

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки**

«На правах рукопису»
УДК _____

«До захисту допущено»
Завідувач кафедри
_____ Дмитро ЛАНДЕ
(підпис)
«_____» _____ 2025 р.

Магістерська дисертація

на здобуття ступеня магістра

**за освітньо-професійною програмою «Системи, технології та
математичні методи кібербезпеки»
спеціальності 125 «Кібербезпека та захист інформації»**

на тему: Дослідження автентичності даних мобільних пристроїв.

Виконав (-ла): здобувач вищої освіти II курсу, групи ФБ-42мп
(шифр групи)

Мартиненко Денис Олександрович
(прізвище, ім'я, по батькові) (підпис)

Керівник к.т.н., доцент кафедри ІБ Коломицев М. В.
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент к.т.н., асистент кафедри інформаційних технологій

Мазуренко Роман Володимирович
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цій магістерській дисертації
немає запозичень з праць інших авторів без
відповідних посилань.
Здобувач вищої освіти _____
(підпис)

Київ – 2025 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки
Рівень вищої освіти – другий (магістерський)
Спеціальність – 125 «Кібербезпека та захист інформації»
Освітньо-професійна програма «Системи, технології та математичні методи
кібербезпеки»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Дмитро ЛАНДЕ
(підпис)

«__» _____ 2025 р.

ЗАВДАННЯ
на магістерську дисертацію здобувачу вищої освіти

Мартиненку Денису Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи:

“Дослідження автентичності даних мобільних пристроїв.”

керівник роботи Коломицев Михайло Володимирович, кандидат наук,
доцент кафедри інформаційної безпеки

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «4» листопада 2025 р. № 4797-с

2. Термін подання здобувачем вищої освіти роботи «12» грудня 2025 р.

3. Вихідні дані до роботи

Науково-технічна література з цифрової криміналістики та кібербезпеки;
міжнародні стандарти; технічна документація Android; методичні матеріали й
виробників forensic-інструментів; набори даних і резервних копій; довідкові
матеріали з криптографічних геш-функцій та перцептивних хешів;
нормативно-правові акти щодо обігу цифрових доказів;

4. Зміст роботи

Аналіз нормативних підходів і стандартів digital forensics та особливостей архітектури Android у контексті джерел цифрових артефактів. Обґрунтування та розроблення rule-based методики перевірки автентичності з системою індикаторів і правилами скорингу. Опис програмної реалізації прототипу мовою Python, форматів даних та звітності. Проведення експериментальних досліджень на контрольних сценаріях підміни/маніпуляцій.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): Презентація до захисту магістерської дисертації

6. Дата видачі завдання 1 вересня 2025 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	01.09.2025	ВИКОНАНО
2	Огляд наукової літератури	01.09.2025 - 25.09.2025	ВИКОНАНО
3	Огляд міжнародних стандартів	26.09.2025 - 01.10.2025	ВИКОНАНО
4	Огляд архітектури Android у контексті цифрової криміналістики	02.10.2025 - 10.10.2025	ВИКОНАНО
5	Написання теоретичного блоку роботи	11.10.2025 - 21.11.2025	ВИКОНАНО
6	Формування концепції rule-based підходу	22.11.2025 - 01.12.2025	ВИКОНАНО
7	Формування системи індикаторів	02.12.2025 - 08.12.2025	ВИКОНАНО
8	Написання програмного прототипу перевірки автентичності даних	09.12.2025 - 14.12.2025	ВИКОНАНО
9	Підготовка до попереднього захисту та захист магістерської дисертації	15.12.2025 - 18.12.2025	ВИКОНАНО
10	Підготовка до захисту та захист магістерської дисертації	19.12.2025 - 25.12.2025	ВИКОНАНО

Здобувач вищої освіти

(підпис)

Денис МАРТИНЕНКО

(Власне ім'я, ПРІЗВИЩЕ)

Керівник роботи

(підпис)

Михайло КОЛОМИЦЕВ

(Власне ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

Дана робота обсягом 133 сторінки, включає 11 ілюстрацій, 2 таблиці, 13 формул, 1 додаток та містить посилання на 17 джерел літератури.

Об'єкт дослідження: цифрові дані мобільних пристроїв, що формуються в процесі функціонування операційної системи Android та встановлених застосунків.

Предмет дослідження: методи, алгоритми та засоби верифікації автентичності цифрових артефактів мобільних пристроїв з використанням індикаторного rule-based підходу.

Мета дослідження: розробити та експериментально обґрунтувати методику виявлення ознак несанкціонованого втручання у цифрові дані мобільних пристроїв на базі ОС Android, а також створити прототип інструменту для автоматизованої попередньої перевірки автентичності мобільних цифрових артефактів.

Методи дослідження: аналіз літературних джерел, порівняльний аналіз наявних forensic-інструментів, побудова тестових сценаріїв втручання в цифрові дані, методи структурного аналізу для виявлення логічних аномалій, програмно-моделювальні rule-based засоби на мові програмування Python.

Отримані результати: спроектовано індикаторну модель і rule-based скоринг. Реалізовано модульний Python-прототип, що нормалізує час до UTC, виконує перевірки, формує пояснюваний звіт і «карту розбіжностей». Експериментально підтверджено здатність системи виявляти типові маніпуляції з метаданими, часовими мітками та записами SQLite.

Результати роботи апробовано: тези опубліковано у науковому журналі «Наукоємні технології».

Ключові слова: цифрова криміналістика, автентичність даних, Android, EXIF, MediaStore, SQLite, резервні копії, rule-based індикатори, скоринг, кібербезпека.

ABSTRACT

The volume of the thesis is 133 pages, 11 figures, 2 tables, 13 formulas, 1 appendix and references 17 sources.

Research object: digital data of mobile devices generated by the Android operating system and installed applications.

The subject of the research: methods, algorithms, and tools for verifying the authenticity of mobile digital artifacts using an indicator-based rule-based approach.

The purpose of the research: to develop and experimentally substantiate a methodology for detecting signs of unauthorized tampering in mobile device data on Android, and to create a prototype tool for automated preliminary authenticity verification of mobile digital artifacts.

Research methods: analysis of scholarly sources, comparative analysis of existing forensic tools, construction of test scenarios for data tampering, structural analysis methods to detect logical anomalies, and rule-based software modeling in Python.

Obtained results: an indicator model and a rule-based scoring scheme were designed. A modular Python prototype was implemented that normalizes time to UTC, performs the checks, and produces an explainable report and a “discrepancy map.” Experiments confirmed the system’s ability to detect typical manipulations of metadata, timestamps, and SQLite records.

The results were validated through publication of theses in the scientific journal «Science-based technologies».

Keywords: digital forensics, data authenticity, Android, EXIF, MediaStore, SQLite, backups, rule-based indicators, scoring, cybersecurity.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ	7
ВСТУП	9
1 ТЕОРЕТИЧНІ ОСНОВИ ЗАБЕЗПЕЧЕННЯ АВТЕНТИЧНОСТІ ЦИФРОВИХ ДАНИХ МОБІЛЬНИХ ПРИСТРОЇВ.....	15
1.1 Поняття автентичності цифрових даних та її значення в інформаційній безпеці	16
1.2 Мобільні пристрої як джерело цифрових доказів	20
1.3 Основні загрози достовірності даних на мобільних пристроях	25
1.4 Огляд міжнародних стандартів і підходів до забезпечення автентичності цифрових даних	29
1.5 Методи та інструменти перевірки цілісності і автентичності цифрових артефактів. Аналіз сучасних програмних рішень для digital forensics	33
Висновки до розділу 1	37
2 АНАЛІЗ СТРУКТУРИ ДАНИХ ТА ДЖЕРЕЛ ЦИФРОВИХ АРТЕФАКТІВ МОБІЛЬНИХ ПРИСТРОЇВ	38
2.1 Архітектура операційної системи Android у контексті криміналістичного аналізу	39
2.2 Типи цифрових артефактів: EXIF-метадані, SQLite-бази, системні журнали, резервні копії.....	43
2.3 Потенційні сценарії підробки та модифікації цифрових даних	47
2.4 Проблематика співставлення часових міток та логічної послідовності подій	53
Висновки до розділу 2	55
3 РОЗРОБКА МОДЕЛІ ПЕРЕВІРКИ АВТЕНТИЧНОСТІ МОБІЛЬНИХ ДАНИХ	57
3.1 Загальна концепція запропонованого підходу	58
3.2 Вибір методів перевірки автентичності та оцінки достовірності	63
3.3 Формування системи індикаторів	66
3.4 Rule-based модель оцінювання	79
Висновки до розділу 3	82
4 ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОТОТИПУ МОВОЮ PYTHON	83
4.1 Модуль ingestion	84
4.2 Модуль extractors	87
4.3 Модуль indicators	90
4.4 Модуль scoring	95
4.5 Модуль reporting	97
Висновки до розділу 4	100
ВИСНОВКИ.....	102
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ	107
ДОДАТОК А. КОД МОВОЮ PYTHON ПРОГРАМНОГО ПРОТОТИПУ ОЦІНКИ АВТЕНТИЧНОСТІ МОБІЛЬНИХ ДАНИХ.....	110

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

EXIF — Exchangeable Image File Format (вбудовані метадані зображень)

JPEG/PNG/HEIF/MP4 — формати мультимедіа (зображення/відео)

GPS — Global Positioning System (координати в EXIF)

UTC — Coordinated Universal Time (універсальний координований час)

TZ — Time Zone (часовий пояс)

DST — Daylight Saving Time (перехід на літній/зимовий час)

Unix time (epoch) — кількість секунд/мс від 01.01.1970 UTC

MediaStore — системний каталог медіа Android (date_added, date_modified тощо)

Content URI — посилання виду content:// для доступу через провайдери контенту

FS — File System (файлова система)

EXT4/F2FS — поширені файлові системи Android

mtime/ctime/atime — час модифікації/зміни inode/останнього доступу у ФС

SQLite — вбудована СУБД мобільних застосунків

WAL/SHM — журнали SQLite Write-Ahead Logging / Shared Memory

VACUUM — операція перезапаковування/«ущільнення» SQLite

JSON — JavaScript Object Notation (формат обміну даними)

CSV — Comma-Separated Values (табличний текстовий формат)

Hash — криптографічний хеш (контрольна сума об'єкта)

MD5/SHA-1/SHA-256 — алгоритми хешування

pHash/aHash/dHash — перцептивні хеші для порівняння зображень

Thumbnail — мініатюра/ескіз зображення

Backup — резервна копія (архів даних/маніфест)

Manifest — описовий файл резервної копії (списки, хеші, метадані)

Indicator — індикатор (правило/ознака потенційної підміни)

Scoring — скоринг, інтегральна оцінка достовірності

Rule-based — підхід на основі формальних правил

Timeline — хронологія подій

Timeline conflict — неможлива послідовність подій у нормалізованому часі

Chain of Custody (CoC) — безперервний ланцюг володіння доказами

ADB — Android Debug Bridge (інструмент доступу/вилучення)

APK — Android Package (пакунок застосунку)

UUID — Universally Unique Identifier (універсальний унікальний ідентифікатор)

IMEI — International Mobile Equipment Identity (ідентифікатор пристрою)

ICCID — Integrated Circuit Card Identifier (ідентифікатор SIM)

MIME type — тип вмісту (наприклад image/jpeg, video/mp4)

DPI — Dots Per Inch (щільність пікселів екрана/зображення)

ROI — Region of Interest (ділянка інтересу в зображенні)

OCR — Optical Character Recognition (розпізнавання тексту)

SDK — Software Development Kit (набір для розробника)

ВСТУП

Актуальність дослідження:

У XXI столітті мобільні пристрої стали невід'ємною частиною життєдіяльності людини, поєднуючи функції персонального комп'ютера, комунікаційного засобу, електронного гаманця, камери та засобу ідентифікації особи. За статистикою, понад 80% цифрової інформації, що генерується користувачами, так чи інакше створюється або обробляється через мобільні пристрої. Це робить смартфони ключовими вузлами інформаційного простору, а дані, які на них зберігаються, — критично важливими у контексті забезпечення кібербезпеки, цифрової ідентичності та судово-технічних розслідувань.

Мобільні операційні системи, насамперед Android, створюють багаторівневе середовище збереження інформації, що включає системні журнали, бази даних SQLite, EXIF-метадані мультимедіа, файли кешу, резервні копії та інші цифрові артефакти. У сукупності ці елементи формують складну екосистему цифрових слідів користувача, які можуть бути використані для відтворення хронології подій, підтвердження місця, часу, авторства або самого факту взаємодії з пристроєм. Водночас така структура даних є вразливою до навмисних модифікацій — зокрема шляхом редагування метаданих, зміни часових міток, підміни або часткового видалення записів у базах даних.

З розвитком технологій цифрової обробки зображень та відкриттям доступу до інструментів низького рівня (root-доступ, ADB, forensic API) зловмисники отримали можливість тонко змінювати інформаційні атрибути файлів без помітного втручання в їхній зміст. Такі зміни часто залишаються непоміченими при поверхневому аналізі або під час використання

стандартних forensic-інструментів. Виявлення навіть незначних ознак фальсифікації стає надзвичайно складним завданням, що потребує поєднання криптографічних, структурних і поведінкових методів аналізу.

Проблема автентичності цифрових даних має особливе значення у правовій та експертній практиці. Згідно з положеннями міжнародних і національних стандартів (ISO/IEC 27037:2012, NIST SP 800-101 Rev.1, ENFSI Best Practice Manual for Mobile Phone Forensics), будь-який цифровий артефакт може розглядатися як доказ лише за умови доведення його достовірності та цілісності. У судовій експертизі це означає, що експерт повинен підтвердити, що дані не зазнали змін з моменту їх створення або вилучення. У практиці кібербезпеки це завдання проявляється у необхідності виявлення спроб маніпуляції цифровими журналами, підроблення подій, зміни журналів автентифікації чи GPS-координат.

Класичні методи цифрової криміналістики — такі як обчислення контрольних хешів (MD5, SHA-256) або порівняння копій — не завжди дозволяють виявити складні форми фальсифікації. Наприклад, зміна EXIF-метаданих у JPEG-файлі без порушення його структури не призводить до зміни контрольної суми; аналогічно, редагування SQLite-бази з подальшою вакуумізацією таблиць може залишити мінімум слідів для звичайного хеш-контролю. Це створює потребу у комплексному підході до верифікації, який поєднує багат шарову перевірку різних характеристик файлів і баз даних.

З наукової точки зору, проблема автентичності цифрових даних є багатокомпонентною. Вона охоплює питання математичного моделювання поведінки цифрових артефактів, розробки алгоритмів кореляційного аналізу часових послідовностей, побудови систем оцінювання достовірності на основі ознак, а також створення інструментів автоматизованого аналізу, що можуть підтримувати роботу експерта.

Така інтеграція підходів — від класичних методів контролю цілісності до rule-based або machine learning систем — відкриває нові можливості для підвищення точності судово-технічного аналізу.

Крім того, з позиції інформаційного права та політики кібербезпеки, підтвердження автентичності мобільних даних є однією з умов реалізації принципів доказовості та довіри до електронних документів, визначених Законом України «Про електронні довірчі послуги» та «Про захист інформації в інформаційно-телекомунікаційних системах». У контексті зростання кількості кіберінцидентів, пов'язаних із витоком або піддробкою персональних даних, це питання має не лише технічне, а й суспільне та правове значення.

Актуальність теми також підтверджується практичними потребами слідчих, судово-експертних та корпоративних структур, які стикаються з необхідністю швидкої, але достовірної оцінки цифрових матеріалів. Існує дефіцит інструментів, здатних автоматично оцінювати достовірність даних без необхідності ручного втручання спеціаліста. Саме тому створення прототипу програмного засобу, який виконує первинну rule-based перевірку цифрових артефактів, має вагоме наукове та прикладне значення.

Таким чином, дослідження автентичності даних мобільних пристроїв є актуальним як у теоретичному, так і у прикладному аспектах. Воно спрямоване на вирішення однієї з ключових проблем сучасної кібербезпеки — гарантування довіри до цифрових доказів та захисту інформаційної правди у цифровому середовищі. Розробка алгоритмів, методик і програмних рішень для підтвердження автентичності даних створює основу для підвищення ефективності криміналістичних розслідувань, судово-технічної експертизи та загального рівня інформаційної безпеки держави.

Мета дослідження:

Розробити та експериментально обґрунтувати методику виявлення ознак несанкціонованого втручання у цифрові дані мобільних пристроїв на базі ОС Android, а також створити прототип інструменту для автоматизованої попередньої перевірки автентичності мобільних цифрових артефактів.

Завдання дослідження:

1. Провести аналіз сучасних загроз достовірності мобільних даних та методів їх криміналістичної перевірки.
2. Дослідити структуру цифрових артефактів мобільних пристроїв (EXIF, timestamps, SQLite, thumbnails, backups).
3. Розробити методику rule-based перевірки автентичності мобільних даних із використанням системи індикаторів.
4. Реалізувати Python-прототип інструменту для автоматизованого виявлення потенційних ознак модифікації.
5. Провести тестування розробленого інструменту на контрольних наборах даних і проаналізувати результати ефективності.
6. Сформулювати практичні рекомендації щодо застосування запропонованої методики у цифровій криміналістиці.

Об'єкт дослідження:

Цифрові дані мобільних пристроїв, що формуються в процесі функціонування операційної системи Android та встановлених застосунків.

Предмет дослідження:

Методи, алгоритми та засоби верифікації автентичності цифрових артефактів мобільних пристроїв з використанням індикаторного rule-based підходу.

Методи дослідження:

Аналіз літературних джерел, порівняльний аналіз наявних forensic-інструментів, побудова тестових сценаріїв втручання в цифрові дані, методи структурного аналізу для виявлення логічних аномалій, програмно-моделювальні rule-based засоби на мові програмування Python.

Новизна одержаних результатів:

Наукова новизна одержаних результатів полягає у розробленні комплексного підходу до підтвердження автентичності мобільних даних, який ґрунтується на багатофакторній індикаторній оцінці.

Уперше запропоновано rule-based модель інтеграції кількох цифрових індикаторів (EXIF_FS_INCONSISTENT, SQL_ROW_DIFF, HASH_MISMATCH, THUMBNAIL_MISMATCH тощо), сформовано алгоритм побудови агрегованого ризикового показника (score) для визначення достовірності артефактів та створено Python-прототип системи автоматизованої перевірки автентичності мобільних даних із можливістю формування технічного звіту для попередньої судово-технічної експертизи.

Практичне значення отриманих результатів:

Результати дослідження мають практичне значення для підрозділів кіберполіції, експертних лабораторій, інформаційно-безпекових підрозділів підприємств і розробників forensic-програмного забезпечення.

Розроблений Python-прототип може бути використаний як допоміжний інструмент для попереднього аналізу цифрових доказів, що дозволяє скоротити час первинної експертизи, визначити найбільш підозрілі файли та виявити нетипові відхилення у цифрових параметрах.

Методика є універсальною, розширюваною та може бути адаптована для інших платформ, зокрема iOS.

Апробація результатів роботи:

Основні результати роботи було апробовано у науковій публікації у журналі «Наукоємні технології» (публікація у випуску Том 69 №1 2026 р.), де представлено концепцію rule-based оцінювання автентичності цифрових артефактів та результати експериментальних досліджень на контрольних даних Android-пристроїв.

1 ТЕОРЕТИЧНІ ОСНОВИ ЗАБЕЗПЕЧЕННЯ АВТЕНТИЧНОСТІ ЦИФРОВИХ ДАНИХ МОБІЛЬНИХ ПРИСТРОЇВ

Забезпечення достовірності та автентичності цифрових даних є одним із центральних завдань сучасної кібербезпеки та цифрової криміналістики.

В умовах, коли значна частина інформаційного обігу відбувається через мобільні пристрої, саме вони стають критично важливими джерелами доказів у розслідуваннях кіберінцидентів, правопорушень, корпоративних конфліктів чи судових спорів. Проте висока мобільність, динамічність даних і численні можливості їх редагування створюють передумови для маніпуляцій, що ускладнюють підтвердження їхньої справжності.

Мобільні телефони, планшети та інші портативні пристрої зберігають величезні обсяги цифрових артефактів — від журналів дзвінків і текстових повідомлень до геолокаційних даних, мультимедійних файлів і резервних копій. Ці артефакти формуються автоматично під час експлуатації пристрою, проте їхня структура та ступінь захищеності залежать від особливостей системної архітектури та політики безпеки конкретної ОС.

Будь-яке втручання у ці дані, зокрема редагування EXIF-метаданих, зміну часових міток або маніпуляції з базами SQLite, може призвести до спотворення інформаційної картини подій і, відповідно, помилкових висновків під час технічного аналізу.

Саме тому в сучасній цифровій криміналістиці особливого значення набуває верифікація автентичності даних мобільних пристроїв — тобто перевірка їх походження, цілісності, послідовності створення та узгодженості між різними джерелами.

Встановлення автентичності є обов'язковою передумовою прийняття цифрових матеріалів як допустимих доказів у судовому процесі, а також

базовим елементом довіри до будь-якого результату цифрового розслідування.

Дослідження теоретичних основ є необхідним етапом для подальшого формування власної методики оцінки достовірності мобільних даних, розробки алгоритму rule-based перевірки та створення Python-прототипу системи попередньої автоматизованої верифікації.

Таким чином, цей розділ визначає науково-методологічну базу для реалізації практичної частини магістерської роботи та окреслює ключові положення, на яких ґрунтується подальше дослідження.

1.1 Поняття автентичності цифрових даних та її значення в інформаційній безпеці

Автентичність цифрових даних є одним із базових понять у сфері інформаційної безпеки, цифрової криміналістики та судово-технічної експертизи.

Під терміном «автентичність» у контексті електронної інформації розуміють підтвердження того, що дані є справжніми, походять від зазначеного джерела, не були змінені або підроблені, а також відображають реальні події, обставини чи дії, до яких вони належать. Іншими словами, автентичність — це властивість інформації, що забезпечує довіру до її походження, цілісності та змісту.

У теоретичному розумінні автентичність є тісно пов'язаною з поняттями цілісності (integrity) та невідмовності (non-repudiation). Якщо цілісність забезпечує відсутність змін у даних від моменту створення до моменту перевірки, то автентичність розширює це визначення, включаючи підтвердження джерела створення, контексту та коректності змісту.

Саме тому автентичність вважається складовою загального трикутника інформаційної безпеки — «конфіденційність, цілісність, доступність», але у криміналістичному аспекті вона виступає самостійною властивістю доказового значення даних.

У науковій літературі виділяють кілька рівнів автентичності:

1. Технічна автентичність — підтвердження того, що файл або запис не був змінений на фізичному чи логічному рівні (наприклад, перевірка хеш-сум).
2. Логічна автентичність — підтвердження внутрішньої узгодженості інформаційних атрибутів, наприклад, коректності часових міток, зв'язку між подіями, метаданими та логами.
3. Контекстна автентичність — оцінка відповідності даних реальним обставинам, у яких вони були створені, з урахуванням поведінкових або середовищних факторів.

Для мобільних пристроїв ці рівні мають особливе значення, оскільки цифрові артефакти часто створюються автоматично (наприклад, EXIF-метадані фотографій, журнали дзвінків або координати GPS), а користувач може не усвідомлювати факт їхнього формування.

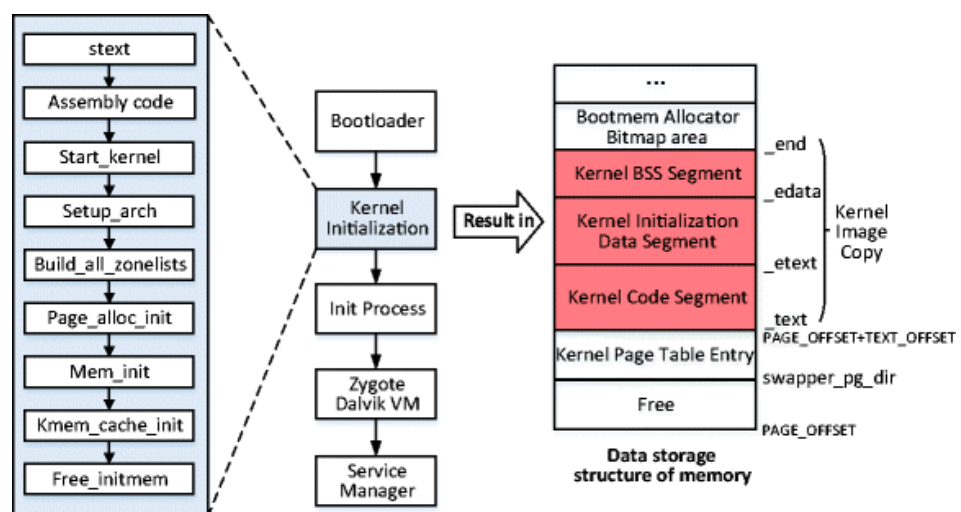


Рисунок 1.1 — Процедура завантаження Android та структура зберігання даних у пам'яті після ініціалізації ядра

Це створює як перевагу (висока деталізація цифрового сліду), так і вразливість — можливість непомітного редагування або часткової підміни інформації.

У системі забезпечення інформаційної безпеки автентичність виконує кілька ключових функцій:

1. Гарантування достовірності — підтвердження того, що цифрові дані не були сфальсифіковані або змінені;
2. Підтримання доказової сили — забезпечення можливості використання цифрових артефактів у судових і дисциплінарних розслідуваннях;
3. Формування довіри до інформаційних систем — забезпечення прозорості та передбачуваності цифрових процесів;
4. Захист від репутаційних та фінансових ризиків — попередження маніпуляцій з цифровими записами, що можуть призвести до неправдивих рішень або втрат.

