. A. Hoog, "Android Forensic Techniques," in Android Forensics, Syngress, 2011.
75. Bhushan, S., Saha, S., Lamkuche, H. S., Samara, G., & Itmazi, J. (2026). Behavioral Profiling of Android Malwares: A Dynamic Analysis Approach. In Business Resilience and Business Innovation for Sustainability: The Double-Edged Role of Artificial Intelligence and Other Disruptive Technologies (pp. 1851-1871). Cham: Springer Nature Switzerland.

76. A. Prasad, S. Choudhury, B. Paul and J. Pande, "Advancement in Cyber Security and Digital Forensics: Proceedings of ICACSDF 2025," in Springer, 2025.
77. Ö. Günay, B. Gül and F. Ertam, "Comparative Analysis of Digital Forensics Methods on Android Devices," Firat University Journal of Experimental and Computational Engineering, vol. 5, no. 1, 2026.
78. A. Aljahdali, N. Alsaidi, M. Alsafri, A. Alsulami and T. Almutairi, "Mobile Device Forensics," Romanian Journal of Information Technology and Automatic Control, vol. 31, no. 3, p. 81–96, 2021.
79. Sakshi, A. Malik and A. K. Sharma, "Forensic Analysis of Hidden Artifacts: A Comprehensive Investigation Framework for Smart Wearables," Journal of Forensic Sciences, p. 1–16, 2026.
80. J. Sylve, A. Case, L. Marziale and G. G. Richard, "Acquisition and Analysis of Volatile Memory from Android Devices," Digital Investigation, vol. 8, p. S175–S184, 2012.
81. N. Shantaram, "Forensic Analysis of Navigation Applications on Android and iOS Platforms," Purdue University, West Lafayette, IN.
82. E. A. Abanga, "Evidence Based Reconstruction for Digital Forensics," in Research Nexus in IT, Law, Cyber Security & Forensics, Research Nexus, 2024, p. p. 185.
83. T. Sutter, T. Kehrer, M. Rennhard, B. Tellenbach and J. Klein, "Dynamic Security Analysis on Android: A Systematic Literature Review," IEEE Access, 2024.
84. P. H. Nguyen, C. Turkay, G. Andrienko and N. Andrienko, "Understanding User Behaviour through Action Sequences: From the Usual to the Unusual," IEEE Transactions on Visualization and Computer Graphics, 2018.
85. M. Ahmed and M. N. A. Khan, "A Review of Forensic Analysis Techniques for Android Phones," International Journal of Computer Applications.
86. S. L. Garfinkel, "Digital Forensics Research: The Next 10 Years," Digital Investigation, p. S64–S73, 2010.
87. S. Khalid and F. B. Hussain, "VolMemDroid—Investigating Android Malware Insights with Volatile Memory Artifacts," Expert Systems with Applications, p. 124347, 2024.
88. R. Mayrhofer, "An Architecture for Secure Mobile Devices," Security and Communication Networks, vol. 8, no. 10, p. 1921–1935, 2014.
89. S. O'Shaughnessy and A. Keane, "Impact of Cloud Computing on Digital Forensic Investigations," Advances in Digital Forensics IX, vol. 410, p. 291–303, 2013.
90. G. Lin, Z. Cui and X. Chen, "Detecting API Compatibility Issues of Android Applications Based on Screen Transition Graphs," Empirical Software Engineering, vol. 31, p. 77, 2026.
91. N. Krishnan, "Advancing Multi-Agent Systems Through Model Context Protocol: Architecture, Implementation, and Applications," arXiv preprint arXiv:2504.21030, 2025.
92. M. M. Hasan, H. Li, E. Fallahzadeh and B. Adams, "Model Context Protocol (MCP) at First Glance: Studying the Security and Maintainability of MCP Servers," arXiv preprint arXiv:2506.13538, 2025.
93. A. Al-Dhaqm, S. A. Razak, R. A. Ikuesan, V. R. Kebande and K. Siddique, "A Review of Mobile Forensic Investigation Process Models," IEEE Access, vol. 8, p. 173359–173375, 2020.
94. J. Ingemarsson, E. Andersson, J. Kävrestad and M. Birath, "A Socio-Technical Perspective on Forensic Challenges in Smartphones and Smartwatches: A Systematic Literature Review," in CEUR Workshop Proceedings, 2024.