Міжнародні стандарти, зокрема ISO/IEC 27037:2012 та NIST SP 800-101 Rev.1, визначають автентичність як одну з базових вимог до цифрових доказів.

Відповідно до цих документів, будь-який об'єкт цифрового середовища може бути визнаний доказовим лише за умови підтвердження його походження та незмінності.

З цієї точки зору, автентичність є невід'ємною частиною процесів Chain of Custody — безперервного ланцюга фіксації, передачі, зберігання та аналізу цифрових доказів. Порушення автентичності хоча б на одному з етапів цього ланцюга робить доказ недопустимим у судовій практиці.

В контексті мобільної криміналістики забезпечення автентичності ускладнюється динамічною природою мобільних систем: постійною зміною стану пам'яті, автоматичним оновленням даних, синхронізацією з хмарними

сховищами, а також взаємодією численних застосунків із різними рівнями доступу до файлової системи.

Для прикладу, просте оновлення месенджера може змінити структуру бази SQLite, а автоматичне створення резервної копії — змінити часові мітки у файловій системі. Такі зміни можуть бути легітимними, але без відповідної фіксації вони сприймаються як можливі ознаки втручання.

Важливим аспектом автентичності є також можливість її перевірки — наявність технічних та процедурних механізмів, що дозволяють незалежно відтворити процес перевірки.

У цьому сенсі принцип reproducibility (відтворюваності) є одним із наріжних каменів цифрової криміналістики.

Якщо дослідник або експерт не може повторити перевірку з тими самими результатами, автентичність вважається непідтвердженою.

Для досягнення цього застосовуються різноманітні методи — від традиційного хешування файлів (MD5, SHA-1, SHA-256) до кореляційного аналізу метаданих і виявлення аномалій у часових послідовностях.

Однак навіть найточніші математичні методи не гарантують абсолютної достовірності без урахування контексту формування даних. Тому сучасні дослідження спрямовані на поєднання технічних і логічних підходів — перевірку узгодженості даних у різних рівнях системи: файловому, застосунковому, хмарному та поведінковому.

З практичної точки зору, підтвердження автентичності є не лише технічним, але й організаційним завданням. У сфері кібербезпеки воно тісно пов'язане з управлінням інцидентами, аудитом безпеки, виявленням вторгнень і реагуванням на них. У цифровій криміналістиці воно виступає основою для побудови доказової бази. А в системах електронного документообігу — гарантією довіри до електронних підписів, журналів аудиту та результатів обробки інформації.

Таким чином, поняття автентичності цифрових даних є багаторівневим і комплексним. Воно поєднує технічні, правові та процедурні складові, утворюючи фундаментальну основу для всієї системи забезпечення інформаційної безпеки.

Забезпечення автентичності даних мобільних пристроїв — це не лише питання збереження технічної цілісності, а й підтвердження достовірності цифрового світу, на якому базується сучасне суспільство.

1.2 Мобільні пристрої як джерело цифрових доказів

У сучасній цифровій екосистемі мобільні пристрої посідають особливе місце як найбільш інформативні та водночас найвразливіші джерела цифрових доказів.

Смартфони та планшети стали центральним елементом персонального цифрового середовища, через який користувач взаємодіє з банківськими сервісами, соціальними мережами, корпоративними системами, службами геолокації та електронними засобами комунікації. Згідно з дослідженнями ENFSI Mobile Forensics Guidelines (2020), до 80% інформації, яка використовується під час криміналістичних розслідувань, походить саме з мобільних пристроїв.

З погляду цифрової криміналістики мобільний пристрій є комплексним об'єктом збору даних, що поєднує різнорівневі джерела інформації — від фізичних секторів пам'яті до логічних структур баз даних і метаданих застосунків.

На відміну від класичних комп'ютерних систем, архітектура мобільних ОС передбачає більш складну і динамічну модель зберігання даних, у якій важливу роль відіграють системні журнали, кеші, файли конфігурацій, метадані зображень, SQLite-бази, історії повідомлень і резервні копії. Усі ці

артефакти можуть бути використані як цифрові докази, за умови забезпечення їх автентичності та збереження ланцюга передачі (chain of custody).

Основні категорії цифрових доказів у мобільних пристроях:

1. Фотографії, відео та аудіозаписи є одними з найінформативніших видів даних. Вони зберігають метадані у форматі EXIF (Exchangeable Image File Format), що містять дату й час створення, модель пристрою, географічні координати, параметри зйомки та навіть версію прошивки камери. Ці параметри дозволяють встановити не лише автентичність самого зображення, а й обставини його створення, проте редагування EXIF-даних — одна з найпоширеніших форм фальсифікації цифрових матеріалів.
2. Практично всі сучасні мобільні застосунки (контакти, повідомлення, месенджери, браузер, карти, банківські програми) використовують SQLite як легку вбудовану систему керування базами даних. В Android системні бази contacts2.db, calllog.db, mmssms.db, browser.db, media.db тощо містять історію викликів, SMS-повідомлень, списки контактів, веб-журнали та мультимедіа. Вилучення і дослідження цих баз є важливим етапом мобільного forensic-аналізу, оскільки вони зберігають детальний слід користувацької активності.
3. У системі Android основні журнали (logcat, events log, kernel log, dmesg) фіксують дії користувача, встановлення або видалення додатків, помилки, мережеві з'єднання, доступ до файлової системи, а також події ядра. Вони часто використовуються для підтвердження послідовності подій та співставлення часових міток із іншими джерелами інформації.
4. До важливих артефактів належать історії переписки у таких застосунках, як WhatsApp, Telegram, Viber, Signal, Messenger, що зберігаються у вигляді локальних або зашифрованих SQLite-баз. Ці

журнали містять не лише текст повідомлень, але й інформацію про відправників, одержувачів, статус доставки, вкладення та часові мітки. Вони мають особливе значення під час розслідувань кіберінцидентів, шахрайства чи незаконного обігу інформації.

5. Логи GPS, історія Wi-Fi-з'єднань, Bluetooth-журнали та бази Location History дозволяють встановити місцезнаходження користувача в певний момент часу. Ці дані мають високу доказову силу, але водночас є об'єктом частих маніпуляцій — наприклад, шляхом штучного зміщення координат або видалення записів.
6. Резервні копії (backup) створюються системою Android або сторонніми додатками і можуть зберігатися локально чи в хмарних сервісах (Google Drive, Dropbox тощо). Вони містять метадані про час створення, обсяг, структуру та контрольні суми файлів. Виявлення розбіжностей між версіями резервних копій дозволяє виявити спроби підміни чи редагування даних у минулому.
7. Файли типу settings.db, accounts.db, packages.xml, events.db містять інформацію про конфігурацію системи, встановлені програми, параметри користувача та події встановлення або оновлення ПЗ. Аналіз цих даних дає змогу відтворити логіку дій користувача або програмного забезпечення в момент інциденту.

Особливість мобільних пристроїв полягає у їхній динамічній природі даних.

На відміну від традиційних комп'ютерів, де дані можуть залишатися незмінними протягом тривалого часу, у смартфонах більшість записів оновлюється в режимі реального часу: системні логи мають обмежений розмір буфера, бази SQLite регулярно переписуються, а резервні копії синхронізуються з хмарними сховищами.

Через це отримати повний і незмінений образ системи вкрай складно без фізичного доступу до пристрою або використання спеціалізованих forensic-інструментів.

Іншою важливою особливістю є змішаний рівень доступу до даних. Android реалізує багаторівневу систему захисту — sandboxing, контроль дозволів, шифрування FBE (File-Based Encryption) — що ускладнює пряме вилучення даних.

Для глибокого криміналістичного аналізу часто потрібно отримати root-доступ або фізичний дамپ пам'яті, що може суперечити політиці безпеки або вимогам ланцюга збереження доказів.

Крім того, мобільні пристрої постійно взаємодіють з мережевими сервісами — соціальними мережами, поштовими клієнтами, системами хмарного збереження.

Це призводить до розподілення цифрових слідів між локальною пам'яттю та віддаленими серверами, що створює додаткові складнощі при перевірці автентичності. Наприклад, у деяких випадках локальний запис може бути змінений користувачем, тоді як хмарна версія зберігає початковий стан — або навпаки.

Ще однією характерною рисою є наявність прихованих артефактів. Навіть після видалення файлів з інтерфейсу користувача, їхні фрагменти можуть залишатися в кешах, журналах чи тінювих копіях системи. Ці залишкові дані часто є безцінними для підтвердження фактів втручання або відновлення справжнього змісту інформації.

Мобільні пристрої сьогодні виконують роль інтегрованих цифрових свідків людської діяльності. Вони фіксують не лише контент, але й поведінкові аспекти користувача — послідовність дій, місце, час, частоту звернень до певних сервісів. У криміналістичній практиці ці дані дозволяють:

1. підтвердити факт перебування особи у певній геолокації;
2. встановити контакти між суб'єктами;
3. відтворити хронологію подій;
4. підтвердити чи спростувати версію події;
5. виявити спроби приховування чи модифікації інформації.

Таблиця 1.1 — Основні типи доказів у мобільних пристроях

Категорія	Приклади файлів/баз	Інформативність	Ризик фальсифікації
Мультимедіа	EXIF JPEG, MP4, 3GP	Висока	Середній
SQLite-бази	contacts.db, sms.db	Висока	Високий
Журнали подій	logcat, events.db	Середня	Низький
Геолокація	GPS, Wi-Fi logs	Висока	Середній

У сфері кібербезпеки аналіз мобільних артефактів дозволяє виявляти порушення політик безпеки, компрометацію пристроїв, витоки конфіденційної інформації та ознаки внутрішніх інцидентів.

З цієї причини формування методів, що забезпечують перевірку автентичності цифрових даних мобільних пристроїв, має вирішальне значення для забезпечення довіри до результатів технічних розслідувань та інформаційної цілісності.

1.3 Основні загрози достовірності даних на мобільних пристроях

Забезпечення достовірності цифрових даних на мобільних пристроях є складним завданням, оскільки інформація в них постійно змінюється, синхронізується та перезаписується.

Дані можуть бути модифіковані як унаслідок легітимних процесів, так і в результаті цілеспрямованого втручання користувача або сторонніх осіб.

Це створює численні ризики для автентичності цифрових доказів, особливо коли йдеться про матеріали, що мають доказове значення у судово-технічних або інформаційно-безпекових розслідуваннях.

Одним із найпоширеніших напрямів фальсифікації є зміна метаданих мультимедійних файлів, зокрема EXIF-параметрів у фотографіях і відеозаписах.

У метаданих можуть зберігатися такі критично важливі атрибути, як дата і час створення, модель камери, географічні координати, а також параметри експозиції.

Використовуючи спеціальні утиліти на кшталт ExifTool або Metadata++, зломисник може змінити ці параметри без порушення структури файлу, що унеможлиблює виявлення підробки звичайними методами перевірки контрольних сум.

Такі маніпуляції здатні спотворити справжню хронологію подій, створити хибне враження щодо місця зйомки або часу створення зображення, що особливо небезпечно під час криміналістичних експертиз.

У деяких випадках редагування зображення супроводжується невідповідністю між його вмістом і кешованою мініатюрою, яка зберігає початковий вигляд файлу. Порівняння таких мініатюр з оригіналом може свідчити про факт редагування або підробки цифрового матеріалу, навіть якщо саме зображення виглядає цілком автентичним.

Іншою серйозною загрозою є фальсифікація часових атрибутів і системних журналів подій.

У системі Android кожен файл має декілька часових міток — створення (ctime), модифікації (mtime) та доступу (atime). Їх зміна можлива як у процесі звичайної експлуатації, так і шляхом навмисного редагування через root-доступ або спеціальні файлові менеджери.

Змінивши системний час, користувач може створити файли, що виглядають старішими або новішими за реальні, чим формує хибну послідовність дій.

У логах подій також можливе підроблення міток часу, що веде до некоректної реконструкції хронології.

Такі спотворення особливо небезпечні в інформаційній безпеці, адже навіть мінімальне відхилення у часі може змінити трактування причинно-наслідкових зв'язків між подіями.

Надзвичайно вразливим об'єктом маніпуляцій є бази даних SQLite, які використовуються практично всіма системними компонентами та застосунками Android.

Ці бази зберігають журнали викликів, повідомлень, історію браузера, медіатеки та інші цифрові артефакти. Вони легко редагуються за допомогою відкритих утиліт, при цьому структура бази залишається валідною, а зміни можуть бути непомітними для поверхневого аналізу.

До типових способів фальсифікації належать редагування або видалення записів, вставка нових даних, що симулюють певні дії користувача, зміна часових полів або стирання слідів редагування за допомогою vacuum-операцій.

Виявити таке втручання складно навіть за наявності хеш-маніфестів, адже модифікація однієї таблиці не обов'язково змінює загальний хеш усієї бази.

Часто відновлення оригінального стану можливе лише через порівняння резервних копій або аналіз вільного простору бази, де можуть залишатися фрагменти видалених записів.

Не менш небезпечним є втручання у системи резервного копіювання та хмарної синхронізації. Android створює резервні копії як локально, так і у хмарних сховищах, а користувач або стороння особа може замінити ці файли на інші, попередньо відредаговані. Це дозволяє сформувати альтернативну версію цифрової історії пристрою — наприклад, створити резервну копію без компрометуючих записів або повідомлень.

Редагування резервних копій можливе через підміну архівів, зміну контрольних сум у маніфестах або навіть підміну дати створення файлу. Такі зміни складно виявити без глибокого технічного аналізу, оскільки система сприймає підроблені резервні копії як автентичні.

Серед інших форм маніпуляцій варто відзначити втручання у кеші мініатюр, де можуть зберігатися попередні версії редагованих файлів, зміну системних конфігураційних баз і логів через root-доступ, інжекцію шкідливого коду, який модифікує журнали подій, а також видалення тимчасових файлів, що містять початкові сліди активності користувача.

У деяких випадках злочинці застосовують фізичне перезаписування секторів пам'яті для повного знищення залишкових артефактів, унаслідок чого навіть спеціалізовані forensic-інструменти не можуть відновити попередній стан даних.

Проблема достовірності мобільних даних ускладнюється тим, що фальсифікації можуть бути багаторівневими і здійснюватися одночасно на кількох рівнях системи.

Наприклад, користувач може змінити часову мітку файлу, відредагувати EXIF-дані фотографії, а потім створити нову резервну копію, яка міститиме вже підроблену інформацію.

Така комбінація маніпуляцій робить традиційні методи перевірки — хешування, сигнатурний аналіз або побітове порівняння — малоефективними, оскільки вони не враховують логічні взаємозв'язки між різними джерелами інформації.

У таких випадках потрібен багаторівневий підхід, що оцінює сукупність ознак: часових, структурних, поведінкових та контекстуальних.

Таким чином, основна загроза достовірності мобільних даних полягає не лише у можливості їхньої технічної модифікації, але й у складності виявлення таких змін.

Система Android, попри наявність механізмів цілісності, не забезпечує абсолютного захисту від цілеспрямованих редагувань. Це обумовлює потребу у створенні спеціалізованих методик та програмних рішень, що дозволяють проводити rule-based або комплексну перевірку автентичності.

Такі підходи передбачають аналіз не лише окремих атрибутів файлів, а й їх взаємозв'язків, узгодженості часових послідовностей, відповідності між мініатюрами, базами даних і резервними копіями.

Лише комплексне поєднання цих параметрів може забезпечити об'єктивне підтвердження або спростування автентичності цифрових матеріалів.

Саме усвідомлення масштабності цих загроз і обумовлює необхідність розробки нових інструментів, здатних автоматизувати процес оцінки достовірності даних, що й стало основою даного дослідження.

1.4 Огляд міжнародних стандартів і підходів до забезпечення автентичності цифрових даних

Першим і найбільш визнаним документом у цій сфері є ISO/IEC 27037:2012 — Guidelines for Identification, Collection, Acquisition and Preservation of Digital Evidence.

Цей стандарт визначає ключові принципи роботи з цифровими доказами та наголошує, що будь-який об'єкт, який може мати доказове значення, повинен бути вилучений і збережений у такий спосіб, щоб гарантувати його автентичність і цілісність.

Основна ідея документа полягає у забезпеченні так званого chain of custody — безперервного ланцюга зберігання доказів, який фіксує всі дії з даними від моменту їх виявлення до подання у суді.

У стандарті чітко вказано, що будь-яке втручання у структуру чи атрибути файлу без відповідної фіксації є порушенням автентичності, навіть якщо зміни не торкаються змісту інформації.

Важливим доповненням до ISO/IEC 27037 є стандарт ISO/IEC 27041:2015 — Guidance on Assuring Suitability and Adequacy of Investigation Methods.

Він встановлює критерії оцінки придатності методів цифрового аналізу, тобто перевіряє, чи може певна методика бути визнана надійною з точки зору доказового процесу.

Саме цей документ наголошує на необхідності відтворюваності (reproducibility) та прозорості процедур, що безпосередньо пов'язано з поняттям автентичності даних.

Додатково, ISO/IEC 27042:2015 — Guidelines for the Analysis and Interpretation of Digital Evidence конкретизує етапи аналітичної обробки даних і визначає вимоги до збереження доказового значення під час аналізу.

У ньому зазначено, що при перевірці цифрових доказів важливо зберігати не лише самі файли, але й метадані, тимчасові мітки, журнали дій та контекст формування даних. Таким чином, автентичність розглядається не лише як незмінність змісту, а як цілісність усієї інформаційної екосистеми, у якій ці дані існували.

У сфері мобільної криміналістики ключову роль відіграє документ NIST Special Publication 800-101 Revision 1 — Guidelines on Mobile Device Forensics, розроблений Національним інститутом стандартів і технологій США.

Цей стандарт регламентує процедури вилучення та аналізу даних із мобільних пристроїв, включно з Android, iOS та іншими платформами. У ньому визначено етапи збору доказів, методи збереження даних, особливості роботи з флеш-пам'яттю, SIM-картами, резервними копіями та хмарними синхронізаціями.

Особливу увагу приділено збереженню автентичності артефактів під час копіювання — наприклад, вимозі щодо використання write-blocker-пристроїв, верифікації хешів після кожної операції копіювання та фіксації логів усіх дій експерта.

У контексті даної роботи цей стандарт важливий тим, що підкреслює необхідність багаторівневої перевірки — технічної, логічної та контекстної.

Європейська мережа судово-криміналістичних інститутів (ENFSI) у своїх “Best Practice Manuals for Mobile Phone Forensics” (2020) розробила низку практичних рекомендацій, спрямованих на стандартизацію підходів до аналізу мобільних пристроїв.

Вони охоплюють усі стадії дослідження — від вилучення пристрою до формування звіту. ENFSI наголошує, що автентичність цифрових доказів залежить не лише від технічної процедури, але й від організаційного

контролю: правильного документування, дотримання процедур зберігання, кваліфікації експерта та незалежної перевірки результатів.

У документі також наведено вимогу до використання щонайменше двох незалежних методів перевірки під час підтвердження достовірності цифрового артефакту, що узгоджується з концепцією комплексного rule-based аналізу.

Окремої уваги заслуговує принцип Chain of Custody, який розглядається у всіх міжнародних стандартах як базовий елемент забезпечення автентичності.

Він означає, що кожна дія з цифровим об'єктом — копіювання, передача, збереження, аналіз, створення резервної копії — має бути задокументована із зазначенням часу, місця, особи, яка здійснювала дію, та інструментів, що використовувалися.

Лише за таких умов можна довести, що цифрові матеріали не зазнали несанкціонованих змін у процесі дослідження. У практиці судово-технічної експертизи це є обов'язковою вимогою, оскільки навіть мінімальне порушення ланцюга довіри призводить до втрати доказового статусу матеріалів.

Крім стандартів, існують також міжнародні методичні підходи, які не мають статусу обов'язкових, але широко застосовуються у науковій та експертній практиці. Зокрема, це ACPO Guidelines for Digital Evidence (Association of Chief Police Officers, Великобританія), які містять чотири базові принципи роботи з цифровими доказами.

Згідно з ними, жодна дія не повинна змінювати дані, будь-які маніпуляції мають бути повністю документовані, а аналітичні процедури повинні бути повторюваними та незалежно перевіреними. Ці принципи фактично заклали основу сучасної філософії цифрової криміналістики, орієнтованої на відтворюваність та достовірність.

Таким чином, аналіз міжнародних стандартів свідчить, що автентичність цифрових даних розглядається як комплексне поняття, що охоплює технічну, організаційну, правову та процедурну складові.

Незважаючи на існування численних методичних документів, більшість із них орієнтовані на класичні комп'ютерні системи та не повною мірою враховують специфіку мобільних пристроїв, зокрема динамічну структуру даних, багаторівневу систему зберігання, залежність від хмарних сервісів і синхронізацій. Саме це створює передумови для подальших досліджень, спрямованих на адаптацію принципів цифрової автентифікації до мобільного середовища.

У результаті огляду нормативних джерел можна зробити висновок, що міжнародна спільнота визнає перевірку автентичності цифрових даних ключовою умовою достовірності цифрових доказів.

Проте сучасні стандарти здебільшого не передбачають автоматизованих інструментів для оцінки автентичності у складних середовищах, таких як Android. Це визначає наукову нішу даного дослідження — розробку програмного рішення, здатного виявляти ознаки несанкціонованих змін у цифрових артефактах на основі багатофакторного аналізу індикаторів.

1.5 Методи та інструменти перевірки цілісності і автентичності цифрових артефактів. Аналіз сучасних програмних рішень для digital forensics

Найпоширенішими технічними методами перевірки цілісності є криптографічне хешування, цифрове підписування та порівняльний аналіз контрольних сум. Хешування передбачає обчислення унікального ідентифікатора файлу за допомогою алгоритмів MD5, SHA-1, SHA-256 або SHA-3. Якщо навіть один байт даних змінюється, хеш-код набуває іншого значення, що дозволяє виявити факт редагування.

Хоча цей метод широко застосовується, він не завжди гарантує виявлення логічних змін — наприклад, якщо користувач змінює EXIF-метадані, а потім обчислює новий хеш, система сприйматиме файл як “новий” і не зможе порівняти його з оригіналом без наявності базового маніфесту.

Цифровий підпис (digital signature) доповнює хешування механізмом криптографічного підтвердження авторства та незмінності даних за допомогою асиметричних ключів. Він забезпечує не лише контроль цілісності, а й автентичність джерела, однак використовується переважно для офіційних документів, електронних транзакцій або захищених каналів зв’язку, а не для аналізу мобільних артефактів.

Інший підхід — порівняння образів (image comparison), тобто побітове співставлення двох копій пам’яті чи файлів. Цей метод ефективний для виявлення змін між контрольним і досліджуваним екземпляром, але потребує наявності оригіналу, що не завжди можливо в реальних розслідуваннях. Крім того, він не дає змоги оцінити логічні взаємозв’язки між різними об’єктами, наприклад, між EXIF, SQLite і журналами системи.

Для підвищення надійності перевірки автентичності застосовується також кореляційний і сигнатурний аналіз, коли дані зіставляються між

різними джерелами. Наприклад, дата створення фотографії перевіряється відносно часової мітки в системному журналі, GPS-координати — відносно мережевих логів, а записи у базах SQLite — відносно історії резервного копіювання.

Такий підхід дозволяє виявити неузгодженість у контексті, навіть якщо окремі елементи виглядають технічно автентичними. У сучасній криміналістиці активно розвивається напрям rule-based аналізу, який полягає у створенні набору правил (індикаторів), за якими автоматично визначається наявність або відсутність ознак втручання.

Застосування цих методів реалізується у низці професійних програмних рішень для цифрової криміналістики. Найвідомішим серед них є Cellebrite UFED (Universal Forensic Extraction Device) — комплексний апаратно-програмний продукт, який дозволяє вилучати, дешифрувати та аналізувати дані з мобільних пристроїв, SIM-карт і хмарних сховищ. Програма підтримує тисячі моделей телефонів і забезпечує створення повного логічного або фізичного образу пам'яті.

Однак, попри високу ефективність, Cellebrite орієнтований переважно на вилучення даних і базову перевірку цілісності (через хеш-контроль), але не проводить глибокий аналіз узгодженості метаданих або виявлення логічних аномалій у часових послідовностях.

Іншим потужним інструментом є Magnet AXIOM, який спеціалізується на інтегрованому аналізі артефактів із мобільних пристроїв, комп'ютерів і хмарних джерел.

Його перевагою є гнучкий модуль аналітики, що дозволяє відображати зв'язки між користувачами, подіями, геолокаціями та медіа. Проте, попри розширений набір функцій, Magnet AXIOM залишається переважно системою оглядового типу: він ідентифікує відомі артефакти, але не оцінює їхню автентичність за комплексом індикаторів. Система не завжди виявляє

тонкі зміни у SQLite-базах або несумісності між EXIF-даними та файловими мітками.

Програмний продукт Autopsy, розроблений у межах проєкту The Sleuth Kit, є безкоштовним інструментом цифрової криміналістики, який надає широкі можливості для файлового аналізу, побітового порівняння, створення контрольних хешів і відновлення видалених даних.

Завдяки своїй відкритості Autopsy часто використовується для наукових досліджень і навчальних цілей. Проте цей інструмент орієнтований переважно на класичні комп'ютерні носії (HDD, SSD) і має обмежені можливості щодо сучасних мобільних систем. У ньому відсутній модуль, який би дозволяв інтегровано перевіряти автентичність різних типів артефактів (зображення, SQLite, резервні копії) у межах єдиної логічної моделі.