95. B. K. Sharma, M. Walia, Y. Sharma and M. T. A. Beig, "Advances and Challenges in Mobile Phone Forensics," in 2024 International Conference on Communication, Computer Sciences and Engineering (IC3SE), 2024.
96. D. Delija, D. Sudec, G. Sirovatka and M. Žagar, "How to Do a Forensic Analysis of Android 11 Artifacts," in How to Do a Forensic Analysis of Android 11 Artifacts, Opatija, Croatia, 2022.
97. I. Kara, "Fileless Malware Threats: Recent Advances, Analysis Approach through Memory Forensics and Research Challenges," Expert Systems with Applications, vol. 214, pp. 1-20, 2022.
98. F. S. Alsubaei, A. A. Almazroi and W. S. Atwa, "BERT Ensemble Based MBR Framework for Android Malware Detection," Scientific Reports, p. 14027, 2025.
99. R. Bhan, R. Pamula, K. S. Kumar, N. K. Jyotish, P. C. Tripathi, P. Faruki and J. Gajrani, "DLCDroid: An Android Apps Analysis Framework to Analyse the Dynamically Loaded Code," Scientific Reports, vol. 15, no. 1, 2025.
100. A. I. Aysan, F. Sakiz and S. Sen, "Analysis of Dynamic Code Updating in Android with Security Perspective," IET Information Security, vol. 13, p. 269–277, 2019.
101. M. Qabalin, M. N. Aldalaien and M. Alkasassbeh, "Android Spyware Detection Using Machine Learning: A Novel Dataset," Sensors, vol. 22, no. 15, p. 5765, 2022.
102. S. Ma, Z. Tang, Q. Xiao, J. Liu, T. T. Duong, X. Lin and H. Zhu, "Detecting GPS Information Leakage in Android Applications," in 2013 IEEE Global Communications Conference (GLOBECOM), 2013.

103. M. Khedkar, A. K. Mondal and E. Bodden, "A Study of Privacy-Related Data Collected by Android Apps," Automated Software Engineering, vol. 33, no. 2, 2026.
104. Lookout Threat Lab , Google Threat Analysis Group, "Hermit Spyware Analysis," [Online]. Available: https://blog.google/threat-analysis-group/hermit-spyware/.
105. L. Sugiura, M. Button and D. Blackbourn, "The Technification of Domestic Abuse: Methods, Tools and Criminal Justice Responses," Criminology & Criminal Justice, vol. 22, 2024.
106. S. L. Sanna, D. Maiorca and G. Giacinto, "An Explainable Memory Forensics Approach for Malware Analysis," arXiv:2602.19831, 2026.
107. J. Awoleye, S. Mavire, A. Munyira and K. Magora, "Forensic Analysis Frameworks for Encrypted Cloud Storage Investigations," International Journal of Computer Applications, vol. 187, no. 1, p. 8–19, 2025.
108. A. A. Ahmed, K. Farhan, M. I. H. Ninggal and G. Alselwi, "Retrieving and Identifying Remnants of Artefacts on Local Devices Using Sync.com Cloud," Sensors, vol. 25, p. 106, 2025.
109. F. Akbar, M. Hussain, R. Mumtaz, Q. Riaz, A. W. Wahab and K. Jung, "Permissions-Based Detection of Android Malware Using Machine Learning," Symmetry, vol. 14, no. 4, p. 718.
110. Google LLC, "Android security paper 2024," 2024. [Online]. Available: https://services.google.com/fh/files/misc/android-security-paper-2024.pdf.
111. V. S. Makarov, "Features of forensic examination of mobile devices running on the android operating system," Bulletin of Kharkiv National University of Internal Affairs, vol. 111, no. 4, p. 378–384, 2025.
112. T. Siameh, A. A. Addobea, C.-H. Liu and E. K. Fiah, "A Systematic Security Analysis of Model Context Protocol: Vulnerabilities, Exploits, and Mitigations," in 2026 IEEE 5th International Conference on AI in Cybersecurity (ICAIC), Houston, TX, USA, 2026.