Ще одним розповсюдженим рішенням є Oxygen Forensic Suite, який забезпечує глибоке вилучення та аналіз даних із мобільних пристроїв, зокрема з резервних копій і хмарних акаунтів. Охуген пропонує багатофункціональну аналітичну панель, інструменти для відновлення видалених повідомлень і побудови графів взаємодій користувача.

Однак, подібно до інших комерційних продуктів, він переважно фокусується на зборі та класифікації даних, а не на верифікації їх достовірності. Аналіз виконується в основному за змістовими атрибутами, тоді як контроль узгодженості часових, структурних і контекстних параметрів потребує додаткової ручної роботи експерта.

Попри значний розвиток індустрії forensic-рішень, жоден із наведених інструментів не забезпечує повноцінного механізму автоматизованої оцінки автентичності даних на рівні логічної взаємодії між різними типами артефактів.

Вони здатні обчислювати хеші, виявляти видалені файли або фрагменти, проте не аналізують глибоку кореляцію між EXIF, SQLite, файловими мітками і резервними копіями. Відповідно, при складних сценаріях фальсифікації, коли зміни узгоджені між кількома джерелами, такі системи не виявляють аномалії.

У сучасній науці зростає інтерес до інтегрованих методів rule-based або machine learning аналізу, які можуть використовувати набір індикаторів — наприклад, розбіжність між часовими мітками, невідповідність хешів мініатюр, зміни у логічній послідовності подій, відсутність кореляції між GPS-логами та журналами викликів.

На основі цих індикаторів формується score-рейтинг достовірності, який дозволяє визначати рівень довіри до конкретного цифрового об'єкта.

Саме на такому принципі базується концепція даної роботи — створення Python-прототипу, що реалізує rule-based підхід до виявлення ознак несанкціонованого втручання.

Таблиця 1.2 — Порівняльна характеристика сучасних інструментів digital forensics та запропонованого rule-based прототипу

Інструмент / метод	Основні функції	Підтримка перевірки автентичності	Рівень автоматизації	Типи аналізованих артефактів	Методика оцінювання достовірності	Обмеження / недоліки
Cellebrite UFED	Фізичне та логічне вилучення даних, декодування, формування звітів	Часткова — контроль хешів та журналювання дій	Високий	Пам'ять пристрою, SIM, SD, резервні копії	Порівняння контрольних сум та логів	Не перевіряє логічну узгодженість між артефактами
Magnet AXIOM	Інтегрований аналіз мобільних, десктопних і хмарних даних	Часткова — перевірка хешів та автентичності копій	Високий	EXIF, SQLite, хмарні журнали, повідомлення	Автоматична ідентифікація джерела + маніфести	Не виконує контекстну оцінку часових міток і взаємозв'язків
Autopsy (Sleuth Kit)	Безкоштовний forensic-аналіз файлової системи, пошук, хеш-каталоги	Технічна автентичність (хеш, контроль цілісності)	Середній	Логічні образи, файлові структури	Побітове порівняння та контроль хешів	Не підтримує аналіз мобільних систем Android/iOS
Oxygen Forensic Suite	Вилучення, відновлення, аналітика комунікацій і геолокацій	Часткова — журналювання та верифікація даних під час копіювання	Високий	Повідомлення, контакти, GPS, хмара	Формальні метадані та ручна перевірка	Відсутня інтегрована логічна верифікація артефактів
Метод хешування (MD5, SHA-256)	Контроль цілісності через обчислення унікального ідентифікатора файлу	Повна на рівні байтової ідентичності	Високий (автоматизований)	Будь-які файли	Бінарне порівняння	Не враховує логічний контекст і кореляції
Rule-based Python-прототип (розроблений у роботі)	Автоматизована перевірка автентичності даних мобільних пристроїв	Повна (технічна, логічна, контекстна)	Високий (повна автоматизація обробки)	EXIF, SQLite, резервні копії, thumbnails, часові мітки	Rule-based система з оцінкою score за сукупністю індикаторів (HASH_MISMATCH, EXIF_FS_INCONSISTENT, SQL_ROW_DIFF, THUMBNAI_MISMATCH, BACKUP_MISMATCH)	Експериментальний прототип; потребує розширення індикаторів та інтеграції з існуючими forensic-інструментами

Висновки до розділу 1

У результаті проведеного теоретичного аналізу встановлено, що забезпечення автентичності цифрових даних мобільних пристроїв є одним із ключових завдань сучасної цифрової криміналістики та кібербезпеки. Мобільні пристрої формують складну багаторівневу інформаційну екосистему, у якій цифрові артефакти — EXIF-метадані, бази SQLite, журнали подій, резервні копії — мають високу доказову цінність, але водночас залишаються вразливими до фальсифікацій.

Досліджено, що найбільш поширеними загрозами достовірності є маніпуляції метаданими, часовими мітками, журналами системи, а також підміна резервних копій. Традиційні методи перевірки, засновані на хешуванні або побітовому порівнянні, не здатні виявляти контекстні зміни та логічні невідповідності між артефактами.

Огляд міжнародних стандартів (ISO/IEC 27037, NIST SP 800-101, ENFSI Guidelines) показав, що світова практика орієнтується на дотримання принципів збереження цілісності та ланцюга зберігання доказів (chain of custody), однак не пропонує готових рішень для автоматизованої перевірки автентичності в динамічному середовищі Android.

Порівняння сучасних forensic-інструментів (Cellebrite, Magnet AXIOM, Autopsy, Oxygen Forensic Suite) підтвердило, що більшість систем орієнтовані на вилучення та каталогізацію даних, але не виконують комплексного логічного аналізу узгодженості цифрових об'єктів. Це створює потребу у розробці нових підходів, що дозволяють оцінювати автентичність не лише на рівні файлів, а й на рівні взаємозв'язків між ними.

Таким чином, результати теоретичного дослідження обґрунтовують доцільність створення rule-based методики перевірки достовірності цифрових артефактів та розробки Python-прототипу системи автоматизованого аналізу автентичності мобільних даних.

2 АНАЛІЗ СТРУКТУРИ ДАНИХ ТА ДЖЕРЕЛ ЦИФРОВИХ АРТЕФАКТІВ МОБІЛЬНИХ ПРИСТРОЇВ

У цьому розділі зосереджено увагу на тих компонентах екосистеми Android, які безпосередньо впливають на можливість перевірки автентичності цифрових артефактів, та на процедурах підготовки цих артефактів до подальшого індикаторного аналізу.

На відміну від загального теоретичного огляду, тут розглядаються лише форенсично релевантні елементи: фізичні області зберігання й простори ізоляції застосунків (/data, sandbox-папки, каталоги мультимедіа), службові реєстри системи (MediaStore, записи інсталяцій і налаштувань) та типові сховища даних користувача (мультимедіа з EXIF, бази SQLite, кеші мініатюр, резервні копії).

Метою є чітко визначити, що саме і де саме вилучати, у якому форматі це фіксувати та як нормалізувати часові й структурні атрибути для коректної кореляції між незалежними джерелами.

Подальший виклад формалізує типи артефактів з позиції їхньої доказової цінності та вразливості до модифікацій, окреслює поширені сценарії підробки (маніпуляції EXIF-полями, зміщення часових міток, редагування записів у SQLite, підміна резервних копій, розбіжності між мініатюрами та оригіналами) і пов'язує ці сценарії з конкретними ознаками, придатними до автоматизованого виявлення.

Окремо встановлюються процедурні вимоги — створення хеш-маніфестів до/після доступу, робота з read-only копіями, експорт EXIF і службових таблиць у структурованому вигляді (CSV/JSON), збереження транзакційних файлів SQLite (-wal, -shm), уніфікація часу до UTC з фіксацією часової зони та переходів DST, а також виділення незалежних контрольних

точок (MediaStore, маніфести резервного копіювання) для подальшої кореляції.

Результатом розділу є відтворюваний набір підготовлених атрибутів, які слугують вхідними даними для rule-based системи індикаторів і скорингової моделі.

2.1 Архітектура операційної системи Android у контексті криміналістичного аналізу

Для завдання перевірки автентичності цифрових артефактів Android розглядається як багатоваршова система з чітко відокремленими зонами зберігання, механізмами ізоляції застосунків і сервісами каталогізації вмісту.

На базовому рівні важливими є файлові розділи /system, /vendor, /product (переважно тільки для читання), /data (користувацькі та апкові дані) і /sdcard (emulated external storage; фізично розміщений у /data/media/0 для основного користувача).

У контексті форенстики нас цікавлять насамперед /data та всі похідні від нього простори зберігання, оскільки саме там існують артефакти, потрібні для кореляції часових міток і логічної узгодженості подій.

Ізоляція застосунків реалізується через sandbox-каталоги /data/user/0/<package>/ (або /data/data/<package>/ у старіших версіях), де зберігаються локальні SQLite-бази (.../databases/), конфігурації (.../shared_prefs/), кеші (.../cache/) та службові файли.

Для перевірки автентичності критичне значення мають саме локальні бази даних і супутні журналові файли -wal/-shm, які фіксують транзакційні операції; їх вилучення разом із основною базою є обов'язковою умовою коректної інтерпретації послідовності вставок/видалень/оновлень. Окремі системні сервіси ведуть реєстри стану пристрою та ПЗ:

/data/system/packages.xml, packages.list, каталоги /data/system/users/0/, а також сховища постачальників контенту (content providers), серед яких ключову роль у мультимедійній сфері відіграє MediaStore (бази com.android.providers.media).

Наявність scoped storage (починаючи з Android 10) змінює патерни доступу до файлів, однак артефакти продовжують індексуватись MediaStore з присвоєнням ідентифікаторів, атрибутів медіатипів і часових полів.

Саме MediaStore виступає незалежною “третьою” точкою істини щодо дат додавання/модифікації медіафайлів, яка використовується для кореляції з EXIF та файловими мітками. Для фотографій/відео важливі колонки на кшталт date_added (секунди UNIX-епохи), date_modified, mime_type, bucket_id, а у старіших версіях — шлях _data (нині позначений як застарілий, але все ще може бути присутнім).

Нові релізи Android частіше повертають URI формату content:// замість прямого шляху, що вимагає коректного витягання атрибутів через API або інструменти, які підтримують роботу з content providers.

Файлова система зазвичай представлена EXT4 або F2FS; часові атрибути файлів включають mtime (модифікація вмісту), ctime (зміна метаданих inode; не плутати зі “створенням”), atime (останній доступ, часто вимкнений із міркувань продуктивності).

На деяких пристроях доступний атрибут створення (birth/creation time), але покладатися на нього не можна — він не гарантований. Це накладає методичну вимогу: події “створення” артефактів слід реконструювати з поєднання EXIF DateTimeOriginal, MediaStore date_added та контекстних записів (наприклад, інсталяцій/оновлень застосунків), а не з одного поля файлової системи. Для виявлення підробок критичною є внутрішня узгодженість часових рядів між цими рівнями.

EXIF-метадані зберігаються безпосередньо в контейнерах зображень/відео (JPEG/HEIC/MP4) і містять поля `DateTimeOriginal`, `CreateDate`, `ModifyDate`, геодані (`GPSLatitude`, `GPSLongitude`), відомості про пристрій (`Make`, `Model`, іноді `SerialNumber`, `LensModel`), а також параметри зйомки.

Через доступність редагування EXIF стандартними інструментами ці поля не можуть розглядатися як єдине джерело істини; їх роль у перевірці автентичності — бути зіставленими з іншими рівнями: файловими атрибутами, записами `MediaStore` та кешами мініатюр.

Окрему доказову цінність мають кеші `thumbnails` (системні та апкові): вони нерідко зберігають попередні версії зображень або згортки контенту, що дозволяє виявляти `THUMBNAIL_MISMATCH` — невідповідність між поточним медіафайлом і його колишнім “зліпком”.

Журнали комунікацій і системні сховища подій реалізовані через постачальників контенту та локальні бази. Типові розташування: журнали дзвінків — у просторі `com.android.providers.contacts` (база `callog.db` або відповідна таблиця в об’єднаній БД контактів), SMS/MMS — у `com.android.providers.telephony` (база `mmssms.db`), мультимедіа — у `com.android.providers.media` (внутрішні/зовнішні бази каталогу медіа).

У цих таблицях часові колонки зберігаються у мілісекундах/секундах UNIX-епохи (неоднорідність форматів — типовий ризик помилкової інтерпретації). Для підтвердження послідовностей подій важливими є також системні реєстри інсталяцій/оновлень пакунків: зіставлення моменту появи артефакту з часом встановлення або першого запуску відповідного застосунку дозволяє виявляти неможливі сценарії (дані “існують” до появи ПЗ, яке їх генерує).

Починаючи з Android 7, основним механізмом захисту даних є File-Based Encryption (FBE) з ключами на рівні користувача/профілю та типами захищеності (наприклад, доступність після розблокування).

Для криміналістики це означає, що логічне вилучення в більшості випадків здійснюється вже з розблокованого пристрою або зі захищених копій (backup), а фізичне знімання флеш-пам'яті без ключів часто позбавлене практичного сенсу. У правовому та процедурному плані це підсилює значущість ланцюга збереження (Chain of Custody) та журналювання дій: будь-який доступ до розшифрованих даних повинен бути фіксований, а отримані копії негайно хешуються.

Резервні копії (локальні архіви ADB/tar, вміст Google Drive/OEM-backup) і їх маніфести є цінним контрольним шаром. Вони надають метадані про включені файли, їх дати та контрольні суми; порівняння “копія ↔ поточний стан” дозволяє виявляти BACKUP_MISMATCH — розбіжності у структурі й часі, характерні для підміни або відкату історії.

Зауважимо, що в умовах scoped storage копії можуть не містити повного шляху, натомість — ідентифікатори контенту; тому під час нормалізації потрібна уніфікація посилань (URI/ID → логічний об'єкт).

Оскільки часові ряди є центральним елементом верифікації автентичності, джерела часу повинні розглядатись критично. Android синхронізує годинник через NTP/NITZ, локальні tzdata отримують оновлення, а зміна часової зони/літнього часу (DST) може створювати легітимні “стрибки”.

Тому всі часові атрибути під час попередньої обробки зводяться до UTC з фіксацією вихідної TZ/DST у метаданих витягу; подальша кореляція між EXIF → MediaStore → файловими атрибутами → базами застосунків здійснюється на єдиній часовій шкалі з контрольними допусками Δt для конкретних типів подій.

Нарешті, важливо враховувати операційні зміни, які не є фальсифікацією, але змінюють артефакти: оновлення застосунків і міграції схем БД, оптимізація (VACUUM), пересканування медіакаталогу (reindex), клонування даних при міграції профілю користувача.

Саме тому архітектурне розуміння зв'язків між sandbox-даними застосунків, файловими атрибутами та каталогом MediaStore є передумовою для коректної інтерпретації виявлених розбіжностей: ми відрізняємо операційно очікувані ефекти від ознак потенційної підміни (SQL_ROW_DIFF, EXIF_FS_INCONSISTENT, THUMBNAIL_MISMATCH, BACKUP_MISMATCH).

2.2 Типи цифрових артефактів: EXIF-метадані, SQLite-бази, системні журнали, резервні копії

У контексті перевірки автентичності мобільних даних найвищу доказову цінність мають чотири класи артефактів: метадані мультимедіа (насамперед EXIF), локальні сховища застосунків на основі SQLite, системні журнали й реєстри, а також резервні копії зі своїми маніфестами. Їхня особливість не лише в окремому змісті, а у взаємній кореляції часових і структурних атрибутів. Саме ця кореляція дає можливість відрізнити легітимні операційні зміни від ознак цілеспрямованого втручання.

EXIF-метадані зберігаються в контейнерах JPEG/HEIC/MP4 і описують момент та умови фіксації кадру:

- DateTimeOriginal і CreateDate як базові часові атрибути;
- GPSLatitude, GPSLongitude, іноді GPSAltitude як геоприв'язка;
- Make, Model, інколи SerialNumber або LensModel як апаратний контекст;
- параметри експозиції та, за наявності, вбудовану мініатюру.

Для потреб автентифікації ці поля не розглядаються ізольовано. Будь-яке значення часу чи координат звіряється з файловими атрибутами (мінімум mtime, за можливості birth/creation time) і записами медіакаталогу системи (MediaStore), що відображають факт індексації та доступу до об'єкта з боку ОС.

Невідповідність між DateTimeOriginal і часовими полями MediaStore, а також розбіжність між поточним кадром і кешованою мініатюрою (за вмістом або датами) є типовими індикаторами можливого редагування. Практично це вимагає витягання EXIF у структурований формат (CSV/JSON) з суворою уніфікацією часових зон і точності до секунди або мілісекунди (усі часові відмітки нормалізуються до UTC із фіксацією вихідної TZ/DST у метаданих витягу).

Global Positioning System	
GPS Altitude	31.9 m
GPS Latitude	6deg 14' 7.620"
GPS Longitude	106deg 49' 30.210"
Image Information	
Date and Time	2018:08:24 15:47:27
Manufacturer	Apple
Model	iPhone 6s
Photograph Information	
Aperture	F2.2
Exposure Bias	0 EV
Exposure Mode	Auto
Exposure Program	Auto
Exposure Time	1/874 s
Flash	No, auto
FNumber	F2.2
Focal Length	4.2 mm
ISO Speed Ratings	25
Metering Mode	Multi-segment
Shutter speed	1/874 s
White Balance	Auto

Рисунок 2.1 — Приклад таблиці EXIF

SQLite-бази є основним носієм прикладних журналів активності — дзвінки, SMS/MMS, історії месенджерів, медіатеки, кеш-таблиці, налаштування.

Фізично вони розміщуються у sandbox-просторах застосунків (/data/user/0/<package>/databases/) або у провайдерів контенту системи.

Для коректної інтерпретації обов'язковим є вилучення не лише основного файлу, а й супровідних журналів `-wal` і `-shm`, що фіксують транзакції та буферизацію сторінок.

З позиції автентичності ключовими є часові колонки (часто як епоха у секундах/мілісекундах), первинні ключі та автоінкременти, зовнішні ключі й службові індекси.

Будь-яка атипова послідовність ID, «ущільнення» після `VACUUM`, розриви в автоінкременті, поява записів із часовими мітками, що передують інсталяції або першому запуску відповідної програми, трактуються як сигнали для поглибленого аналізу.

Під час підготовки даних фіксується схема (`.schema`), статистика таблиць, а також виконується безпечний дамп у формат, придатний для повторного відтворення результатів. Формат часових полів у різних БД неоднорідний, тому на етапі нормалізації застосовується єдина шкала UTC, а вихідні одиниці виміру часу зберігаються у службовому описі витягу.

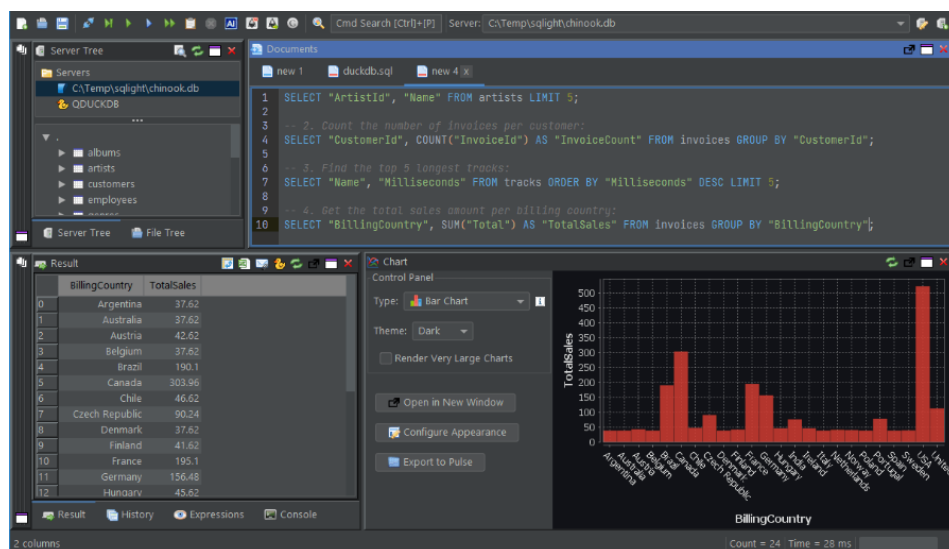


Рисунок 2.2 — Приклад роботи з SQLite-базою

Системні журнали та реєстри виконують роль незалежного шару підтвердження подій.

До них належать каталоги пакунків і історія інсталяцій/оновлень (наприклад, `packages.xml`, пов'язані списки у `/data/system`), записи медіакаталогу (MediaStore), а також журнали подій ядра і користувацького простору, доступні в межах логічного вилучення.

З практичної точки зору саме поля MediaStore на кшталт `date_added/date_modified`, MIME-тип і, у старіших релізах, шлях `_data` дозволяють підтвердити або спростувати хронологію появи і модифікації файлів незалежно від EXIF і файлової системи.

Порівняння моментів інсталяції/оновлення застосунку з першою появою притаманних йому артефактів дає змогу виявляти неможливі послідовності («дані існують до появи ПЗ, що їх генерує»), тоді як аналіз журналів створює підґрунтя для зв'язування дій користувача, системних сервісів і результатних змін у БД.

Резервні копії та їхні маніфести є контрольним «знімком» стану системи у часі. Вони можуть існувати локально (архіви ADB/tar, OEM-бекапи) або у хмарних сервісах.

У кожному випадку дослідницький інтерес зосереджено на переліках включених об'єктів, їхніх контрольних сумах, датах створення і відновлення. Порівняння поточного стану з найближчою за часом копією виявляє підміну історії (наприклад, «нова» копія з «старими» датами, інші набори файлів у тих самих директоріях) і формує індикатор невідповідності резервування фактичній еволюції артефактів.

Враховуючи особливості `scoped storage` та URI-ідентифікаторів, у маніфестах може не бути повних шляхів, тому під час попередньої обробки кожен запис приводиться до логічного представлення об'єкта, яке придатне для подальшої кореляції.

Окремою категорією, що часто недооцінюється, є кеші мініатюр. Вони можуть існувати як у системних каталогах мультимедіа, так і в приватних кешах застосунків і мають власний життєвий цикл формування й оновлення.

Порівняння вмісту кешованої мініатюри з поточним файлом (на рівні хешів або простих перцептивних дескрипторів) дозволяє зафіксувати ситуації, коли оригінал було редаговано без відповідного оновлення кешу.

Цей факт сам по собі не доводить фальсифікацію, однак у поєднанні з часовими зсувами та розбіжностями EXIF/MediaStore створює стійкий візерунок невідповідності.

2.3 Потенційні сценарії підробки та модифікації цифрових даних

Практичне завдання перевірки автентичності мобільних артефактів передбачає ідентифікацію не лише фактів зміни вмісту, а й логічних невідповідностей між незалежними джерелами часу та структури.

Типові сценарії втручання реалізуються на різних рівнях — від редагування метаданих мультимедіа до цілеспрямованого перепакування баз даних і маніпуляцій із резервними копіями.

У контексті подальшого rule-based аналізу ці сценарії доцільно описувати через спостережувані патерни, що зводяться до індикаторів невідповідності й надалі агрегуються у скорингову оцінку.

Найбільш розповсюдженим є редагування EXIF-полів у JPEG/HEIC/MP4, а саме:

1. зміщення DateTimeOriginal та CreateDate для корекції хронології,
2. ін'єкція GPS-координат,
3. підміна Make/Model з метою імітації іншого пристрою.

Такі зміни рідко залишають технічні артефакти у контейнері, проте проявляються як логічна розбіжність із файловими атрибутами (mtime/ctime) і записами MediaStore (date_added/date_modified).

Якщо час у EXIF виявляється «старшим» за момент індексації у MediaStore на величину, що перевищує очікуване вікно допуску після зйомки, або «молодшим» за файлову мітку модифікації без наявної операції редагування (наприклад, трансформації/перекодування), формується сукупність ознак, яку в роботі ми фіксуємо як EXIF_FS_INCONSISTENT.

Додатковим сигналом виступає невідповідність кешованої мініатюри поточному зображенню — THUMBNAIL_MISMATCH, коли вміст мініатюри відображає попередню версію кадру або інші параметри, ніж має оригінал.

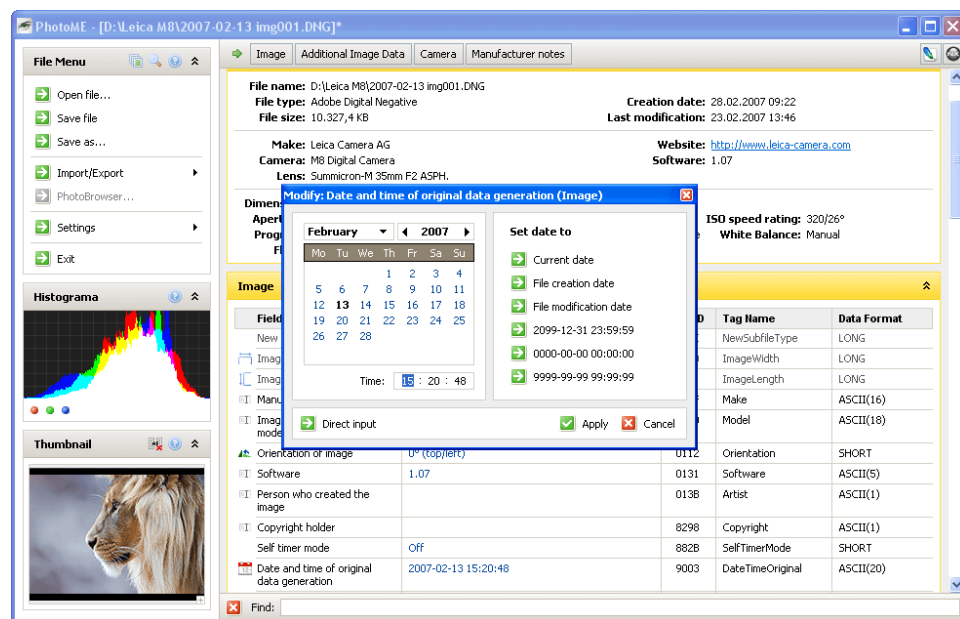


Рисунок 2.3 — Приклад редагування EXIF-полів

Другий клас втручань охоплює маніпуляції з часовими рядами через системний годинник, часові зони та літній час.

Класичні прийоми — встановлення «минулої» дати перед створенням артефакту, зміна TZ/DST після створення, копіювання файлів із збереженням вихідних міток.

На практиці це породжує петлі або інверсії у часовій лінії подій. Медіафайл «з'являється» до інсталяції камери, запис дзвінка має мітку, що передуює створенню контакту, повідомлення в чаті датуються моментом до першого запуску застосунку.

Такі конфлікти не завжди тягнуть за собою HASH-розбіжності, проте проявляються як неможливі послідовності у нормалізованій хронології.

У скоринговій моделі вони підсилюють індикатори EXIF_FS_INCONSISTENT та SQL_ROW_DIFF, а також можуть фіксуватись додатковим службовим прапорцем «`timeline_conflict`» для пояснюваності.

Третій, найбільш «гнучкий» рівень — редагування SQLite-сховищ (дзвінки, SMS/MMS, журнали месенджерів, медіатеки).

Навіть точкові правки одного поля здатні порушити внутрішню цілісність БД. Змінюється логіка автоінкрементів, виникають «дірки» у послідовностях первинних ключів, розходяться зовнішні ключі, порушується конгруентність індексів.

Використання VACUUM після серії правок призводить до «ущільнення», що маскує видалення, а відсутність або обрив журналів `-wal/-shm` у момент «критичних» змін позбавляє нас очікуваних слідів.

У сукупності ці ознаки проектуються в індикатор SQL_ROW_DIFF, який ми формуємо не як «одну умову», а як композицію простих тестів — перевірка моногонічності ID, коректності часових форматів і одиниць виміру, відповідності часових полів інсталяційним подіям пакунків, наявності узгоджених записів у суміжних таблицях (`threads/messages` тощо).

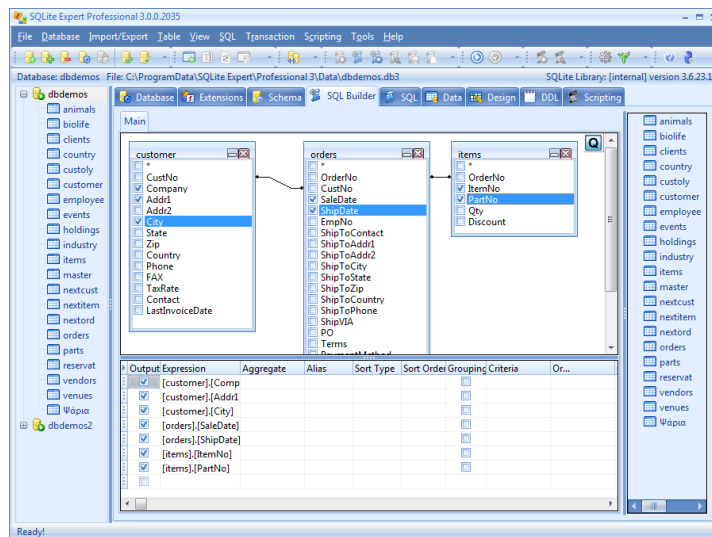


Рисунок 2.4 — Приклад редагування SQLite

Окремого розгляду потребує сценарій підміни історії через резервні копії.

Тут характерні два вектори — імпорт «очищеної» копії замість актуального стану та модифікація маніфестів/контрольних сум у самих архівах. Після відновлення такої копії артефакти зникають, з'являються «ретроспективно правильно» марковані файли, а каталоги MediaStore переіндексовують вміст з новими date_added.

Якщо склад копії, її контрольні суми та часові поля не узгоджуються з фактичною еволюцією даних, ми отримуємо BACKUP_MISMATCH. Ознаками є відсутність очікуваних об'єктів, несумісність дат у маніфестах і різні набори файлів за однаковими логічними локаціями.

Антифоренсичні практики інколи включають перекодування відео/зображень і «добросовісне» редагування в редакторах, що переписують EXIF і перераховують часові поля, — спроба легітимізувати зміну вмісту.

Такі операції самі по собі не є злочинними, але залишають узгоджені технічні сліди. Змінюються MIME-типи або кодеки, зникають оригінальні теги камер, у MediaStore формується очікуваний зсув між `date_modified` і `date_added`.

У цих випадках скоринг не має реагувати так само, як на цілеспрямовану фальсифікацію. Ми знижуємо вагу окремих локальних індикаторів і вимагаємо підтвердження з інших шарів (наприклад, `timeline`-кореляцією або крос-перевіркою кешів мініатюр).

Саме тому для кожного індикатора визначаються пороги спрацьовування і контекстні умови, які відсікають легітимні сценарії обробки.

Слід враховувати і «міграційні» джерела аномалій, наприклад перенесення профілю користувача або окремих директорій між пристроями, повторна індексація галереї, оновлення застосунків із зміною схем БД.

Такі події здатні створити хвилю «штучних» розбіжностей у мітках часу, відмінності в ID-просторах, розриви посилань у медіатеці. У методиці вони маркуються як нейтральні тригери і потребують підтвердження іншими індикаторами, перш ніж впливатимуть на підсумковий бал.

Інакше кажучи, одинична технічна невідповідність не повинна автоматично трактуватись як ознака підробки — вагу сигналу визначає узгоджена «картина» з кількох незалежних джерел.

Узагальнюючи, кожен зі згаданих сценаріїв відбивається у специфічному наборі спостережуваних ефектів, які ми проектуємо на індикатори:

1. EXIF_FS_INCONSISTENT для розривів між EXIF, файловими мітками й MediaStore;
2. THUMBNAIL_MISMATCH для невідповідності кешованого та актуального вмісту;
3. SQL_ROW_DIFF для структурно-часових аномалій у SQLite;
4. BACKUP_MISMATCH для конфліктів між резервними копіями та поточним станом;
5. HASH_MISMATCH для байтових розбіжностей там, де очікується ідентичність.

Подальша скоринг-модель об'єднує ці індикатори з урахуванням контекстних допусків і формує числову оцінку достовірності, яка вже використовується для прийняття рішення у практичному аналізі.

Такий підхід дозволяє відрізнити технічно правдоподібні редагування від узгоджених фальсифікацій, знижуючи частку хибнопозитивних висновків і підвищуючи доказову силу виявлених невідповідностей.

2.4 Проблематика співставлення часових міток та логічної послідовності подій

Перевірка достовірності мобільних даних у середовищі Android неможлива без відтворення єдиної хронології подій, оскільки часові атрибути формуються різними компонентами системи, зберігаються у різних форматах і мають різну семантику.

До основних джерел часу належать файлові атрибути (mtime, іноді birth/creation time, а також ctime, що відображає зміну метаданих inode), записи медіакаталогу (передусім date_added і date_modified, які фіксують моменти індексації та оновлення), часові поля контейнерів мультимедіа (EXIF DateTimeOriginal, CreateDate, GPS-час без явної TZ), а також стовпці у таблицях SQLite (значення UNIX-епохи в секундах або мілісекундах).

Кожне з цих полів описує свій етап життєвого циклу артефакта, від створення сцени камерою, запису на носій, індексацію системою, редагування чи міграцію. Через це пряме порівняння «як є» призводить до помилок інтерпретації, а будь-які висновки мають спиратися на взаємну кореляцію джерел.

Головні труднощі зумовлені неоднорідністю форматів і точності (секунди проти мілісекунд), відсутністю часової зони в окремих EXIF-полях, асинхронними процесами ОС (затримки індексації медіатеки, відкладене формування кешів), впливом змін TZ/DST, а також легітимними операційними подіями (оновлення застосунків, VACUUM у БД, повторна індексація) і, окремо, можливими ручними втручаннями у системний час.

Тому першим кроком є нормалізація. Усі часові мітки зводяться до UTC з ретельною фіксацією початкової часової зони, стану переходів DST та вихідних одиниць виміру. Для EXIF без TZ застосовується контекстна прив'язка (до поточної TZ пристрою в період події, до сусідніх артефактів або мережевих журналів часу).

Після нормалізації доцільно встановити правила інтерпретації для класів об'єктів.

Для фотографій базовою подією є момент фіксації сцени (EXIF `DateTimeOriginal`), але він повинен бути узгоджений з моментом появи в індексі (`date_added`) і з файловою міткою модифікації, яка закономірно зсувається вперед у випадку редагування чи перекодування.

Для журналів комунікацій пріоритет мають часові колонки прикладних таблиць, однак вони співвідносяться з подіями інсталяції/оновлення відповідних пакетів та, за наявності, з іншими залежними подіями (наприклад, створення контакту, перший запуск застосунку).

Для резервних копій «якорями» виступають дати формування/відновлення та інформація маніфестів. Їхня хронологія має пояснювати появу або зникнення файлів, а також очікувану переіндексацію у медіакаталозі.

Критично важливо відмежувати технічно очікувані лаги від справжніх несумісностей. Затримка індексації медіатеки, асинхронне оновлення мініатюр, копіювання файлів зі збереженням міток або пакетний імпорт із зовнішнього носія здатні створювати відчутні часові відхилення без ознак підробки.

Тому встановлюються допустимі вікна різниці часу для типових пар подій (зйомка → індексація; редагування → модифікація; імпорт → індексація; відновлення з копії → поява в каталозі), які залежать від класу артефактів і характеру операції.

Натомість хронологічні інверсії (наявність артефакта до інсталяції застосунку, «повернення назад» в межах часових рядів без пояснюваної операції) вважаються несумісними і потребують окремої уваги під час подальшого аналізу.

Остаточне подання часу формують як єдину подієву лінійку, де кожен артефакт описується набором пов'язаних подій (створення/фіксація, запис, індексація, редагування, резервування/відновлення), між якими визначено очікуваний причинно-часовий порядок і припустимі лаги.

Усі вихідні значення, перетворення (у тому числі конверсія точності й TZ), а також контекстні припущення документуються в службових матеріалах (протоколах витягу, журналах попередньої обробки, хеш-маніфестах), що забезпечує відтворюваність і процесуальну придатність.

Такий підхід дозволяє інтерпретувати часові відхилення у відповідному контексті, мінімізувати хибні трактування та створити надійну основу для подальших розділів, де будуть формалізовані правила оцінки узгодженості та механізми їх програмної реалізації.

Висновки до розділу 2

Цей розділ сформував прикладну основу для подальшої перевірки автентичності мобільних артефактів Android і чітко визначив, з якими даними та в якому вигляді слід працювати.

Уточнено архітектурні зони інтересу та роль системних індексаторів, насамперед MediaStore як незалежної «третьої» точки істини для дат появи та модифікації медіафайлів, що надалі співставляються з EXIF і файловими мітками уніфіковано й відтворювано.

Таке співставлення неможливе без розуміння поведінки файлової системи та її часових атрибутів, зокрема обмеженості поля creation time і необхідності реконструювати момент створення об'єкта через поєднання EXIF DateTimeOriginal, MediaStore date_added та контекстних записів про встановлення або оновлення застосунків.

У межах розділу систематизовано класи артефактів із фокусом на їх доказову цінність і типові слабкі місця. Для мультимедіа визначено перелік полів EXIF і підхід до їх кореляції з часовими рядами ОС, для прикладних даних уточнено вимоги до повних дамів SQLite разом з журналами транзакцій wal і shm, а для системних реєстрів показано, як події інсталяції та індексації слугують контрольними маркерами у хронології.

Сукупно це дозволяє відрізнити очікувані технологічні лаги від аномалій та формувати стабільні патерни невідповідностей, які пізніше проєктуються в індикатори.

Розділ також зафіксував коло загроз, що спотворюють достовірність артефактів, і пояснив, чому класичні побітові методи або просте хешування не покривають логічні неузгодженості між різними шарами даних.

Це стосується маніпуляцій EXIF, зсувів системного часу, редагування прикладних таблиць SQLite та підміни резервних копій, які змінюють картину подій без видимої втрати цілісності файлу.

Окремо акцентовано на впливі часових міток у файловій системі та логах, а також на уразливості SQLite до точкових правок і «уцільнення» історії, що потребує суворої процедури видалення та нормалізації для коректної інтерпретації результатів.

Ключовим результатом є підготовлений, відтворюваний корпус даних з єдиною часовою шкалою і документованим контекстом TZ та DST, у якому EXIF, MediaStore, файлові атрибути та прикладні журнали узгоджуються за правилами пріоритетів і допустимими вікнами різниці.

3 РОЗРОБКА МОДЕЛІ ПЕРЕВІРКИ АВТЕНТИЧНОСТІ МОБІЛЬНИХ ДАНИХ

У цьому розділі формулюється практична методика перевірки автентичності мобільних артефактів на основі узгодженого аналізу незалежних джерел часу і структури даних.

Методика спирається на підготовлений у попередньому розділі набір артефактів з єдиною часовою шкалою та документованим контекстом і переходить від опису джерел до правил, за якими фіксуються ознаки можливих втручань.

Основний принцип полягає в поєднанні простих перевірок на рівні файлів і метаданих з кореляційними тестами між EXIF файловими атрибутами MediaStore та прикладними записами у SQLite щоб замість одиничних спостережень отримати узгоджений візерунок поведінки даних.

Методика побудована як rule based підхід з явними індикаторами ознак невідповідності та прозорою логікою прийняття рішення.

Кожен індикатор визначає конкретну перевірку із входами умовами спрацювання та очікуваними винятками а результат індикаторів агрегується у скорингову оцінку достовірності з вагами і порогами класів.

Така конструкція забезпечує відтворюваність пояснюваність і можливість незалежної перевірки результатів, оскільки кожен етап супроводжується протоколюванням вихідних значень перетворень і проміжних висновків. У межах підходу враховано технологічні лаги індексації формування кешів і відновлення з резервних копій а також легітимні операційні події що можуть змінювати часові мітки без ознак фальсифікації.

Вихід методики визначається як рішення про рівень довіри до артефакта та структурований набір пояснювальних матеріалів які містять перелік спрацювань індикаторів з їхніми значеннями карту часових розбіжностей і посилання на первинні витяги.

Вимоги до якості результатів включають мінімізацію хибнопозитивних спрацювань завдяки агрегуванню по кількох незалежних джерелах, контроль відтворюваності через незмінний конвеєр нормалізації та облік контексту часової зони і переходів літнього часу.

Обмеження підходу стосуються неповноти даних за умов зашифрованих або частково недоступних сховищ, а також випадків коли артефакти породжені сторонніми застосунками зі специфічною логікою ведення журналів.

3.1 Загальна концепція запропонованого підходу

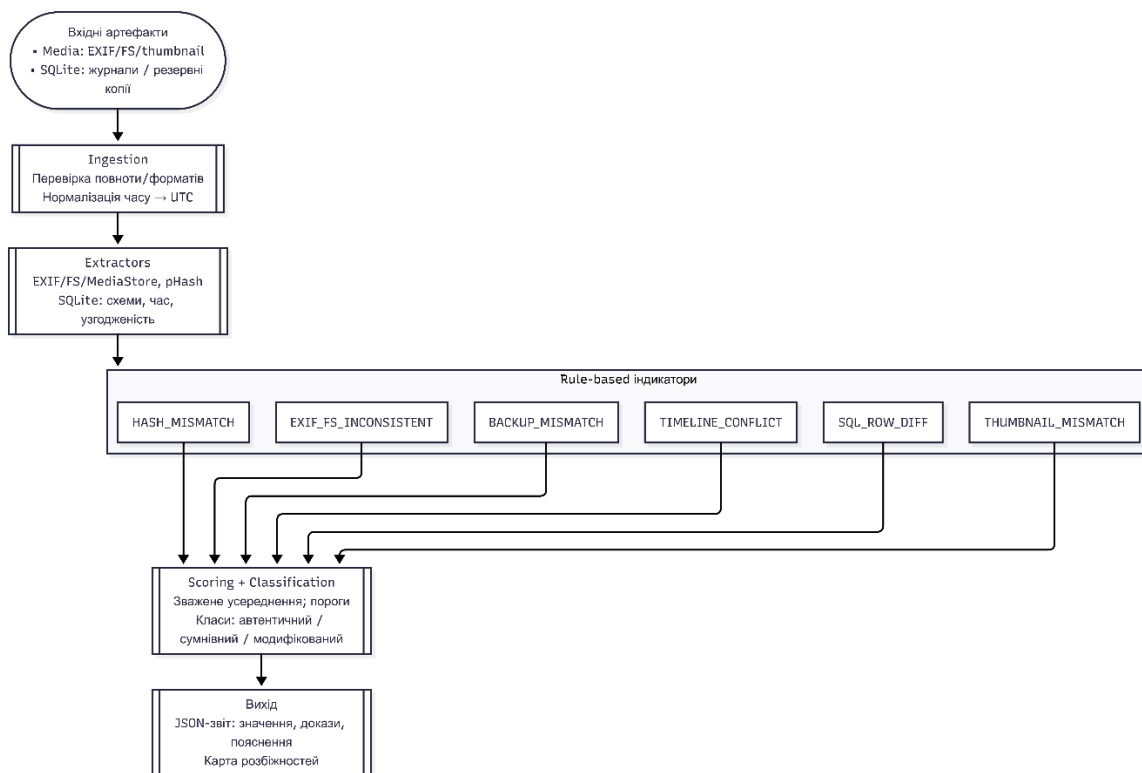


Рисунок 3.1 — Блок схема запропонованого підходу

Запропонований підхід розв’язує практичну проблему підтвердження автентичності мобільних артефактів не через ізольовані перевірки, а через системну кореляцію незалежних джерел даних і часу.

Ідея полягає у багатошаровій валідації, де фактичний вміст і метадані медіафайлів, системні записи індексації, прикладні журнали у базах SQLite та відомості з резервних копій розглядаються як взаємопов’язані свідчення однієї подієвої історії.

Така постановка дозволяє виявляти не лише прямі ознаки редагування, а насамперед логічні несумісності між рівнями системи, які найчастіше й свідчать про фальсифікацію.

Новизна підходу полягає у формалізації цієї логіки у вигляді прозорої rule based системи індикаторів з подальшим агрегуванням результатів у кількісну оцінку достовірності, яка супроводжується відтворюваними доказовими матеріалами.

Концептуально методика спирається на три опори.

Перша опора це нормалізований часовий контекст де всі мітки часу з EXIF файлової системи медіакаталогу і прикладних баз зведено до єдиної шкали з фіксацією часової зони і переходів літнього часу.

Друга опора це індикаторна модель яка перетворює типові патерни невідповідностей на формальні ознаки рівня артефакта, зокрема розрив між EXIF і файловими мітками та індексацією, аномалії у послідовностях і схемах SQLite, невідповідність мініатюр поточному зображенню і конфлікти між резервними копіями та актуальним станом.

Третя опора це пояснюваний скоринг у якому кожен індикатор має вагу та поріг спрацювання а підсумковий бал доповнюється коротким обґрунтуванням на основі конкретних зіставлень часу і вмісту.

Сукупно це забезпечує ідентичність результатів при повторному запуску і дає можливість зовнішньої перевірки висновків.

Значимість підходу визначається тим що він закриває методичний розрив між екстракцією даних і їх доказовою верифікацією.

Індустріальні інструменти орієнтовані на збирання та зручну навігацію по артефактах однак не надають автоматизованої логіки узгодження часових рядів на різних рівнях.

У випадках коли зловмисник узгоджено змінює декілька шарів даних окреслені засоби рідко сигналізують про проблему оскільки кожний шар окремо виглядає правдоподібно.

Натомість запропонована схема порівнює шари між собою вводить допуски для легітимних лагів і відсікає одиничні локальні відхилення як недостатні для висновку.

Це безпосередньо знижує частоту хибнопозитивних спрацювань і підвищує довіру до автоматичних висновків на етапі попереднього аналізу.

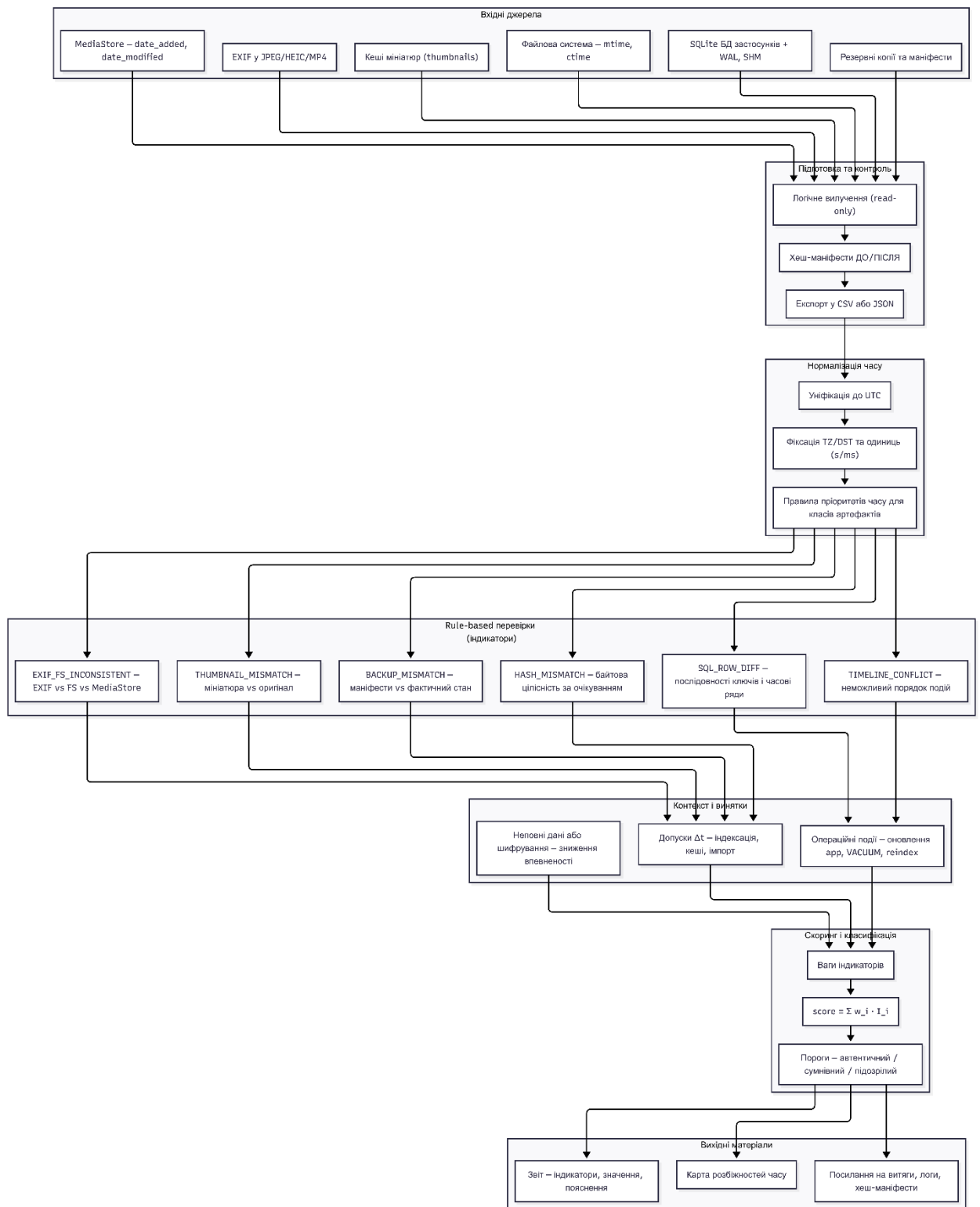


Рисунок 3.2 — Детальна блок схема запропонованого підходу

Наукова новизна проявляється у трьох рішеннях.

По перше, запропоновано подієвий опис артефакта де кожен об'єкт представляється мінімальним набором причинно часових подій, що дозволяє застосовувати однакові методики оцінки цілісності до різних типів даних.

По друге, розроблено формальний словник індикаторів з чіткими умовами спрацювання, винятками і пріоритетами джерел часу.

По третє, введено пояснюваний скоринг що не лише дає числовий бал але й матеріалізує підстави рішення у вигляді карт розбіжностей, логів нормалізації і посилань на первинні витяги, завдяки чому результат може бути незалежно перевірений іншим експертом або судом.

Практична цінність полягає у тому що правило орієнтований аналіз легко вбудовується у наявні процеси.

Його можна запускати перед глибинним оглядом у комерційних платформах і використовувати як фільтр ризику для пріоритизації кейсів.

У межах лабораторного пайплайна він виконує роль стандартизованої процедури попередньої валідації, яка документує всі перетворення, фіксує контрольні суми і зберігає проміжні артефакти.

Така організація спрощує підтримку ланцюга зберігання і робить результати повторюваними у часі, незалежно від версій інструментів та людського фактора.