113. J.-N. Hilgert, C. Jakobs, M. Külper, M. Lambertz, A. Mahr and E. Padilla, "Chances and Challenges of the Model Context Protocol in Digital Forensics and Incident Response," arXiv preprint arXiv:2506.00274, 2025.
114. A. Ghaffari, "What are the Current Trends and Challenges in Digital Forensic Analysis of Mobile Applications?: A Systematic Literature Review," Uppsala University, 2025.
115. M. Moreb, Practical Forensic Analysis of Artifacts on iOS and Android Devices: Investigating Complex Mobile Devices, Berkeley, CA: Apress, 2022.
116. E. Kovacs, "Android 17 Beta Strengthens Secure-by-Default Design for Privacy and App Security," 2026. [Online].
117. Tamma, R., Skulkin, O., Mahalik, H., & Bommisetty, S. (2020). Practical mobile forensics: Forensically investigate and analyze iOS, Android, and Windows 10 devices. Packt Publishing Ltd.
118. T. Sutter, "Simple spyware: Android's invisible foreground services and how to (ab)use them," arXiv preprint arXiv:2011.14117, December 2019.
119. Center for Infrastructure Assurance and Security, "Memory forensics–guided execution reconstruction for android devices," 2026. [Online]. Available: https://cina.gmu.edu/projects/memory-forensics-guided-execution-reconstruction-for-android-devices-2/.
120. C. Wang and L. Wang, "Computer software malicious intrusion detection method based on cn and dynamic and static behavior analysis," Security and Privacy, vol. 9, no. 2, p. e70203, 2026.
121. R. Pan, Y. Liang, L. Li, H. Du, T.-W. Kuo and C. J. Xue, "Efficient Hash-Free Mobile Cloud Backup via Operation-Log Versioning," ACM Transactions on Storage, 2024.
122. R. Kaur, M. Athar and K. Kaur, "Review of enhancing mobile forensic analysis with artificial intelligence," vol. 12, p. 34–46, 2025.
123. A. Caruso, "Forensic Analysis of Mobile Spyware: Investigating Security, Vulnerabilities, and Detection Challenges in Android and iOS Platforms," Politecnico di Torino, 2024.
124. Y. Salem, M. Owda and A. Y. Owda, "Towards Digital Forensics 4.0: A Multilevel Digital Forensics Framework for Internet of Things (IoT) Devices," International Journal of Wireless and Microwave Technologies, vol. 14, no. 2, p. 27–54, 2024.
125. S. Kim, J. Jung, H. Kang, Y. Yoon, S. Cho, M. Park and S. Han, "An Effective Automotive Forensic Technique Utilizing Various Logs of Android-Based In-Vehicle Infotainment Systems," Forensic Science International: Digital Investigation, p. 301990, 2025.
126. Oh, J., Lee, S., & Hwang, H. (2024). Forensic detection of timestamp manipulation for digital forensic investigation. IEEE Access, 12, 72544-72565.
127. D. Kim and S. Lee, "Study of Identifying and Managing the Potential Evidence for Effective Android Forensics," Forensic Science International: Digital Investigation, vol. 33, p. 200897, 2020.

128. N. Singh and S. Tripathy, "Unveiling the Veiled: An Early Stage Detection of Fileless Malware," Computers & Security, p. 104231, 2024.
129. Y. Dehfouli and A. H. Lashkari, "Memory Analysis for Malware Detection: A Comprehensive Survey Using the OSCAR Methodology," ACM Computing Surveys, vol. 58, no. 4, p. 1–58, 2025.
130. H. Nyholm, K. Monteith, S. Lyles, M. Gallegos, M. DeSantis, J. Donaldson and C. Taylor, "The Evolution of Volatile Memory Forensics," Journal of Cybersecurity and Privacy, vol. 2, p. 556–572, 2022.
131. Blessing, J., Anderson, R. J., & Beresford, A. R. (2025). Keydroid: A large-scale analysis of secure key storage in android apps. arXiv preprint arXiv:2507.07927.
132. A. Muñoz, R. Ríos, R. Román and J. López, "A Survey on the (In)Security of Trusted Execution Environments," Computers & Security, vol. 129, p. 103180, 2023.
133. Kanakasabapathi RS and Judith JE. A Comprehensive Analysis of Cloud data storage and security: A Narrative Review. Premier Journal of Science 2025;15:100206.