Водночас методика проєктована на розширення, адже нові індикатори і джерела контексту можуть бути додані без зміни основної логіки, а ваги та пороги можуть калібруватися під конкретний клас пристроїв або операційне середовище. Завдяки цьому підхід масштабований і здатний еволюціонувати разом із практикою цифрової криміналістики.

Таким чином, загальна концепція поєднує уніфікований часовий простір, формалізовані індикатори невідповідності та пояснюваний скоринг у один відтворюваний процес.

Це створює відсутню ланку між фактом наявності даних і обґрунтованим висновком про їхню достовірність, підвищує доказову силу результатів і забезпечує практичну придатність методики для використання у реальних розслідуваннях.

3.2 Вибір методів перевірки автентичності та оцінки достовірності

У цій роботі обрано поєднання детермінованих перевірок і кореляційного аналізу між незалежними джерелами даних. Рішення зумовлене вимогами відтворюваності, пояснюваності та процесуальної придатності, адже результати мають бути зрозумілими для експерта і перевірними у повторному запуску.

Перевага віддана rule based підходу замість моделей машинного навчання, оскільки в реальних кейсах доступ до маркованих датасетів обмежений, а прозорість правил є критичною для судової практики.

Методи згруповано навколо чотирьох осей перевірки, а саме: цілісність вмісту, узгодженість метаданих у межах одного файлу, узгодженість між шарами системи та відповідність подієвій хронології.

Перша вісь це перевірка байтової цілісності та ідентичності об'єктів. Для кожної одиниці зберігання будуються контрольні суми та фіксуються у маніфестах, після чого повторний підрахунок на копії для аналізу засвідчує незмінність даних під час роботи. За потреби застосовується зіставлення об'єктів у різних джерелах, наприклад у резервній копії і у поточному стані, із виявленням розбіжностей на рівні хешів. Цей метод не вирішує логічні конфлікти часу, проте є базовим запобіжником і створює опорну лінію для

подальших висновків, а у випадках очікуваної ідентичності сигналізує про підміну чи втрату даних.

Друга вісь це внутрішня перевірка метаданих медіафайлів. З EXIF видобуваються поля, що описують момент фіксації сцени, геоприв'язку, модель пристрою та інші технічні параметри. Ці значення звіряються з файловими атрибутами, а також з контентними характеристиками самого файлу. Виявляється неможливе поєднання даних, наприклад зміна часу без відповідної зміни міток файлової системи, конфлікт між датою у EXIF та фактом появи у каталозі мультимедіа, вбудована мініатюра що не відповідає поточному зображенню.

Для мініатюр, а також для швидкої перевірки схожості версій використовуються перцептивні хеші, які дозволяють уловити зміни вмісту, навіть якщо розмір та формат збігаються. Така комбінація дає змогу виявляти не лише редагування тегів, а й приховане редагування самого контенту.

Третя вісь це структурно часовий аналіз прикладних баз даних. Локальні SQLite сховища досліджуються на предмет узгодженості схем, послідовностей ключів і монотонності часових колонок.

Оцінюється наявність транзакційних журналів, відповідність індексів, коректність типів і одиниць часу, синхронність взаємопов'язаних таблиць, наприклад ниток і повідомлень у месенджерах. Будь які неузгодженості трактуються через набір атомарних тестів, які разом утворюють індикатор структурної розбіжності.

Перевага такого підходу у тому, що він не залежить від конкретної програми і переноситься між різними джерелами, оскільки оперує спільними для SQLite властивостями, зокрема автоінкрементами, зовнішніми ключами і журналюванням.

Четверта вісь це кореляція між шарами системи та реконструкція хронології. Після нормалізації часу до єдиної шкали будуються очікувані

причинні зв'язки, наприклад зйомка передує індексації, редагування зсуває мітку модифікації, поява даних у застосунку не може передувати його встановленню. Для кожної пари подій визначається допустиме вікно різниці, що відсікає штатні лаги на кшталт індексації чи формування кешів. Конфлікти, які не пояснюються такими лагами, позначаються як аномалії і переносяться в індикаторну модель.

Такий підхід зменшує кількість хибнопозитивних висновків і водночас зберігає чутливість до сценаріїв послідовної підміни.

Оцінка достовірності формалізована у вигляді скорингової функції, що агрегує результати індикаторів.

Кожен індикатор має вагу, яка відображає його доказову силу в конкретному контексті. Підсумковий бал інтерпретується у трьох класах, зокрема автентичний, сумнівний і підозрілий, причому для останніх двох класів система формує пояснення у вигляді карти розбіжностей і посилань на вихідні витяги.

Така форма презентації результату відповідає вимогам відтворюваності та робить рішення придатним для подальшого ручного аналізу або судової експертизи.

Усі проміжні результати фіксуються у логах і маніфестах, що гарантує відтворюваність і можливість незалежної перевірки.

Комбінація зазначених методів створює цілісну процедуру первинної верифікації, яка закриває розрив між простим вилученням даних і обґрунтованим висновком про їхню автентичність.

3.3 Формування системи індикаторів

Надалі формалізуємо систему індикаторів, які відображають ознаки можливого втручання у мобільні артефакти.

Кожен індикатор описується послідовно:

1. Призначення й контекст застосування, перелік вхідних даних і припущень;
2. Правило розрахунку з поясненням логіки, типові допуски та відомі винятки;
3. Формат доказових матеріалів, що потрапляють до звіту.

Щоб забезпечити коректну кореляцію між різними джерелами (EXIF, файлові атрибути, MediaStore, SQLite, резервні копії), усі часові значення попередньо зведено до єдиної шкали та уніфіковано за одиницями виміру.

Значення індикаторів інтерпретуються на відрізок від нуля до одиниці, де нуль відповідає відсутності ознак аномалії, а одиниця — сильному спрацюванню.

Якщо для конкретного артефакта певний канал спостереження відсутній, індикатор не обчислюється, а далі в агрегуванні враховується через маску доступності.

Формалізація базових припущень:

$$t^{UTC} = t^{лок} - \text{offset}_{TZ}(t) - \text{offset}_{DST}(t), \quad I_k \in [0, 1], \quad m_k = \begin{cases} 1, & \text{індикатор доступний} \\ 0, & \text{індикатор недоступний} \end{cases} \quad (3.1)$$

У кожному підпункті нижче спочатку подано текстове обґрунтування суті індикатора, після чого наведено відповідну формулу з поясненням, що саме обчислюється, на яких даних і для чого це робиться у межах загальної методики.

3.3.1 Індикатор EXIF_FS_INCONSISTENT

Індикатор EXIF_FS_INCONSISTENT фіксує хронологічні невідповідності між трьома незалежними джерелами часу для медіаоб'єкта: моментом фіксації сцени, зафіксованим у EXIF, файловими атрибутами модифікації та додавання, а також записом індексації в медіакаталозі системи (MediaStore).

Його призначення — виявляти ситуації, коли зміна часових міток у метаданих або у файловій системі не підтримується узгодженою подієвою логікою: зйомка має передувати появі в індексі мультимедіа, а подальше редагування має відображатися в атрибутах модифікації.

Вхідними даними є нормалізовані до UTC часові мітки DateTimeOriginal з EXIF, date_added з MediaStore та mtime із файлової системи. За наявності використовується також birthtime/ctime, але вони не є обов'язковими.

Передбачається, що часові значення приведені до єдиних одиниць виміру (секунди) та скориговані на часовий пояс і перехід на літній час.

Допуски для різниці між EXIF і MediaStore інтерпретуються як технологічний лаг індексації, а для різниці між EXIF і файловим mtime — як легітимний ефект редагувань чи системних операцій копіювання.

Винятками, які можуть давати великі, але добросовісні відхилення, є пакетний імпорт із зовнішніх носіїв, масова переіндексація після оновлення системи, відновлення з резервної копії або міграція до нового пристрою. У цих випадках рішення приймається з урахуванням додаткового контексту, а індикатор супроводжується відповідною позначкою у звіті.

На виході формується числове значення в інтервалі від нуля до одиниці разом із доказовими матеріалами: сирі часові мітки, обчислені різниці, застосовані допуски та коротке пояснення причин спрацювання.

Формально індикатор обчислюється за двома базовими різницями часу, де всі величини в UTC:

$$\Delta t_1 = |t_{\text{add}} - t_{\text{EXIF}}|, \quad \Delta t_2 = |t_{\text{mtime}} - t_{\text{EXIF}}| \quad (3.2)$$

t_{EXIF} — час зйомки з EXIF DateTimeOriginal,

t_{add} — час появи в MediaStore (date_added),

t_{mtime} — файловий час модифікації.

Нормоване значення індикатора задається як максимум відносних перевищень двох різниць над їхніми допусками, з обрізанням до відрізка $[0,1]$:

$$I_{\text{EXIF_FS}} = \min \left(1, \max \left(\frac{\Delta t_1}{\tau_{\text{add}}}, \frac{\Delta t_2}{\tau_{\text{mod}}} \right) \right) \quad (3.3)$$

τ_{add} — допуск на індексацію (типово хвилини),

τ_{mod} — допуск на відображення редагування в mtime (типово від кількох до десятків хвилин залежно від політики).

Обчислення виконується у модулі кореляції часу після етапу нормалізації часових міток до UTC та уніфікації одиниць виміру; у звіт додаються початкові значення t_{EXIF} , t_{add} , t_{mtime} , отримані Δt_1 , Δt_2 , використані τ_{add} , τ_{mod} , а також зазначення виявлених виняткових обставин (імпорт, відновлення, переіндексація), якщо вони вплинули на інтерпретацію результату.

3.3.2 Індикатор THUMBNAIL_MISMATCH

Індикатор THUMBNAIL_MISMATCH виявляє невідповідність між кешованою мініатюрою (thumbnail), яку система або застосунок згенерували раніше, та поточним вмістом оригінального зображення/кадра відео.

Призначення індикатора — фіксувати сценарії, коли контент був змінений після створення мініатюри, або коли в кеші залишилась застаріла мініатюра, яка не відповідає актуальному файлу.

Вхідні дані:

1. Оригінальний медіафайл (JPEG/HEIC/PNG/MP4),
2. Відповідна мініатюра з кешу/MediaStore/додатка,
3. Нормалізовані часові мітки їх створення/оновлення (за наявності).

Передумови:

1. Хеші обох файлів по байтовій цілісності вже перевірені;
2. Зображення зведені до уніфікованих розмірів і кольорового простору для коректного порівняння перцептивних ознак.
3. Для відео використовується єдиний репрезентативний кадр або узгоджений метод вибірки кадрів (наприклад, перший I-кадр).

Типові легітимні винятки:

1. Масштабування і перекодування, які супроводжуються регенерацією thumbnail,
2. Зміна EXIF-тегів без зміни піксельного вмісту (у такому разі очікується збіг перцептивних ознак),
3. Пересинхронізація кешів після оновлення застосунку.

На виході індикатор повертає нормовану оцінку у $[0;1]$ та надає доказові матеріали:

1. Приклад парного відображення «thumbnail $\leftrightarrow$ оригінал»,
2. Значення перцептивної відстані,
3. Використаний метод/параметри порівняння,
4. Часовий контекст (коли згенерована мініатюра відносно модифікації оригіналу).

$$I_{\text{THUMB}} = \min \left(1, \frac{d}{\tau_{\text{pHash}}} \right), \quad (3.4)$$

де τ_{pHash} — робочий поріг чутливості (для 64-бітного pHash зазвичай 10–20).

За наявності часових міток t_{th} (створення/оновлення thumbnail) і t_{mtime} (модифікації оригіналу) додатково оцінюємо причинність:

$$\Delta t_{\text{th}} = t_{\text{th}}^{\text{UTC}} - t_{\text{mtime}}^{\text{UTC}}, \quad (3.5)$$

і вважаємо розбіжність більш релевантною, якщо $\Delta t_{\text{th}} > 0$ (мініатюра новіша за оригінал, але суттєво відрізняється) або якщо $\Delta t_{\text{th}} < 0$ (мініатюра давніша, але не була регенерована після редагування контенту).

Обчислення виконується у модулі порівняння візуального вмісту після уніфікації розміру/формату та перевірки хешів цілісності.

У звіті наводяться скрін-пара, значення d , вибраний τ_{pHash} , а також коментар щодо часової послідовності подій, якщо вона вплинула на трактування результату.

3.3.3 Індикатор SQL_ROW_DIFF

Індикатор SQL_ROW_DIFF призначений для фіксації структурно-часових аномалій у локальних базах даних SQLite, що використовуються мобільними застосунками для журналів викликів, повідомлень, медіакаталогів і кешів.

Його завдання — виявляти неприродні зміни у схемі, послідовностях і часових рядах, які не узгоджуються з типовою транзакційною поведінкою рушія SQLite і логікою роботи застосунку.

Вхідними даними слугують дампи БД у режимі лише читання разом із журналами -wal/-shm (за наявності), опис схеми та індексів (PRAGMA), а також нормалізовані до UTC опорні часові точки, зокрема час інсталяції або оновлення застосунку, створення каталогу даних тощо.

Передумови такі — перевірено байтову цілісність файлів БД, часові поля зведено до єдиних одиниць (секунди), визначено базові інваріанти для відповідного класу артефактів (наприклад, повідомлення не може передувати створенню відповідного чату).

Логіка індикатора ґрунтується на «атомарних тестах» — мінімальних незалежних перевірок, кожна з яких виявляє конкретний тип відхилення і повертає бінарне або нормоване значення у відрізок [0;1].

До базового переліку входять:

1. Монотонність автоінкрементних ідентифікаторів (спрацьовує, якщо трапляється $id_{k+1} < id_k$ у часовому або ключовому порядку),
2. Цілісність зовнішніх ключів (частка рядків, для яких FK посилається на відсутній батьківський запис),
3. Єдність одиниць часу (виявлення змішування секунд і мілісекунд у тій самій колонці),

4. Часові інверсії відносно моменту інсталяції чи створення каталогу даних (наявність рядків із $t_row < t_install$),
5. Транзакційність масових змін (ознаки великих пакетних оновлень без відповідного журналу WAL/SHM у відповідному часовому вікні),
6. Консистентність індексів і унікальних обмежень (виявлення розбіжностей між індексом і базовою таблицею або дублів унікальних ключів).

За потреби до цього переліку додається тест синхронності споріднених таблиць, який контролює причинну послідовність операцій, наприклад додавання message після створення thread і коректне каскадне видалення.

Результат кожного атомарного тесту позначається як $t_i \in [0;1]$ і агрегується у підсумкове значення індикатора з використанням ваг $\alpha_i \geq 0$, $\sum \alpha_i = 1$, що відображають доказову силу відповідної невідповідності для конкретного класу даних. Формально індикатор визначається як:

$$I_{SQL} = \min(1, \sum \alpha_i t_i) \quad (3.6)$$

Вміст його «пояснювального пакета» у звіті включає перелік спрацювань із прикладами рядків/ключів, витяги з PRAGMA для схеми та індексів, хронометку інсталяції/оновлення застосунку, посилання на наявність або відсутність WAL/SHM.

Початкові ваги задаються експертно (зазвичай вищі для порушень монотонності, FK-цілісності та транзакційності), після чого калібруються на пілотних наборах, щоб досягти бажаного балансу чутливості та специфічності.

Інтерпретація SQL_ROW_DIFF завжди виконується з урахуванням легітимних винятків:

1. Офіційні міграції схеми (зміна schema_version, додавання колонок/індексів),
2. Операції VACUUM/REINDEX після оновлень,
3. Відновлення з резервної копії, що можуть викликати тимчасові розсинхронізації або «стрибки» ідентифікаторів.

Такі ситуації маркуються у звіті, а внесок відповідних тестів або знижується, або переноситься до розряду «попереджувальних».

Обчислення індикатора виконується у конвеєрі після етапу вилучення БД та журналів (із фіксацією хешів і часу доступу), далі — у модулі атомарних перевірок, і завершально — в індикаторному модулі, який формує значення I_{SQL} і додає всі доказові матеріали.

У підсумку SQL_ROW_DIFF надає відтворювану, пояснювану і переносну між застосунками міру структурної неузгодженості, що є важливим незалежним сигналом у сукупній оцінці автентичності мобільних артефактів.

3.3.4 Індикатор BACKUP_MISMATCH

Індикатор BACKUP_MISMATCH призначений для виявлення розбіжностей між станом артефактів, зафіксованим у резервній копії, та їхнім поточним представленням у файловій системі й індексах пристрою.

Його завдання — фіксувати підміну, вибіркове видалення або неконсистентне відновлення даних, коли маніфест бекапу декларує один склад і контрольні суми, а «жива» система демонструє інший.

Вхідними даними слугують маніфест резервної копії з переліком файлів, їх розмірами, часовими мітками та контрольними сумами, а також інвентаризація актуального стану файлової системи і, за можливості, записів медіаіндексу.

Передбачається, що обчислення виконується над порівнюваними станами одного й того самого джерела, синхронізованими в часі або прив'язаними до конкретного знімка. Усі часові мітки нормалізовані до UTC, а контрольні суми перераховано без змін контенту.

Типові легітимні винятки пов'язані з інкрементальними політиками резервування, коли маніфест відображає лише дельти відносно попереднього повного знімка, а також із селективними копіями, де частина директорій свідомо не включається. У цих випадках розбіжності інтерпретуються з урахуванням політики бекапу та дати знімка.

На виході індикатор повертає нормоване значення у відрізок від нуля до одиниці разом із доказовими матеріалами: переліком невідповідних елементів, прикладами змінених хешів, відсутніх чи «зайвих» файлів, а також часовим контекстом знімка, що дозволяє відтворити порівняння.

Формально порівнюються дві множини: опис із маніфесту резервної копії та актуальний інвентар, для яких окремо обчислюється частка невідповідних хешів і частка відсутніх чи надлишкових елементів.

Нехай $\#\{hash_diff\}$ — кількість позицій, де $H_{backup} \neq H_{live}$ за умови очікуваної ідентичності, а $\#\{перевіраних\}$ — число елементів, для яких хеш доступний з обох боків, тоді:

$$p_1 = \frac{\#\{hash_diff\}}{\#\{перевіраних у backup\}} \quad (3.7)$$

Аналогічно, нехай $\#\{missing \text{ або } extra\}$ — кількість елементів, відсутніх у одному зі станів без пояснюваних причин політики, а $\#\{очікуваних\}$ — розмір опорної множини для знімка; тоді:

$$p_2 = \frac{\#\{missing \text{ or } extra\}}{\#\{очікуваних\}} \quad (3.8)$$

Нормоване значення індикатора визначається як:

$$I_{BACKUP} = \min(1, \max(p_1, p_2)) \quad (3.9)$$

Обчислення виконується у модулі зіставлення списків після валідації таймстемпа знімка та політики резервування.

У звіт додаються приклади розбіжностей із зазначенням шляхів, розмірів, обох хешів, джерела очікуваного значення та коментар про те, чи могла політика інкрементального або селективного бекапу легітимно пояснити виявлену різницю.

3.3.5 Індикатор HASH_MISMATCH

Індикатор HASH_MISMATCH використовується для контролю байтової ідентичності артефактів між «очікуваним» та «фактичним» станами і є базовим запобіжником у ланцюгу збереження доказів.

Його призначення — зафіксувати будь-яку зміну вмісту файла або об'єкта даних на етапах вилучення, копіювання, аналізу чи відновлення з резервних копій.

На відміну від індикаторів, що оцінюють логічну узгодженість часових міток або структуру сховищ, HASH_MISMATCH дає бінарну відповідь щодо тотожності байтових послідовностей і тим самим визначає еталонну «нульову лінію» для подальшої інтерпретації інших сигналів.

Вхідними даними є контрольні суми (наприклад, SHA-256) об'єкта, обчислені у двох реперних точках: у джерелі (або зафіксовані у маніфесті резервної копії, або отримані безпосередньо під час первинного вилучення) та у робочій копії, з якою працює аналітик.

Передумови включають незмінність режиму доступу під час обчислення (тільки читання), ідентичність області даних, що хешується (весь файл, без пропусків і «дір»), а також фіксацію метаданих обчислення — часу, версії алгоритму та розміру об'єкта.

Типові легітимні винятки стосуються випадків, коли «очікуваний» хеш належить іншому стану того самого артефакта (наприклад, іншому інкрементальному знімку) або коли на копії виконувалась офіційна процедура нормалізації формату із задокументованим перерахунком контрольних сум.

У таких сценаріях співставлення здійснюється між коректними парними станами, а сам індикатор супроводжується приміткою про контекст порівняння.

Обчислення проводиться на етапах:

1. Створення вихідного маніфесту (під час вилучення або читання з backup),
2. Верифікація робочої копії перед аналізом,
3. Підсумкова перевірка після завершення всіх операцій.

Це дозволяє виключити як технічні артефакти випадкових пошкоджень, так і навмисні модифікації під час роботи.

Формально значення індикатора визначається порівнянням криптографічних хешів:

$$H(o) = SHA - 256(o), \quad I_{HASH} = 1\{H_{exp} \neq H_{act}\}, \quad (3.10)$$

де H_{exp} — «очікуваний» хеш із маніфесту або первинного вилучення,

H_{act} — «фактичний» хеш поточного екземпляра об'єкта o .

У звіт виносяться обидва значення хешів, розмір і шлях до файла, часові позначки обчислення, джерело «очікуваного» значення та, за потреби, прив'язка до конкретного знімка резервної копії.

У разі невідповідності індикатор має пріоритетну доказову силу й блокує довіру до похідних висновків щодо такого артефакта до моменту усунення причин розбіжності або повторного відтворення вилучення.

3.3.6 Індикатор TIMELINE_CONFLICT

Індикатор TIMELINE_CONFLICT призначений для виявлення неможливого або неприйняттого порядку подій між різними шарами системи після нормалізації часу до єдиної шкали.

Його завдання — фіксувати хронологічні суперечності, які не можуть бути пояснені технологічними лагами (індексацією, кешуванням, міграцією), зокрема ситуації, коли артефакт «з'являється» до інсталяції застосунку, запис у MediaStore передує моменту зйомки за EXIF, або час модифікації файлу не відображає виконане редагування. Вхідними даними є подієві пари ($a \rightarrow b$) із семантикою причинності (наприклад, «зйомка $\rightarrow$ індексація», «редагування $\rightarrow$ mtime», «інсталяція $\rightarrow$ перший запис у БД», «відновлення $\rightarrow$ поява у каталозі»), нормалізовані часові мітки t_a^{UTC} , t_b^{UTC} та політика допусків $\tau_{a \rightarrow b}$, що відсікає легітимні затримки. Для кожної пари обчислюється перевищення допуску $\epsilon_{a \rightarrow b} = (t_b - t_a) - \tau_{a \rightarrow b}$.

Конфлікт фіксується, якщо зафіксовано інверсію $t_b < t_a$ або $\epsilon_{a \rightarrow b} > 0$ без підтвердженого пояснення (наприклад, масова переіндексація після оновлення системи чи офіційне відновлення з резервної копії). Індикатор повертає бінарне значення у відрізьку $[0;1]$ із пріоритетною доказовою силою, оскільки порушує базову причинність.

Обчислення виконується у модулі кореляції подій після уніфікації часових міток та застосування політики допусків для кожної пари. У звіт виноситься карта розбіжностей із переліком конфліктних пар, їхніми значеннями t_a , t_b , допусками $\tau_{a \rightarrow b}$, обчисленими $\epsilon_{a \rightarrow b}$ і посиланнями на джерела часу (EXIF, MediaStore, атрибути ФС, журнали SQLite). За наявності контекстних артефактів, що пояснюють відхилення (оновлення ПЗ, міграція, відновлення), вони додаються як примітки, а індикатор маркується відповідною ремаркою для коректної інтерпретації в сукупному скорингу.

3.4 Rule-based модель оцінювання

Rule-based скоринг агрегує результати окремих індикаторів у єдину числову оцінку ризику модифікації артефакта та відносить його до одного з класів достовірності.

Модель побудована так, щоб бути відтворюваною, пояснюваною й придатною для процесуального використання. Усі кроки детерміновані, параметри (ваги, пороги, допуски) зафіксовані у конфігурації, а внесок кожного індикатора в підсумок прозоро відображається у звіті.

На вхід скорингу подається вектор нормованих значень індикаторів $I_k \in [0;1]$ та відповідні маски доступності $m_k \in \{0,1\}$, що відсікають відсутні сигнали. Базове агрегування виконується зваженим середнім по доступних індикаторах:

$$\text{score}_0 = \frac{\sum_k w_k m_k I_k}{\sum_k w_k m_k}, \quad w_k \geq 0, \sum_k w_k = 1, \quad (3.11)$$

де w_k — початкові ваги індикаторів, які відображають їхню доказову силу в цільовому контексті (тип джерела, клас артефакта, сценарій перевірки). Усі I_k уже містять внутрішню нормалізацію та обрізання до $[0;1]$, тож «0» означає відсутність ознаки аномалії, «1» — її сильний прояв.

Окремо враховується індикатор порушення причинності часу I_{TIME} (TIMELINE_CONFLICT), що має підвищену доказову силу. Він застосовується як штраф/підсилювач до базового бала:

$$\text{score} = \min(1, \text{score}_0 + \lambda \cdot I_{TIME}), \quad (3.12)$$

де $\lambda \in [0,1]$ — ваговий коефіцієнт впливу часової неузгодженості. Якщо часових конфліктів немає ($I_{TIME}=0$), підсумок дорівнює score_0 . Якщо конфлікт зафіксовано ($I_{TIME}=1$), бал підвищується на λ із жорстким обрізанням до 1.

Така конструкція відділяє «логічно неможливі» сценарії від просто підозрілих.

Класифікація здійснюється за двома порогами:

1. Authentic якщо $\text{score} < T_1$;
2. Questionable якщо $T_1 \leq \text{score} < T_2$;
3. Suspicious якщо $\text{score} \geq T_2$.

Пороги $T_1 < T_2$ задають інтервали впевненості та відокремлюють зону «сумнівних» випадків, які вимагають ручної верифікації. Політика прийняття рішень для організації роботи така:

1. «authentic» — дозволено передавати в подальший аналіз без пріоритету;
2. «questionable» — обов'язкова ручна перевірка із залученням профільних артефактів;
3. «suspicious» — формальний прапорець інциденту з негайним розгляданням та збереженням середовища.

Щоб забезпечити пояснюваність, система додатково обчислює внесок кожного індикатора у фінальний бал:

$$\text{contrib}_k = \frac{w_k m_k I_k}{\sum_j w_j m_j} \Rightarrow \sum_k \text{contrib}_k = \text{score}_0 \quad (3.13)$$

У звіті відображаються «топ-внески» (наприклад, ті, що сукупно дають $\geq 80 \% \text{score}_0$) з прямими посиланнями на первинні витяги:

EXIF/FS/MediaStore-часи, скрін-пара «thumbnail $\leftrightarrow$ оригінал», приклади рядків SQLite, розбіжності з backup-маніфестом.

Таким чином, експерт одразу бачить не лише числовий результат, а й чому система зробила висновок.

Невизначеність, пов'язану з відсутніми даними, модель обробляє через маскування. Якщо деякі канали недоступні ($m_k = 0$), знаменник масштабує ваги лише на доступні сигнали.

За потреби у звіті додається попередження про знижену інформативність. Для підвищення стійкості до одиничних шумових спрацьовувань допускається м'яке згладжування. Індикатори, позначені як «низька надійність» для конкретного класу артефактів, можуть отримувати зменшені ваги або вимагати кумулятивного підтвердження (правило «at least two» у межах групи споріднених індикаторів).

Висновки до розділу 3

У розділі сформовано методику перевірки автентичності мобільних даних на базі rule-based підходу.

Виділено чотири осі контролю:

1. Цілісність вмісту,
2. Внутрішня узгодженість метаданих,
3. Міжшарова кореляція (EXIF—файлова система—MediaStore—SQLite—backup)
4. Відповідність подієвій хронології.

На цій основі визначено систему індикаторів (EXIF_FS_INCONSISTENT, THUMBNAIL_MISMATCH, SQL_ROW_DIFF, BACKUP_MISMATCH, HASH_MISMATCH, TIMELINE_CONFLICT) з чітким описом вхідних даних, правил обчислення та типових допусків і винятків.

Усі часові мітки нормалізуються до UTC; відсутні сигнали обробляються маскою доступності, що зберігає коректність агрегування.

Агрегування індикаторів здійснюється зваженим скорингом із фіксованими вагами та порогами класифікації.

Часові конфлікти (TIMELINE_CONFLICT) враховуються як окремий підсилювальний фактор.

Підхід забезпечує пояснюваність за рахунок відображення внеску окремих індикаторів у підсумковий бал і формування карти виявлених розбіжностей. Отримана методична база є завершеною для переходу до програмної реалізації.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОТОТИПУ МОВОЮ PYTHON

На основі розробленої методики було створено повнофункціональний Python-прототип системи перевірки автентичності цифрових даних мобільних пристроїв.

Прототип складається з логічно розділених модулів, що відповідають етапам аналізу: ingestion, extractors, indicators, scoring та reporting.

Усі модулі реалізовано в межах одного Python-файлу, що спрощує інтеграцію – дані передаються між ними через внутрішні структури (словники та об'єкти), без необхідності проміжного збереження на диск.

Прототип працює з реальними даними файлової системи, охоплюючи різні типи цифрових артефактів: медіафайли (зображення, відео) з EXIF-метаданими, кеші мініатюр (наприклад, файли .thumbdata чи директорії з ескізами), бази даних SQLite (журнали дзвінків, SMS, чати месенджерів, медіакаталоги тощо), системні журнали (.log, JSON, XML-файли) та файли маніфестів резервних копій.

Прототип на вході очікує директорію з файлами, вилученими з мобільного пристрою. Це може бути образ файлової системи, розгорнутий в окрему папку, або структурована вибірка. Користувач може вказати шлях до каталогу для аналізу – у коді передбачено можливість передати його як аргумент командного рядка або налаштувати безпосередньо у виклику методу ingest_directory().

Основні налаштовувані параметри зосереджені у словнику TOLERANCES та змінних модуля scoring. Зокрема, в TOLERANCES задаються допустимі пороги для часових розбіжностей – наприклад, exif_vs_index (τ для затримки індексації фото, кілька хвилин), exif_vs_mtime (τ для розбіжності часу файлу і зйомки, десятки хвилин) тощо.

Також тут міститься поріг для порівняння зображень `phash_threshold` (кількість біт відмінності, понад яку вважаємо зображення невідповідним, за замовчуванням 15).

У модулі `scoring` можна змінити вагові коефіцієнти індикаторів (`INDICATOR_WEIGHTS`) – за замовчуванням всі $=1.0$, але користувач може підвищити вагу критичних показників, як-от `HASH_MISMATCH`, якщо хоче зробити їх вплив сильнішим.

Пороги класифікації `T1`, `T2` теж налаштовуються для чутливості системи (їх можна відрегулювати під конкретні вимоги або статистику помилкових спрацювань).

Далі у розділі детально описано архітектуру прототипу – призначення кожного модуля, ключові аспекти реалізації та взаємодію між модулями.

4.1 Модуль `ingestion`

Модуль `ingestion` відповідає за рекурсивне сканування заданого каталогу (образу файлової системи або папки з даними, вилученими з пристрою) та виявлення всіх підтримуваних артефактів для аналізу.

Він імпортує вихідні дані з мобільного пристрою в систему, збираючи базову інформацію про кожен файл:

1. тип артефакта,
2. шлях,
3. розмір,
4. контрольну суму (хеш)
5. часові атрибути файлової системи.

Ingestion працює з такими основними типами даних:

1. Зображення та відео, форматі JPEG, PNG, HEIC, MP4, MOV тощо). Для них фіксуються EXIF-метадані (у подальшому їх обробить EXIF-екстрактор) – зокрема дата і час зйомки, модель пристрою, координати GPS тощо. З JPEG-файлів також видобувається вбудована мініатюра (thumbnail) для подальшого порівняння з основним зображенням.
2. SQLite-бази даних, наприклад, база повідомлень SMS/MMS mmssms.db, історія чатів месенджера Chats.db, база контактів тощо. Модуль ingestion відкриває такі файли в режимі тільки-читання (через стандартний модуль sqlite3) і витягує базові відомості – наприклад, список таблиць та певні дані, що знадобляться для індикаторів (часові мітки записів, кількість записів, первинні ключі). Ці дані надалі дозволять виявити прогалини в нумерації або хронології записів (для індикатора цілісності бази даних).
3. Файли кешу мініатюр, що містять ескізи зображень (наприклад, .thumbdata на Android). Вони відмічаються для подальшої обробки. Прототип підтримує їх виявлення і може надалі використати для порівняння з оригіналами (через ThumbnailExtractor).
4. Системні журнали та лог-файли з розширеннями .log, .json, .xml та подібні, що містять події системи або застосунків. Ingestion збирає їх як артефати типу "log" для подальшого парсингу (LogExtractor) – це джерело додаткових часових подій і контексту (наприклад, журнали системних подій, установки додатків тощо).
5. Маніфести резервних копій, наприклад, JSON/XML зі списками файлів резервної копії, їх хешами, часами та іншими атрибутами. Вони позначаються як тип "backup_manifest" і будуть використані для контролю узгодженості стану файлової системи з даними резервної копії.

Прототип використовує Python-функції `os.walk` для обходу директорії і визначає тип файлу за його розширенням та вмістом імені (евристично).

Для кожного виявленого підтримуваного файлу створюється об'єкт артефакта з полем шлях, тип, словник метаданих (`meta`) та порожні контейнери для результатів екстракції і індикаторів.

Базові метадані заповнюються одразу – розмір файлу (`st_size`), часові атрибути файлової системи – час останньої модифікації (`st_mtime`) і час створення (`st_ctime`, якщо доступно), – після чого обчислюється криптографічний хеш SHA-256 файлу для контролю цілісності.

Хешування реалізовано з використанням модуля `hashlib` (читання файлу блоками по 4096 байт з оновленням обчислення).

Таким чином, на виході модуля `ingestion` формується список артефактів зі зібраними сирими даними. Цей список передається далі в модуль `extractors` для детальнішого вилучення характеристик.

На етапі `ingestion` відбувається саме збирання даних. Мінімальна обробка – лише те, що необхідно для ідентифікації та цілісності. Більш глибокий аналіз, як-от парсинг вмісту, здійснюватиметься наступними модулями.

4.2 Модуль **extractors**

Модуль **extractors** отримує на вході сирі дані від **ingestion** і виділяє специфічні характеристики кожного артефакта залежно від його типу. Це етап нормалізації даних перед обчисленням індикаторів. Неструктуровані або побітові дані перетворюються у зручні для аналізу структури (словники, списки), фільтрується потрібна інформація. Кожен підтип даних має свій підмодуль-екстрактор:

1. **EXIFExtractor** обробляє медіафайли (зображення/відео), витягуючи EXIF-метадані – вбудовану службову інформацію файлів зображень. Реалізація використовує бібліотеку **Pillow (PIL)** для відкриття файлу і зчитування тегів EXIF.

Відібрані ключові поля включають:

- a. **DateTimeOriginal** (дата й час зйомки),
- b. **Make/Model** (виробник і модель камери або пристрою),
- c. **Software** (програму, якою файл міг бути змінений),
- d. геолокацію (**GPSLatitude**, **GPSLongitude**) та ін.

Особливо важливо нормалізувати час зйомки. Значення

DateTimeOriginal парсується у формат **datetime** і конвертується в **Unix-час** (секунди UTC) – поле **DateTimeOriginal_ts**.

Надалі ці нормалізовані часові мітки слугуватимуть незалежним джерелом часу при порівнянні з файловою системою і іншими джерелами.

Якщо в **JPEG** наявна вбудована мініатюра, екстрактор може її видобути. Поле **Software** також фіксується, якщо, наприклад, там значиться редактор на кшталт "**Adobe Photoshop**", це може бути сигналом редагування зображення.

Відеофайли не мають EXIF, але можуть мати аналогічні метадані (наприклад, дата створення кадру), проте в даному прототипі основна увага – фото.

2. **ThumbnailExtractor** опрацьовує кеші мініатюр та вбудовані ескізи зображень. Його завдання – підготувати пари "мініатюра↔оригінал" для подальшого порівняння.

Реалізовано спрощений підхід. Якщо артефакт є окремим файлом мініатюри (тип "thumbnail_cache") – наразі прототип тільки позначає такі файли, не розбираючи їх повністю (для формату .thumbdata потрібні спеціальні парсери). Якщо ж артефакт – основне зображення/відео, екстрактор здійснює евристичний пошук відповідного thumbnail-файлу серед зібраних артефактів. Зокрема, перевіряється наявність файлів типу "thumbnail_cache" або з назвою, що містить слово "thumbnail" та співпадає частково з іменем оригіналу. Якщо знайдено потенційний ескіз – він завантажується через Pillow (Image.open) і зберігається в пам'яті (в об'єкті артефакта зберігається thumbnail_image та шлях до файлу мініатюри).

Таким чином утворюється пара зображень – основне та його мініатюра, яка далі буде використана індикатором THUMBNAIL_MISMATCH.

3. **SQLiteExtractor** виконує високорівневий аналіз структури та ключових даних SQLite-бази. Екстрактор відкриває файл бази через sqlite3 у режимі read-only (використано URI mode=ro для гарантії незмінності) і отримує схему бази – список таблиць та опис полів (через команду PRAGMA table_info).

У прототипі зберігається структура кожної таблиці (назви колонок, типи, які з них є первинними ключами тощо).

Додатково, вибірково витягуються зразки даних. Екстрактор шукає в таблицях поля, що містять у назві "time" або "date", і вибірково зчитує кілька значень з цих колонок. Це допомагає оцінити хронологію

записів та виявити аномалії форматів часу (наприклад, перемішування секунд і мілісекунд – що перевірятиметься індикатором).

Такі дані надалі використовуються модулем `indicators` для обчислення показника цілісності бази (`SQL_ROW_DIFF`).

4. **LogExtractor** обробляє текстові журнали й лог-файли. Реалізація передбачає спробу структурованого парсингу. Якщо файл має розширення `.json` – екстрактор намагається завантажити JSON-структуру (`json.loads`), якщо `.xml` – парсить XML через `ElementTree`. У випадку успіху отримуються структуровані дані (список словників, дерево XML тощо).

Якщо парсинг за форматом не вдався або файл – простий текстовий лог, екстрактор просто розбиває вміст на строки.

Результат (або структура Python, або список рядків) зберігається як `artifact.extracted['log']`. Таким чином, журнали готові до пошуку часових подій чи інших контекстних даних у наступних модулях. Наприклад, з JSON-логу можна дістати часи певних дій, з системного логу – повідомлення про перезавантаження чи встановлення додатків, які можуть пояснити хронологічні розриви. У контексті прототипу результати `LogExtractor` можуть бути використані для покращення індикатора `TIMELINE_CONFLICT`.

Після проходження всіма екстракторами кожен артефакт отримує структуровані дані, необхідні для перевірок.

Наприклад, для фото буде сформовано словник з полями `datetime_exif`, `device_model`, `software_tag`, `thumb_image_hash` тощо (як показано умовно нижче), а для бази даних – список таблиць, знайдені аномалії тощо. Ці дані передаються в модуль `indicators` для безпосереднього аналізу автентичності.

4.3 Модуль indicators

Модуль indicators приймає структуровані дані кожного артефакта (результати роботи extractors) і обчислює набір індикаторів аномалій, кожен з яких сигналізує про певний аспект потенційної неавтентичності.

Індикатори формалізовано як числові показники у діапазоні $[0,1]$, де 0 означає відсутність відхилень (норма), а 1 – максимальну підозрілість. Кожен індикатор – це окрема функція перевірки, що аналізує одну або кілька характеристик артефакта (або всієї вибірки) на наявність невідповідностей.

Усього в прототипі реалізовано шість індикаторів:

1. **EXIF_FS_INCONSISTENT** порівнює час із метаданих EXIF (час зйомки фотографії) з часовими мітками файлової системи. В ідеалі вони повинні збігатися або відрізнятися незначно (на величину технічної затримки). Значна невідповідність свідчить про можливу маніпуляцію. Наприклад, якщо EXIF-час фото – 5 січня 2025, а файл створено на диску 1 січня 2025 (різниця в –4 дні), це “червоний прапорець” щодо автентичності зображення.
У прототипі обчислюється різниця між EXIF-часом і часом файлу. Якщо різниця перевищує встановлений поріг (τ), індикатор зростає від 0 до 1 пропорційно (лінійно).
Якщо ж зафіксовано логічно неможливу ситуацію – наприклад, файл має мітку модифікації раніше, ніж час зйомки (негативний інтервал), – індикатор виставляється рівним 1 одразу (максимальна підозра). Таким чином, $\text{EXIF_FS_INCONSISTENT} = 1$ означає суттєву розбіжність або неможливий порядок часу EXIF vs файловий час.
2. **HASH_MISMATCH** перевіряє байтову цілісність файлу за допомогою криптографічних хешів. У цифровій криміналістиці порівняння хеш-значень – стандартний метод перевірки тотожності файлів.

Якщо є еталонний хеш (наприклад, з маніфесту резервної копії чи протоколу вилучення), то невідповідність хешів файлу при аналізі означає, що файл змінено або підмінено (індикатор = 1), збіг – що файл ідентичний оригіналу (індикатор = 0).

У нашому прототипі HASH_MISMATCH спрацьовує бінарно для кожного файлу. Якщо еталонного значення немає – цей індикатор для даного файлу не застосовується (вважається 0, оскільки немає основи для порівняння).

3. **SQL_ROW_DIFF** показник цілісності даних у SQLite-базах. Він виявляє структурні або кількісні аномалії в таблицях бази – насамперед відсутні записи чи порушення цілісності зв'язків.

Класичний випадок – пропущені значення первинного ключа (ROWID) в таблиці, де ключ автоінкрементний і має зростати без пропусків.

Якщо, скажімо, останній ID = 120, а реальних записів лише 115, то 5 записів вилучено – індикатор може дорівнювати відношенню $5/120 \approx 0.042$ ($\approx 4.2\%$).

У прототипі SQL_ROW_DIFF реалізовано через низку атомарних перевірок:

- a. монотонність ID – перевіряється, що всі автоінкрементні ключі впорядковані строго зростаюче (немає випадків, де наступний ID менший або рівний попередньому, що могло б вказувати на ручне втручання чи скидання лічильника);

- b. цілісність зовнішніх ключів – для кожного оголошеного зовнішнього ключа перевіряється, чи існує відповідний батьківський запис (виконується LEFT JOIN і рахується кількість "висячих" посилань);

- c. єдність формату часу – для колонок, пов'язаних з часом, перевіряється, чи всі значення в одному масштабі (наприклад, якщо час поданий в одних записах у секундах Unix, а в інших – у

мілісекундах, це спричинить аномально великий розрив між мінімумом і максимумом).

Кожна виявлена проблема збільшує лічильник аномалій, і фінальне значення індикатора є часткою `issues_count / issues_possible` (кількість знайдених проблем на кількість виконаних перевірок).

Значення, близьке до 1, означає суттєві розбіжності або порушення цілісності в БД, тоді як 0 – що структура та послідовність записів виглядають неproblemними. Зауважимо, цей показник не означає "відомо підроблена БД", але вказує на аномалії, які можуть бути як наслідком звичайних операцій (видалення старих записів користувачем), так і приховування даних.

4. **THUMBNAIL_MISMATCH** порівнює вміст основного зображення з його мініатюрою (ескізом). Багато сучасних камер вбудовують у JPEG зменшену копію фото для швидкого попереднього перегляду.

Якщо зображення було відредаговане або підмінене некоректно (наприклад, основне фото замінили, але не оновили вбудований `thumbnail`), або якщо зловмисник намагається видати одне зображення за інше, – мініатюра не відповідатиме оригіналу.

Така невідповідність є відомою ознакою підробки зображень. У прототипі `ThumbnailExtractor` готує пару зображень, після чого індикатор обчислює міру схожості.

Для простоти використано алгоритм `average hash` (усередненого хешу) як наближення до `perceptual hash`. Обидва зображення (оригінал і ескіз) приводяться до єдиного розміру (наприклад, 16×16), конвертуються в градації сірого, і для кожного рахується двійковий шаблон на основі середньої яскравості пікселів. Далі рахується відстань Хеммінга між цими 256-бітними шаблонами – кількість розбіжних біт, `dist`. Якщо зображення ідентичні, `dist = 0`. Якщо вони зовсім різні, `dist` буде високим.

Установлено поріг (наприклад, 15), вище якого вважаємо, що зображення не співпадають (індикатор = 1). Якщо ж dist менше порогу, то обчислюється часткове значення: $\text{індикатор} = \text{dist} / 256$ (частка відмінних пікселів).

Таким чином, `THUMBNAIL_MISMATCH` = 1 означає, що ескіз файлу суттєво відрізняється від його вмісту, що майже напевно вказує на редагування або підміну зображення, а 0 – що мініатюра повністю узгоджується з фото, що очікувано для автентичних знімків.

5. **BACKUP_MISMATCH** призначений для перевірки узгодженості даних пристрою з його резервною копією. Якщо наявний файл-маніфест backup'у (список файлів з контрольними сумами), то порівнюється зміст маніфесту з актуальним станом файлової системи. Індикатор виявляє факти, коли маніфест не співпадає з "живими" даними – наприклад, файл у резервній копії має один хеш, а на пристрої інший (тобто файл було змінено після резервування), або файл, присутній у backup, відсутній на пристрої (його видалено), або навпаки – на пристрої є новий файл, не зафіксований у backup.

У прототипі `BACKUP_MISMATCH` обчислюється як поєднання двох часток – частки файлів з невідповідними хешами та частки файлів, що відсутні/зайві порівняно з маніфестом.

Наприклад, якщо з 100 файлів backup 5 змінені і 5 відсутні, то перша частка 5%, друга 5%, і індикатор може бути близько 0.05 (середнє значень 0.05 і 0.05) або трохи вищим.

Значення 0 означає повну відповідність (усі файли на місці, жоден не модифіковано), 1 – максимальну невідповідність (наприклад, майже всі файли змінені або втрачено).

У звіт цей показник додається для самого маніфест-файлу, а також наводиться список знайдених розбіжностей (імена файлів і тип невідповідності).

Звернемо увагу, що тісно пов'язаний з ним індикатор `HASH_MISMATCH` може застосовуватися для окремих файлів. Якщо `backup` виявив розбіжність хешу для конкретного файлу, цьому файлу можна присвоїти `HASH_MISMATCH = 1`. У реалізації прототипу глобальний показник `backup_mismatch` фіксує такі випадки, але при потребі можна продублювати їх на рівні артефактів.

Обчислення індикаторів виконуються відповідними функціями, які беруть на вхід об'єкт артефакта (або список артефактів, якщо перевірка глобальна) і повертають результат.

Кожна така функція після розрахунку додає результат у структуру `artifact.indicators` у вигляді словника `{'value': ..., 'details': ...}`. Поле `value` – це числове значення $[0..1]$, а `details` – додаткова інформація (наприклад, які саме невідповідності виявлено, перелік проблемних полів тощо).

Для глобального індикатора `TIMELINE_CONFLICT` реалізована окрема функція, що аналізує всі артефакти в цілому.

`TIMELINE_CONFLICT`, на відміну від інших, оцінює узгодженість часової послідовності подій у різних джерелах (файлова система, EXIF, журнали, бази даних). Він покликаний виявити логічно неможливий або неприпустимий порядок подій.

Ситуації, коли подія А повинна передувати В, але згідно часових міток виходить навпаки, або коли затримка між А і В надто велика і не має пояснення.

У прототипі перевіряються такі правила:

1. "зйомка фото → індексація в медіа-каталозі", час додавання файлу в системний медіа-реєстр (`MediaStore`) не повинен передувати часу зйомки і не повинен запізнюватися більше допустимого лагу τ (кількох хвилин);

2. "зйомка $\rightarrow$ поява файлу", час створення файлу на файловій системі (ctime) не може бути раніше EXIF-часу зйомки;

3. "зйомка $\rightarrow$ останнє редагування", час mtime не може передувати часу зйомки. Якщо файл відредаговано, він матиме $mtime \geq t_exif$).

Якщо виявлено будь-який конфлікт (порушення причинно-часової залежності) – встановлюється `TIMELINE_CONFLICT = 1`, і відповідна подія додається до списку деталей конфлікту. Якщо ж усі перевірені залежності дотримані – значення = 0.

Через свою сутність (порушення причинності) цей індикатор має підвищений вплив при оцінюванні, навіть один виявлений конфлікт часу розцінюється як серйозна ознака неавтентичності даних.

4.4 Модуль scoring

Модуль scoring отримує обчислені значення індикаторів для кожного артефакта і агрегує їх у підсумкову оцінку достовірності (score). Також на цьому етапі виконується класифікація кожного артефакта в одну з категорій залежно від рівня виявлених аномалій:

1. Authentic (автентичний),
2. Questionable (сумнівний),
3. Suspicious (підозрілий).

Агрегування виконується як зважене середнє значень індикаторів. Для кожного індикатора задається ваговий коефіцієнт, який відображає вагу ознаки у загальній оцінці. У прототипі за замовчуванням всі ваги прийнято рівними 1.0, тобто кожен з п'яти основних індикаторів впливає рівноцінно. Це зроблено для спрощення і прозорості. В реальних умовах експерт може призначити більшою вагою ті показники, які майже напевно вказують на

підробку (наприклад, THUMBNAIL_MISMATCH чи HASH_MISMATCH), а нижчою – індикатори, що можуть спрацьовувати і при нормальних ситуаціях (наприклад, невеликі EXIF/FS розбіжності).

Нормалізовані значення індикаторів перемножуються на ваги, підсумовуються і діляться на суму всіх ваг – так отримується базовий бал достовірності для артефакта (від 0 до 1).

Далі враховується глобальний часовий конфлікт. Якщо модуль indicators виявив `TIMELINE_CONFLICT = 1` хоча б для однієї пари подій (в цілому по пристрою), то фінальний бал кожного артефакта штрафується додатковим коефіцієнтом λ . У нашому прикладі $\lambda = 0.5$, тобто додається 0.5 до значення аномалій, що еквівалентно підсиленню підозрілості на 50%.

Формально: $\text{final_score} = \min(1.0, \text{base_score} + \lambda * 1.0)$, якщо конфлікт є, і $\text{final_score} = \text{base_score}$ – якщо ні. Таким чином навіть за невеликої базової суми аномалій, але наявного кричущого конфлікту часу, підсумковий бал може суттєво зрости (але не перевищить 1.0).

Насамкінець, отриманий показник порівнюється з двома пороговими значеннями $T1$ і $T2$ для віднесення до категорії:

1. Якщо $\text{score} < T1$, артефакт класифікується як **Authentic** (достовірний) – ознаки маніпуляції відсутні або незначні, дані виглядають автентичними.
2. Якщо $T1 \leq \text{score} < T2$, клас **Questionable** (сумнівний) – виявлено певні підозрілі ознаки, що не однозначно вказують на фальсифікацію, але потребують уваги. Рекомендується детальніша ручна перевірка цих артефактів.
3. Якщо $\text{score} \geq T2$, присвоюється клас **Suspicious** (підозрілий) – дані мають явні ознаки можливого втручання. Такі файли слід розглядати як потенційно підроблені, вживати заходів для їх ізоляції та глибшого аналізу.

У демонстраційній реалізації пороги можна задати конфігураційно, наприклад, використано $T1 = 0.3$ і $T2 = 0.7$ (що інтерпретується як 0–30% аномалій – автентично, понад 70% – підозріло, 30–70% – проміжна зона).

Кожен проаналізований артефакт після модуля `scoring` матиме в своїй структурі підсумкову оцінку (`score`) та визначений клас (`class`), а також (для прозорості) може зберігатися деталізація – які індикатори найбільше вплинули (в прототипі фіксується вклад кожного індикатора). Це забезпечує пояснюваність. Можна побачити, які саме фактори підняли бал.

Наприклад, словник результату може бути: `{'score': 0.49, 'class': 'Questionable', 'indicator_contributions': {'exif_fs_inconsistent': 0.9, 'hash_mismatch': 0.0, ...}}`. Усі ці дані передаються до фінального модулю звітності.

4.5 Модуль `reporting`

Модуль звітності формує підсумковий текстовий звіт про перевірку, придатний для перегляду в консолі або запису у файл.

Звіт містить детальну інформацію про знайдені аномалії для кожного файлу та загальну оцінку автентичності вибірки, у форматі зручному для інтерпретації і включення до додатків дипломної роботи.

Цей модуль виконує роль інтерпретатора результатів. Він бере числові дані, отримані в модулях `indicators` і `scoring`, та представляє їх у читабельному вигляді, доповнюючи поясненнями.

Звіт структуровано по секціях:

1. На початку наводиться короткий підсумок аналізу – скільки артефактів перевірено, скільки з них класифіковано як підозрілі або

сумнівні, скільки автентичні. Це дозволяє одразу оцінити масштаб проблем.

```

≡ report.txt
1  Перевірено артефактів: 4. Підозрілих: 1, сумнівних: 1, автентичних: 2.
2

```

Рисунок 4.1 — Зведена інформація в звіті

2. Далі йде перелік виявлених аномалій по кожному проблемному файлу – коротко, у вигляді пунктів. Кожен пункт містить ім'я артефакта і стислий опис невідповідностей. Такий список дозволяє швидко зорієнтуватися, які саме артефакти мають ознаки маніпуляції і якого роду ці ознаки.

```

3  Виявлені аномалії:
4
5  IMG_0001.JPG: EXIF vs FS time mismatch (EXIF 2025-01-05 10:00:00 vs File 2025-01-01 08:00:00); Thumbnail does not match image content; File hash mismatch.
6
7  IMG_0002.JPG: EXIF vs FS time mismatch (EXIF 2025-07-20 12:00:00 vs File 2025-07-21 04:00:00).
8
9  messages.db: 5 missing message IDs (ROWIDs 9-13) - possible deletions.

```

Рисунок 4.2 — «Карта розбіжностей» в звіті

3. Після карти розбіжностей надається детальніший список, де для кожного файлу наведені значення всіх індикаторів, підсумковий бал та класифікація. Кожен показник супроводжується поясненням (якщо він не нульовий).

```

11  Детальні результати по кожному артефакту:
12  Артефакт: IMG_0001.JPG
13  EXIF_FS_INCONSISTENT: 0.90 (EXIF Date 2025-01-05 10:00:00 vs File Create 2025-01-01 08:00:00)
14  HASH_MISMATCH: 1.00 (SHA-256 не збігається з еталонним)
15  THUMBNAİL_MISMATCH: 1.00 (мініатюра не відповідає зображенню)
16  TIMELINE_MISMATCH: 0.00 (хронологія не порушена)
17  Підсумковий бал: 0.97 → Suspicious
18  Пояснення: Виявлено: розбіжність часових міток, невідповідна мініатюра, невідповідний хеш.
19
20  Артефакт: IMG_0002.JPG
21  EXIF_FS_INCONSISTENT: 0.70 (EXIF Date 2025-07-20 12:00:00 vs File Create 2025-07-21 04:00:00)
22  HASH_MISMATCH: 0.00 (SHA-256 збігається з еталонним)
23  THUMBNAİL_MISMATCH: 0.00 (мініатюра відповідає зображенню)
24  TIMELINE_MISMATCH: 0.00 (хронологія не порушена)
25  Підсумковий бал: 0.35 → Questionable
26  Пояснення: Виявлено: розбіжність часових міток.
27
28  Артефакт: messages.db
29  SQL_ROW_DIFF: 0.04 (відсутні 5 записів, можливе видалення даних)
30  TIMELINE_MISMATCH: 0.00 (хронологія не порушена)
31  Підсумковий бал: 0.04 → Authentic
32
33  Артефакт: system.log
34  TIMELINE_MISMATCH: 0.00 (хронологія не порушена)
35  Підсумковий бал: 0.00 → Authentic
36

```

Рисунок 4.3 — Детальні результати по кожному артефакту

4. Насамкінець генерується узагальнений вердикт щодо всієї сукупності даних. Тут робиться висновок, чи можна вважати дані пристрою автентичними в цілому, чи є підстави сумніватися.

```

36
37  Загальний висновок: Дані в цілому мають підозрілі об'єкти.

```

Рисунок 4.4 — Загальний висновок у звіті

Модуль звітності у прототипі реалізовано у вигляді функції `generate_report(artifacts)`, яка проходить по списку артефактів і формує рядки тексту відповідно до вищезазначеної структури. Відсутність GUI компенсується акуратним форматуванням тексту.

Висновки до розділу 4

У цьому розділі, на основі методики з розділу 3, розроблено повнофункціональний прототип системи перевірки автентичності цифрових даних мобільних пристроїв.

Прототип реалізує всі основні етапи аналізу у вигляді окремих модулів: ingestion, extractors, indicators, scoring та reporting. Впровадження кожного модуля здійснено повністю згідно з описаною методикою, що підтверджує повноту реалізації.

Прототип працює з реальними даними файлової системи, охоплюючи різні типи цифрових артефактів (медіафайли з EXIF-метаданими, кеші мініатюр, бази даних SQLite, системні журнали тощо), тим самим демонструючи придатність запропонованого підходу для практичних сценаріїв. Отримані результати впровадження підтверджують ефективність методики і правильність архітектурних рішень, закладених у попередньому розділі.

Важливо, що розроблений прототип характеризується високою структурованістю, відтворюваністю та прозорістю логіки аналізу. Чітке модульне розділення системи гарантує структурність процесу. Дані послідовно проходять етапи збору, витягування характеристик, обчислення індикаторів, агрегування оцінки та формування звіту.

Така архітектура забезпечує відтворюваність результатів – процедуру перевірки можна повторити з тим самим набором даних і отримати аналогічний результат, що є ключовим для довіри до інструменту.

Логіка прийняття рішень у системі є прозорою. Усі індикатори автентичності формалізовані у вигляді явних правил, а внесок кожного індикатора у підсумковий бал фіксується та відображається у звіті. Це

забезпечує пояснюваність отриманих результатів та можливість незалежної перевірки – кожний висновок системи має зрозуміле обґрунтування.

Результати реалізації підтверджують придатність прототипу до реального застосування і здатність надавати об'єктивний експертний висновок. Прототип успішно застосовано до реальних даних і він коректно виявляє неузгодженості та аномалії у випадку їх наявності.

На основі системи вагових коефіцієнтів індикаторів обчислюється підсумковий score та класифікується весь набір даних як автентичний або сумнівний, з генеруванням відповідного загального вердикту.

Сформований у звіті експертний висновок чітко відображає ступінь достовірності даних пристрою та вказує на виявлені ознаки можливого втручання (за їх присутності). Повнота реалізації всіх передбачених методикою компонентів та досягнуті властивості структурності, відтворюваності й прозорості логіки гарантують, що розроблений інструмент може бути використаний на практиці для надійної перевірки автентичності мобільних даних.

ВИСНОВКИ

У даній роботі досягнуто поставленої мети – розроблення та експериментального обґрунтування методики виявлення ознак несанкціонованого втручання у цифрові дані мобільних пристроїв на базі ОС Android, а також створення прототипу інструменту для автоматизованої попередньої перевірки їх автентичності.

За результатами проведеного дослідження розроблено новий комплексний підхід до підтвердження автентичності мобільних даних, що ґрунтується на багатофакторному індикаторному аналізі із використанням rule-based моделі.

Теоретичний аналіз підтвердив, що забезпечення автентичності цифрових даних мобільних пристроїв є одним із ключових завдань сучасної цифрової криміналістики та кібербезпеки. Встановлено, що мобільні пристрої формують складну багаторівневу екосистему цифрових артефактів (EXIF-метадані, бази даних SQLite, журнали подій, резервні копії тощо) з високою доказовою цінністю, яка водночас залишається вразливою до фальсифікацій.

Виявлено основні загрози достовірності, зокрема маніпуляції метаданими і часовими мітками, редагування системних журналів та баз даних застосунків, підміна резервних копій. Традиційні методи контролю цілісності (хешування, побітове порівняння) не здатні виявляти такі контекстні та логічні невідповідності між артефактами. Огляд міжнародних стандартів (ISO/IEC 27037, NIST SP 800-101, рекомендації ENFSI) та сучасних forensic-інструментів (Cellebrite, Magnet AXIOM тощо) показав, що хоча світова практика наголошує на збереженні цілісності даних і принципу «ланцюга зберігання» (chain of custody), відсутні готові рішення для

автоматизованої перевірки автентичності у динамічному мобільному середовищі. Це підтвердило актуальність розробки нового підходу, який оцінює достовірність не лише на рівні окремих файлів, але й через кореляцію даних з різних джерел.

У другому розділі сформовано прикладну основу для перевірки автентичності мобільних артефактів. Детально досліджено архітектуру зберігання даних Android, визначено ключові зони пам'яті (системні сховища, sandbox-каталоги застосунків, медіатеки) та роль системного індексатора MediaStore як незалежної «третьої точки істини» для часових міток файлів. Розроблено процедури вилучення та нормалізації атрибутів артефактів (EXIF-дані, часові мітки файлової системи, записи SQLite, кеші мініатюр, маніфести backup) у єдину часову шкалу з урахуванням часових поясів та технологічних затримок індексації.

Окреслено типові сценарії фальсифікації (маніпуляції EXIF-метаданими, зсуви системного часу, вибіркове редагування баз SQLite, підміна резервних копій, розбіжності між мініатюрами та оригіналами) та пов'язані з ними ознаки, придатні для автоматизованого виявлення. У результаті підготовлено відтворюваний корпус даних із уніфікованою часовою шкалою та задокументованим контекстом, який дозволяє відрізнити легітимні технологічні лаги від аномалій і слугує вхідною базою для rule-based індикаторного аналізу.

У третьому розділі розроблено методику перевірки автентичності даних на основі rule-based підходу. Виділено чотири основні осі контролю: цілісність вмісту, внутрішня узгодженість метаданих файлів, міжшарова кореляція (узгодженість між EXIF, файловою системою, MediaStore, SQLite та backup) і відповідність загальній хронології подій. На цій основі формалізовано систему цифрових індикаторів автентичності, яка охоплює метрики EXIF_FS_INCONSISTENT, THUMBNAIL_MISMATCH, SQL_ROW_DIFF, BACKUP_MISMATCH, HASH_MISMATCH,

TIMELINE_CONFLICT тощо. Для кожного індикатора визначено набір вхідних даних, умови спрацювання, допустимі винятки та пріоритети джерел часу. Усі часові мітки нормалізуються до UTC, а відсутні дані обробляються маскою доступності сигналів, зберігаючи коректність агрегування результатів.

Підсумкове значення визначається шляхом зваженого скорингу з фіксованими ваговими коефіцієнтами та порогами класифікації; при цьому виявлені конфлікти хронології враховуються як додатковий підсилювальний фактор. Запропонований підхід забезпечує пояснюваність, оскільки внесок кожного індикатора у підсумковий бал фіксується і відображається у звіті, а кожен етап аналізу супроводжується логуванням проміжних результатів. Таким чином, сформована методична база є завершеною та готовою до програмної реалізації.

У четвертому розділі на основі розробленої методики створено повнофункціональний прототип системи перевірки автентичності мовою Python. Прототип складається з низки модулів, що відповідають ключовим етапам аналізу: ingestion (імпорт даних), extractors (витяг атрибутів), indicators (обчислення індикаторів), scoring (агрегування оцінки) та reporting (генерація звіту). Кожен модуль реалізовано згідно з проектною методикою, що підтвердило повноту виконання поставлених вимог.

Прототип працює з реальними даними файлової системи, охоплюючи різні типи цифрових артефактів (медіафайли з EXIF-метаданими, кеші мініатюр, бази даних SQLite — журнали викликів, SMS, чати месенджерів, медіакаталоги тощо, системні логи, маніфести резервних копій). Тим самим продемонстровано придатність запропонованого підходу для практичних сценаріїв. Дані послідовно проходять усі стадії аналізу — від збору до формування кінцевого звіту — в єдиному конвеєрі, що забезпечує структурованість процесу без втрат інформації.

Важливо, що розроблений прототип характеризується високою структурованістю, відтворюваністю та прозорістю логіки аналізу.

Архітектура системи гарантує, що процедуру перевірки можна повторити на тому самому наборі даних і отримати аналогічний результат, що є ключовим для довіри до інструменту.

Логіка прийняття рішень у системі є прозорою — усі індикаторні правила реалізовані явно, а внесок кожного з них у кінцевий результат фіксується та відображається у звіті. Це забезпечує пояснюваність отриманих результатів та можливість незалежної перевірки — кожен висновок системи має зрозуміле обґрунтування.

Результати випробувань прототипу підтверджують його придатність до практичного застосування та здатність надавати об'єктивний експертний висновок. Прототип коректно виявляє неузгодженості та аномалії у випадку їх наявності. На основі системи вагових коефіцієнтів індикаторів обчислюється інтегральний ризиковий бал (score) і класифікується весь набір даних як автентичний або сумнівний з формуванням відповідного загального вердикту. Сформований у звіті технічний висновок чітко відображає ступінь достовірності даних пристрою та вказує на виявлені ознаки можливого втручання.

Отримані наукові та практичні результати мають значне прикладне значення для цифрової криміналістики. Запропонований підхід закриває методичний розрив між вилученням даних і їх доказовою перевіркою, підвищуючи доказову силу результатів та рівень довіри до електронних доказів.

Розроблений прототип може слугувати допоміжним інструментом для попереднього аналізу цифрових матеріалів. Його застосування дозволяє суттєво скоротити час первинної експертизи, швидко виокремити найбільш підозрілі файли та виявити нетипові відхилення параметрів.

Методика легко інтегрується у наявні робочі процеси. Її можна запускати перед глибинним аналізом у комерційних forensic-платформах як «фільтр ризику» для пріоритизації випадків або застосовувати як стандартизовану процедуру в лабораторному пайплайні, що документує перевірку цілісності й автентичності даних.

Методика є універсальною та розширюваною. Нові індикатори і джерела контексту можуть бути додані до системи без зміни її основної логіки, а вагові коефіцієнти і порогові значення можуть калібруватися під конкретні класи пристроїв чи умови експлуатації. Це дозволяє підходу еволюціонувати разом із розвитком цифрової криміналістики та появою нових загроз.

Концепцію можна адаптувати до інших платформ, зокрема iOS, з урахуванням специфіки їх файлових систем і журналів. Подальші дослідження доцільно спрямувати на автоматизацію перевірки автентичності в умовах обмеженої доступності даних або шифрування, а також на залучення методів машинного навчання для вдосконалення скорингової моделі (за наявності достатніх навчальних вибірок). Розвиток цих напрямів дозволить розширити можливості розробленої системи та ще більше підвищити надійність автоматизованої перевірки автентичності цифрових доказів, закладеної в даній роботі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. ISO/IEC 27037:2012 – Information technology — Security techniques — Guidelines for identification, collection, acquisition and preservation of digital evidence [Electronic resource]. – Available at: <https://www.iso.org/standard/44381.html> (Accessed: 25.09.2025).
2. ISO/IEC 27041:2015 – Information technology — Security techniques — Guidance on assuring suitability and adequacy of incident investigative methods [Electronic resource]. – Available at: <https://www.iso.org/standard/44405.html> (Accessed: 25.09.2025).
3. ISO/IEC 27042:2015 – Information technology — Security techniques — Guidelines for the analysis and interpretation of digital evidence [Electronic resource]. – Available at: <https://www.iso.org/standard/44406.html> (Accessed: 28.09.2025).
4. NIST Special Publication 800-101 Rev.1 – Guidelines on Mobile Device Forensics. Gaithersburg, MD: NIST, May 2014 [Electronic resource]. – Available at: <https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-101r1.pdf> (Accessed: 28.09.2025).
5. ENFSI Forensic IT Working Group. Best Practice Manual for the Forensic Examination of Digital Technology (ENFSI-BPM-FIT-01), 1st Edition, 2015 [Electronic resource]. – Available at: https://enfsi.eu/wp-content/uploads/2016/09/1._forensic_examination_of_digital_technology_0.pdf#:~:text=ENFSI,November%202015 (Accessed: 04.11.2025).

6. Association of Chief Police Officers (ACPO). ACPO Good Practice Guide for Digital Evidence, Version 5.0, March 2012 [Electronic resource]. – Available at: <https://athenaforensics.co.uk/wp-content/uploads/2019/01/National-Police-Chiefs-Council-ACPO-Good-Practice-Guide-for-Digital-Evidence-March-2012.pdf> (Accessed: 11.11.2025).
7. Закон України “Про електронну ідентифікацію та електронні довірчі послуги” : Закон України від 05.10.2017 р. № 2155-VIII [Електронний ресурс]. – Режим доступу: <https://zakon.rada.gov.ua/go/2155-19> (дата звернення: 13.11.2025).
8. Закон України “Про захист інформації в інформаційно-телекомунікаційних системах” : Закон України від 05.07.1994 р. № 80/94-ВР [Електронний ресурс]. – Режим доступу: <https://zakon.rada.gov.ua/go/80/94-%D0%B2%D1%80> (дата звернення: 13.11.2025).
9. Khang, A., Dhankhar, S., Bhardwaj, S., Verma, A., Sharma, S. K. Data Recovery Techniques for Computer Forensics. Bentham Science Publishers, 2025. [Book] – Available at: <https://www.benthambooks.com/book/9789815274677/#:~:text=ISBN%3A%20978,2174%2F97898152746771250101> (Accessed: 18.11.2025).
10. Guha, A. Digital Forensics for Enterprises: Beyond Kali Linux: Navigate complex legal frameworks, ensure digital evidence admissibility, and establish robust forensics laboratory environments. BPB Publications, 2025. [Book] – Available at: <https://link.springer.com/book/10.1007/978-3-030-38954-3#:~:text=Fundamentals%20of%20Digital%20Forensics> (Accessed: 19.11.2025).

11. Kävrestad, J. Fundamentals of Digital Forensics: Theory, Methods, and Real-Life Applications. Springer, 2020. [Book] – Available at: <https://link.springer.com/book/10.1007/978-3-030-38954-3#:~:text=Fundamentals%20of%20Digital%20Forensics> (Accessed: 22.11.2025).
12. Tykushin, A., Ostrovskaya, S. Incident Response for Windows. Packt Publishing, 2024. [Book] – Available at: <https://msl.overdrive.com/media/10986508#:~:text=Packt%20Publishing> (Accessed: 23.11.2025).
13. Cellebrite UFED (Universal Forensic Extraction Device) – комплекс апаратно-програмних засобів для вилучення та аналізу даних з мобільних пристроїв [Electronic resource]. – Available at: <https://www.cellebrite.com> (Accessed: 28.11.2025).
14. Magnet Forensics – Magnet AXIOM: digital investigation platform for mobile and computer forensics [Electronic resource]. – Available at: <https://www.magnetforensics.com/products/magnet-axiom/> (Accessed: 28.11.2025).
15. Autopsy Digital Forensics – open-source forensic toolkit (GUI to The Sleuth Kit) [Electronic resource]. – Available at: <https://www.autopsy.com> (Accessed: 28.11.2025).
16. Oxygen Forensic Suite (Detective) – комплексна програма для судово-технічної експертизи мобільних пристроїв [Electronic resource]. – Режим доступу: <https://www.oxygenforensics.com> (дата звернення: 28.11.2025).
17. Scientific Working Group on Digital Evidence (SWGDE). SWGDE Best Practices for Mobile Phone Forensics, Version 2.0, February 2013 [Electronic resource]. – Available at: <https://www.swgde.org/documents/Released%20Documents/SWGDE%20Best%20Practices%20for%20Mobile%20Phone%20Forensics%20V2-0> (Accessed: 03.12.2025).

ДОДАТОК А. КОД МОВОЮ PYTHON ПРОГРАМНОГО ПРОТОТИПУ ОЦІНКИ АВТЕНТИЧНОСТІ МОБІЛЬНИХ ДАНИХ

```

import os, hashlib, datetime
from PIL import Image, ExifTags
import sqlite3
import json
import xml.etree.ElementTree as ET

class Artifact:
    """Клас для представлення артефакта (файлу) і зберігання пов'язаної
    інформації."""
    def __init__(self, path, ftype):
        self.path = path          # шлях до файлу
        self.type = ftype         # тип артефакта (image, video, thumbnail_cache,
        sqlite, log, backup_manifest)
        self.meta = {}           # метадані файлу (розмір, часові атрибути, хеш
        тощо)
        self.extracted = {}       # результати роботи extractors (EXIF дані,
        зображення мініатюри, контент БД/логу)
        self.indicators = {}      # результати обчислених індикаторів для цього
        артефакта

class Ingestion:
    """Модуль ingestion: рекурсивне сканування директорії та збирання
    базових метаданих артефактів."""
    def __init__(self):
        self.artifacts = []

    def ingest_directory(self, root_path):
        """Рекурсивно обходить root_path, ідентифікує підтримувані файли та
        збирає їх базові метадані."""

```

```

for dirpath, dirs, files in os.walk(root_path):
    for filename in files:
        filepath = os.path.join(dirpath, filename)
        # Визначення типу файлу за розширенням:
        fext = filename.lower().split('.')[-1] if '.' in filename else ''
        if fext in ['jpg', 'jpeg', 'png', 'heic', 'mp4', 'mov']:
            ftype = 'image' if fext in ['jpg', 'jpeg', 'png', 'heic'] else 'video'
        elif fext in ['thumbdata', 'cache']:
            ftype = 'thumbnail_cache'
        elif fext in ['db', 'sqlite']:
            ftype = 'sqlite'
        elif fext in ['log', 'json', 'xml']:
            ftype = 'log'
        elif 'manifest' in filename.lower() or fext in ['bak', 'backup']:
            # Файли, що назвою вказують на маніфест резервної копії
            ftype = 'backup_manifest'
        else:
            continue # пропустити файли невідомих типів
        # Створення об'єкта Artifact і збирання базової інформації
        artifact = Artifact(filepath, ftype)
        try:
            stat = os.stat(filepath)
            artifact.meta['size'] = stat.st_size
            # Часові мітки файлової системи (Unix час, сек)
            artifact.meta['mtime'] = stat.st_mtime # час останньої модифікації
            artifact.meta['ctime'] = getattr(stat, 'st_ctime', None) # час створення
            # (Windows) або зміни метаданих (Unix)
        except Exception as e:
            artifact.meta['size'] = None
            artifact.meta['mtime'] = None

```

```

        artifact.meta['ctime'] = None
# Обчислення SHA-256 хеша файлу
try:
    hasher = hashlib.sha256()
    with open(filepath, 'rb') as f:
        for chunk in iter(lambda: f.read(4096), b''):
            hasher.update(chunk)
    artifact.meta['hash'] = hasher.hexdigest()
except Exception as e:
    artifact.meta['hash'] = None
# Додати артефакт до списку
self.artifacts.append(artifact)
return self.artifacts

```

```
class Extractors:
```

```
    """Модуль extractors: містить статичні методи для вилучення специфічної
    інформації з артефактів."""
```

```
    @staticmethod
```

```
    def extract_exif_metadata(artifact):
```

```
        """Витягти EXIF-метадані з файлу зображення або відео (якщо
        доступно)."""
```

```
        filepath = artifact.path
```

```
        data = {}
```

```
        try:
```

```
            img = Image.open(filepath)
```

```
            # Одержати EXIF-теги для зображення
```

```
            if hasattr(img, "_getexif"):
```

```
                exif = img._getexif()
```

```
            else:
```

```
                exif = None
```

```
            if exif:
```

```

    for tag, value in exif.items():
        tag_name = ExifTags.TAGS.get(tag, tag)
        data[tag_name] = value
    # Витягти DateTimeOriginal і конвертувати в timestamp
    if 'DateTimeOriginal' in data:
        dt_str = data['DateTimeOriginal'] # формат "YYYY:MM:DD
        HH:MM:SS"
        try:
            dt_obj = datetime.datetime.strptime(dt_str, "%Y:%m:%d
            %H:%M:%S")
            data['DateTimeOriginal_ts'] = int(dt_obj.timestamp())
        except Exception:
            data['DateTimeOriginal_ts'] = None
    else:
        data['DateTimeOriginal_ts'] = None
    # Вбудована мініатюра (якщо є) – місце для можливої реалізації
    # (Pillow не надає прямого доступу до thumbnail, тому ця частина
    наразі не реалізована)
    except Exception as e:
        data['Error'] = str(e)
    artifact.extracted['exif'] = data

    @staticmethod
    def extract_thumbnail(artifact, all_artifacts):
        """Знайти та завантажити відповідне зображення-мініатюру для заданого
        артефакта (за наявності)."""
        # Якщо сам артефакт є кешем мініатюр:
        if artifact.type == 'thumbnail_cache':
            # Парсинг формату .thumbdata не реалізований – позначити, що такий
            файл є
            artifact.extracted['thumbnails'] = [] # список мініатюр (можна
            розширити при реалізації)

```

```

    return

    # Якщо це медіафайл, спробувати знайти окремий файл-мініатюру
    if artifact.type in ['image', 'video']:
        filename = os.path.basename(artifact.path)

        # Кандидати на роль thumbnail – артефакти типу thumbnail_cache або
        # файли з "thumbnail" у назві
        thumb_candidates = [a for a in all_artifacts if a.type == 'thumbnail_cache'
                             or (a.type == 'image' and 'thumbnail' in a.path.lower())]
        for thumb in thumb_candidates:
            if filename in os.path.basename(thumb.path):
                try:
                    img = Image.open(thumb.path)
                    artifact.extracted['thumbnail_image'] = img.copy()
                    artifact.extracted['thumbnail_path'] = thumb.path
                    break
                except Exception:
                    continue

        # Якщо зовнішню мініатюру не знайдено – можна було б використати
        # вбудовану (не реалізовано)
        return

    @staticmethod
    def extract_sqlite(artifact):
        """Відкрити SQLite-базу даних та витягти її схему і вибірковий
        контент."""
        artifact.extracted['sqlite'] = {}

        try:
            conn = sqlite3.connect(f"file:{artifact.path}?mode=ro", uri=True)
            cur = conn.cursor()

            # Список таблиць
            cur.execute("SELECT name FROM sqlite_master WHERE type='table';")

```

```

tables = [row[0] for row in cur.fetchall()]
artifact.extracted['sqlite']['tables'] = tables
artifact.extracted['sqlite']['schema'] = {}
for table in tables:
    # Отримати інформацію про колонки (схему таблиці)
    cur.execute(f"PRAGMA table_info({table});")
    cols = cur.fetchall()
    artifact.extracted['sqlite']['schema'][table] = cols # список кортежів
    (cid, name, type, notnull, dflt_value, pk)
    # Зразки даних: спробувати отримати по кілька значень часу з таблиць
    artifact.extracted['sqlite']['content_samples'] = {}
    for table in tables:
        # Знайти колонки, що містять 'time' або 'date' у назві
        cur.execute(f"PRAGMA table_info({table});")
        cols_info = cur.fetchall()
        time_cols = [col[1] for col in cols_info if any(sub in col[1].lower() for
sub in ['time', 'date'])]
        if time_cols:
            try:
                cur.execute(f"SELECT {' '.join(time_cols)} FROM {table} LIMIT
5;")
                artifact.extracted['sqlite']['content_samples'][table] = cur.fetchall()
            except Exception:
                artifact.extracted['sqlite']['content_samples'][table] = []
    cur.close()
    conn.close()
except Exception as e:
    artifact.extracted['sqlite']['error'] = str(e)

@staticmethod
def extract_log(artifact):

```

```

"""Розпарсити вміст лог-файлу (JSON, XML або plaintext)."""
content = None
path = artifact.path
try:
    with open(path, 'r', encoding='utf-8', errors='ignore') as f:
        text = f.read()
    if path.lower().endswith('.json'):
        # Парсинг JSON
        try:
            content = json.loads(text)
        except Exception:
            content = text # якщо JSON не строго формалізований, залишити
як текст
    elif path.lower().endswith('.xml'):
        # Парсинг XML
        try:
            root = ET.fromstring(text)
            content = ET.tostring(root, encoding='utf-8').decode('utf-8')
        except Exception:
            content = text
    else:
        # Звичайний текстовий лог – розбити на рядки
        content = text.splitlines()
except Exception as e:
    artifact.extracted['log_error'] = str(e)
artifact.extracted['log'] = content

# Конфігурація: допустимі допуски часу (в секундах) та інші параметри
TOLERANCES = {
    'exif_vs_index': 120, # допустима затримка (2 хв) між зйомкою та
індексацією медіафайлу

```

```

    'exif_vs_mtime': 1800,    # допустима різниця (30 хв) між часом зйомки і
    файловим mtime

    'phash_threshold': 15    # поріг різниці average hash (кількість біт), вище
    якого thumbnail вважається невідповідним
}

# Вагові коефіцієнти для кожного індикатора (можна змінювати відповідно
до вагомості ознаки)
INDICATOR_WEIGHTS = {
    'exif_fs_inconsistent': 1.0,
    'hash_mismatch': 1.0,
    'sql_row_diff': 1.0,
    'thumbnail_mismatch': 1.0,
    'backup_mismatch': 1.0
}

# Додатковий коефіцієнт для часових конфліктів ( $\lambda$ )
LAMBDA_TIMELINE = 0.5

# Пороги для класифікації
T1 = 0.3
T2 = 0.7

def compute_exif_fs_indicator(artifact):
    """Обчислити індикатор EXIF_FS_INCONSISTENT для медіафайлу
    (зображення/відео)."""
    result = {'value': 0.0, 'details': {}}
    if artifact.type not in ['image', 'video']:
        return None # не застосовується
    # Отримати необхідні часи
    t_exif = artifact.extracted.get('exif', {}).get('DateTimeOriginal_ts')
    t_fs = artifact.meta.get('mtime')
    t_fs_create = artifact.meta.get('ctime')
    # Час додавання в медіа-індекс (якщо був отриманий, тут може бути None)

```

```

t_index = artifact.extracted.get('exif', {}).get('MediaStoreAdded_ts')
# Допуски часу з конфігурації
tau_index = TOLERANCES.get('exif_vs_index', 0)
tau_mod = TOLERANCES.get('exif_vs_mtime', 0)
# Розрахунок різниць
dt_index = (t_index - t_exif) if (t_exif and t_index) else None
dt_mod = (t_fs - t_exif) if (t_exif and t_fs) else None
# Перевірка умов невідповідності
inconsistencies = []
if dt_index is not None:
    if dt_index < 0:
        inconsistencies.append(('IndexBeforeCapture', dt_index))
    elif dt_index > tau_index:
        inconsistencies.append(('IndexDelay', dt_index))
if dt_mod is not None:
    if dt_mod < 0:
        inconsistencies.append(('MTimeBeforeCapture', dt_mod))
    elif dt_mod > tau_mod:
        inconsistencies.append(('ModDelay', dt_mod))
# Нормалізація значення (найбільше перевищення допуску)
value = 0.0
for issue, diff in inconsistencies:
    if issue in ('IndexBeforeCapture', 'MTimeBeforeCapture'):
        # Немоżliвий порядок подій -> максимальна підозрілість
        value = 1.0
    elif issue == 'IndexDelay':
        value = max(value, (diff - tau_index) / float(tau_index) if tau_index > 0
else 1.0)
    elif issue == 'ModDelay':
        value = max(value, (diff - tau_mod) / float(tau_mod) if tau_mod > 0 else
1.0)

```

```

if value > 1.0:
    value = 1.0
result['value'] = value
# Детальна інформація для звіту
result['details']['t_exif'] = t_exif
result['details']['t_fs_mtime'] = t_fs
result['details']['t_index'] = t_index
result['details']['issues'] = inconsistencies
artifact.indicators['exif_fs_inconsistent'] = result
return result

def compute_thumbnail_mismatch(artifact):
    """Обчислити індикатор THUMBNAIL_MISMATCH шляхом порівняння
    cached thumbnail vs original image."""
    result = {'value': 0.0, 'details': {}}
    if artifact.type not in ['image', 'video']:
        return None
    thumb_img = artifact.extracted.get('thumbnail_image')
    # Відкрити оригінальне зображення
    try:
        orig_img = Image.open(artifact.path).convert('RGB')
    except Exception as e:
        orig_img = None
    if thumb_img is None or orig_img is None:
        # Нема мініатюри або не вдалося відкрити оригінал – вважаємо, що не
        # можемо порівняти
        artifact.indicators['thumbnail_mismatch'] = result
        return result
    # Обчислити average hash (спрощений perceptual hash) для двох зображень
    def average_hash(img, hash_size=8):
        img = img.convert('L').resize((hash_size, hash_size), Image.ANTIALIAS)

```

```

pixels = list(img.getdata())
avg = sum(pixels) / len(pixels)
bits = ".join('1' if p > avg else '0' for p in pixels)
return bits # повернути бітовий шаблон як рядок

hash_orig = average_hash(orig_img, 16) # 16x16 -> 256-бітний хеш
hash_thumb = average_hash/thumb_img, 16)

# Відстань Хеммінга між двома бінарними хешами
dist = sum(c1 != c2 for c1, c2 in zip(hash_orig, hash_thumb))

# Нормування значення: якщо dist перевищує поріг -> 1.0, інакше
пропорційно
threshold = TOLERANCES.get('phash_threshold', 0)
value = dist / float(len(hash_orig)) # частка відмінних бітів (0..1)
if dist >= threshold:
    value = 1.0
result['value'] = value
result['details']['distance'] = dist
result['details']['threshold'] = threshold
artifact.indicators['thumbnail_mismatch'] = result
return result

```

```

def compute_sql_row_diff(artifact):
    """Обчислити індикатор SQL_ROW_DIFF для артефакта типу sqlite
    (SQLite-бази даних)."""
    result = {'value': 0.0, 'details': {}}
    if artifact.type != 'sqlite':
        return None
    details = []
    issues_count = 0
    issues_possible = 0
    try:
        conn = sqlite3.connect(f"file:{artifact.path}?mode=ro", uri=True)

```

```

cur = conn.cursor()
tables = artifact.extracted.get('sqlite', {}).get('tables', [])
for table in tables:
    # 1. Перевірка монотонності первинного ключа (якщо цілий)
    schema = artifact.extracted['sqlite']['schema'].get(table, [])
    pk_cols = [col for col in schema if col[5] == 1] # стовпці, де pk == 1
    for col in pk_cols:
        col_name = col[1]; col_type = col[2]
        if col_type.upper().startswith("INT"):
            cur.execute(f"SELECT {col_name} FROM {table} ORDER BY
{col_name};")
            ids = [row[0] for row in cur.fetchall()]
            for i in range(1, len(ids)):
                if ids[i] is not None and ids[i-1] is not None and ids[i] <= ids[i-1]:
                    issues_count += 1
                    details.append(f"Non-monotonic ID in {table}: {col_name}
sequence")
                    break # достатньо одного випадку на таблицю
            issues_possible += 1
    # 2. Перевірка зовнішніх ключів (чи є "висячі" посилання)
    cur.execute(f"PRAGMA foreign_key_list({table});")
    fk_list = cur.fetchall()
    for fk in fk_list:
        # fk: (id, seq, table, from, to, on_update, on_delete, match)
        ref_table = fk[2]; from_col = fk[3]; to_col = fk[4]
        try:
            cur.execute(f"SELECT COUNT(*) FROM {table} t LEFT JOIN
{ref_table} r ON t.{from_col}=r.{to_col} WHERE r.{to_col} IS NULL;")
            missing = cur.fetchone()[0]
            if missing > 0:
                issues_count += 1

```

```

        details.append(f"Broken FK: {table}.{from_col} -> missing
{ref_table}.{to_col} ({missing} rows)")
    except Exception:
        pass
    issues_possible += 1
# 3. Узгодженість формату часу (секунди vs мілісекунди)
schema_cols = [col[1] for col in schema]
time_cols = [c for c in schema_cols if any(sub in c.lower() for sub in
['time', 'date'])]
for col_name in time_cols:
    try:
        cur.execute(f"SELECT MIN({col_name}), MAX({col_name})
FROM {table};")
        min_val, max_val = cur.fetchone()
        if min_val is None or max_val is None:
            continue
        # Якщо max значно більший за min (ймовірно, мікс мс і с)
        if min_val > 0 and max_val > 0 and max_val / min_val > 1000:
            issues_count += 1
            details.append(f"Time format inconsistency in {table}.{col_name}
(range {min_val} - {max_val})")
    except Exception:
        continue
    issues_possible += 1
cur.close()
conn.close()
except Exception as e:
    details.append(f"SQLite error: {e}")
# Нормалізація: частка виявлених проблем від кількості перевірок
value = (issues_count / float(issues_possible)) if issues_possible > 0 else 0.0
if value > 1.0:

```

```

    value = 1.0

    result['value'] = value

    result['details']['issues'] = details

    artifact.indicators['sql_row_diff'] = result

    return result


def compute_backup_mismatch(artifact, artifacts):
    """Обчислити індикатор BACKUP_MISMATCH шляхом порівняння даних
    маніфесту резервної копії зі станом файлової системи."""
    result = {'value': 0.0, 'details': {}}

    if artifact.type != 'backup_manifest':
        return None

    # Розбір вмісту маніфесту (припускаємо, що лог-екстрактор міг розпарсити
    # JSON/XML маніфест у artifact.extracted['log'])
    manifest_files = {}

    data = artifact.extracted.get('log')

    if data:
        if isinstance(data, dict):
            # Припущення: якщо маніфест у форматі JSON з ключем 'files'
            if 'files' in data:
                for entry in data['files']:
                    key = entry.get('path') or entry.get('name')
                    if key:
                        manifest_files[key] = {
                            'hash': entry.get('hash'),
                            'size': entry.get('size'),
                            'timestamp': entry.get('timestamp')
                        }

        elif isinstance(data, list):
            # Список об'єктів файлів
            for entry in data:

```

```

key = entry.get('path') or entry.get('name')
if key:
    manifest_files[key] = {
        'hash': entry.get('hash'),
        'size': entry.get('size'),
        'timestamp': entry.get('timestamp')
    }

# Порівняти список файлів маніфесту з фактичними артефактами
mismatched_hashes = []
missing_files = []
extra_files = []

# Словник фактичних файлів: відносний шлях -> {hash, size}
actual_files = {}

# Знайти спільний корінь шляхів (для побудови відносних шляхів)
base_path = os.path.commonprefix([artifacts[0].path, artifact.path]) if artifacts
else ""

for art in artifacts:
    if art.type in ['image', 'video', 'thumbnail_cache', 'sqlite', 'log'] and
    art.meta.get('hash'):
        # Зберегти шлях відносно базового каталогу даних
        rel_path = os.path.relpath(art.path, base_path) if base_path else art.path
        actual_files[rel_path] = {'hash': art.meta['hash'], 'size': art.meta['size']}

# Перевірити файли з маніфесту
for m_path, m_info in manifest_files.items():
    if m_path in actual_files:
        # Файл присутній на пристрої – порівняти хеш
        if m_info.get('hash') and actual_files[m_path]['hash']:
            if m_info['hash'].lower() != actual_files[m_path]['hash'].lower():
                mismatched_hashes.append(m_path)
    else:
        # Файл є в маніфесті, але відсутній зараз

```

```

        missing_files.append(m_path)
# Перевірити файли, що є на пристрої, але не зазначені у маніфесті
for a_path in actual_files.keys():
    if a_path not in manifest_files:
        extra_files.append(a_path)
# Обчислити частки невідповідностей
total_manifest = len(manifest_files)
hash_diff_ratio = len(mismatched_hashes) / float(total_manifest) if
total_manifest > 0 else 0.0
missing_ratio = (len(missing_files) + len(extra_files)) / float(total_manifest) if
total_manifest > 0 else 0.0
# Комбінувати (середнє двох метрик)
value = (hash_diff_ratio + missing_ratio) / 2.0
if value > 1.0:
    value = 1.0
result['value'] = value
result['details']['hash_mismatches'] = mismatched_hashes
result['details']['missing_files'] = missing_files
result['details']['extra_files'] = extra_files
artifact.indicators['backup_mismatch'] = result
return result

def compute_hash_mismatch(artifact, expected_hash=None):
    """Обчислити індикатор HASH_MISMATCH для файлу за заданим
    еталонним хешем."""
    result = {'value': 0.0, 'details': {}}
    if expected_hash is None:
        # Якщо немає з чим порівнювати – не застосовується (0.0)
        artifact.indicators['hash_mismatch'] = result
        return result
    actual_hash = artifact.meta.get('hash')

```

```

if actual_hash is None:
    # Якщо з якихось причин нема фактичного хешу (файл не читався) – 0.0
    artifact.indicators['hash_mismatch'] = result
    return result

# Порівняти еталонний і фактичний хеш
if actual_hash.lower() != expected_hash.lower():
    result['value'] = 1.0
    result['details']['expected'] = expected_hash
    result['details']['actual'] = actual_hash
    artifact.indicators['hash_mismatch'] = result
    return result

```

```
def compute_timeline_conflicts(artifacts):
```

```
    """
```

Обчислити індикатор TIMELINE_CONFLICT (TIMELINE_MISMATCH) на основі перевірки послідовності подій між артефактами.

Повертає глобальний результат: {'value': 0 або 1, 'details': [список описів конфліктів]}.

```
    """
```

```
    conflicts = []
```

```
    # Правила причинно-часових залежностей:
```

```
    # 1. Фото (час EXIF) -> Додавання в медіакаталог (час t_index)
```

```
    # 2. Фото (час EXIF) -> Створення файлу (ctime) та остання модифікація файлу (mtime)
```

```
    # 3. (потенційно) Встановлення додатку -> перший запис у його БД (не реалізовано)
```

```
    for art in artifacts:
```

```
        if art.type in ['image', 'video']:
```

```
            t_exif = art.extracted.get('exif', {}).get('DateTimeOriginal_ts')
```

```
            t_index = art.extracted.get('exif', {}).get('MediaStoreAdded_ts')
```

```
            t_ctime = art.meta.get('ctime')
```

```

t_mtime = art.meta.get('mtime')

# 1. Зйомка -> індексація
if t_exif and t_index:
    tau = TOLERANCES.get('exif_vs_index', 0)
    if t_index < t_exif:
        conflicts.append(f"Media index time ({int(t_index)}) is before photo
capture ({int(t_exif)}) for {os.path.basename(art.path)}")
    elif t_index - t_exif > tau:
        conflicts.append(f"Media index time for {os.path.basename(art.path)}
delayed by {int(t_index - t_exif)}s > {tau}s after capture")

# 2. Зйомка -> створення файлу (ctime)
if t_exif and t_ctime:
    if t_ctime < t_exif:
        conflicts.append(f"File creation time ({int(t_ctime)}) is before capture
({int(t_exif)}) for {os.path.basename(art.path)}")

# 2 (продовження). Зйомка -> модифікація файлу (mtime)
if t_exif and t_mtime:
    if t_mtime < t_exif:
        conflicts.append(f"File modification time ({int(t_mtime)}) is before
capture ({int(t_exif)}) for {os.path.basename(art.path)}")

# Інші можливі перевірки (не реалізовано тут)

# 3. Встановлення додатку -> перший запис у БД або лог (потребує
даних з логів)
timeline_conflict_value = 1.0 if conflicts else 0.0
return {'value': timeline_conflict_value, 'details': conflicts}

def score_artifact(artifact, timeline_conflict_flag):
    """Обчислити зважений підсумковий бал достовірності для артефакта та
визначити його клас (Authentic/Questionable/Suspicious)."""
    # Обчислення зваженого середнього значень індикаторів
    num = 0.0
    den = 0.0

```

```

contributions = {}
for ind, weight in INDICATOR_WEIGHTS.items():
    if ind in artifact.indicators:
        val = artifact.indicators[ind]['value']
        if val is not None:
            num += weight * val
            den += weight
            contributions[ind] = val
base_score = (num/den) if den > 0 else 0.0
# Застосувати надбавку за глобальний конфлікт часу
final_score = base_score
if timeline_conflict_flag:
    final_score = min(1.0, base_score + LAMBDA_TIMELINE * 1.0)
# Визначити категорію
if final_score < T1:
    classification = "Authentic"
elif final_score < T2:
    classification = "Questionable"
else:
    classification = "Suspicious"
result = {
    'score': final_score,
    'class': classification,
    'base_score': base_score,
    'indicator_contributions': contributions
}
artifact.indicators['score'] = result
return result

def generate_report(artifacts):

```

```

"""Згенерувати текстовий звіт на основі показників та оцінок
артефактів."""

lines = []

# 1. Зведена інформація
total = len(artifacts)

suspicious = sum(1 for art in artifacts if art.indicators.get('score', {}).get('class')
== "Suspicious")

questionable = sum(1 for art in artifacts if art.indicators.get('score',
{}).get('class') == "Questionable")

authentic = sum(1 for art in artifacts if art.indicators.get('score', {}).get('class')
== "Authentic")

lines.append(f"Перевірено артефактів: {total}. Підозрілих: {suspicious},
сумнівних: {questionable}, автентичних: {authentic}.")

lines.append("") # порожній рядок

# 2. Карта розбіжностей
lines.append("Виявлені аномалії:")

any_anomaly = False
for art in artifacts:

    # Сформувати короткий опис аномалій для кожного проблемного файлу
    issues = []

    # EXIF vs FS
    exif_res = art.indicators.get('exif_fs_inconsistent')

    if exif_res and exif_res['value'] and exif_res['value'] > 0:
        t_exif = exif_res['details'].get('t_exif')
        t_fs = exif_res['details'].get('t_fs_mtime')

        if t_exif and t_fs:

            # конвертувати часи у читабельний формат (YYYY-MM-DD
            HH:MM:SS)

            dt_exif = datetime.datetime.fromtimestamp(t_exif).strftime("%Y-%m-
%d %H:%M:%S")

            dt_fs = datetime.datetime.fromtimestamp(t_fs).strftime("%Y-%m-%d
%H:%M:%S")

```

```

        issues.append(f"EXIF vs FS time mismatch (EXIF {dt_exif} vs File
{dt_fs}))")
    # Thumbnail mismatch
    th_res = art.indicators.get('thumbnail_mismatch')
    if th_res and th_res['value'] and th_res['value'] > 0:
        issues.append("Thumbnail does not match image content")
    # SQL diff (для БД)
    sql_res = art.indicators.get('sql_row_diff')
    if sql_res and sql_res['value'] and sql_res['value'] > 0:
        # Наприклад, X missing rows або інші деталі
        issues.append("Database integrity issues")
    # Backup mismatch (для маніфесту)
    bk_res = art.indicators.get('backup_mismatch')
    if bk_res and bk_res['value'] and bk_res['value'] > 0:
        issues.append("Backup manifest discrepancy")

    # Якщо є конфлікт часу (глобальний), можна зазначити окремо, але він
    не відноситься до конкретного файлу прямо.

    # Формування рядка
    if issues:
        any_anomaly = True
        file_name = os.path.basename(art.path)
        lines.append(f"- {file_name}: " + "; ".join(issues) + ".")
    if not any_anomaly:
        lines.append("- (Не виявлено невідповідностей)")
    lines.append("")

    # 3. Детальна таблиця показників і оцінок
    lines.append("Детальні результати по кожному артефакту:")
    for art in artifacts:
        file_name = os.path.basename(art.path)
        score_info = art.indicators.get('score', {})
        art_class = score_info.get('class', 'N/A')

```

```

score_val = score_info.get('score', 0.0)
lines.append(f"Артефакт: {file_name}")
# Перелік індикаторів
for ind_name, ind_res in art.indicators.items():
    if ind_name == 'score':
        # не перераховувати сам score у цьому переліку
        continue
    val = ind_res['value']
    # вивести значення з двома знаками після коми
    lines.append(f" {ind_name.upper()}: {val:.2f}")
# Підсумковий бал і клас
lines.append(f" Підсумковий бал: {score_val:.2f} → **{art_class}**")
# Додаткове пояснення (якщо сумнівний або підозрілий)
if art_class in ["Questionable", "Suspicious"]:
    explain = []
    if art.indicators.get('exif_fs_inconsistent', {}).get('value', 0) > 0:
        explain.append("розбіжність часових міток")
    if art.indicators.get('thumbnail_mismatch', {}).get('value', 0) > 0:
        explain.append("невідповідна мініатюра")
    if art.indicators.get('hash_mismatch', {}).get('value', 0) > 0:
        explain.append("невідповідний хеш")
    if art.indicators.get('sql_row_diff', {}).get('value', 0) > 0:
        explain.append("аномалії в базі даних")
    if art.indicators.get('backup_mismatch', {}).get('value', 0) > 0:
        explain.append("розбіжність з резервною копією")
    if explain:
        lines.append(f" *Пояснення:* Виявлено: {' '.join(explain)}.")
    lines.append("") # розділовий порожній рядок між артефактами
# 4. Загальний висновок
if suspicious > 0:

```

```

        conclusion = "підозрілі"
    elif questionable > 0:
        conclusion = "сумнівні"
    else:
        conclusion = "автентичні"

    lines.append(f"***Загальний висновок:** Дані в цілому { 'мають підозрілі об'єкти' if suspicious>0 else ('мають деякі сумнівні моменти' if questionable>0 else 'виглядають автентичними') }.")

    return "\n".join(lines)

# Блок запуску (якщо скрипт виконувати окремо)
if __name__ == "__main__":
    import sys
    dir_path = sys.argv[1] if len(sys.argv) > 1 else "."
    ing = Ingestion()
    artifacts = ing.ingest_directory(dir_path)
    # Застосувати екстрактори до кожного артефакту
    for art in artifacts:
        if art.type in ['image', 'video']:
            Extractors.extract_exif_metadata(art)
            Extractors.extract_thumbnail(art, artifacts)
        elif art.type == 'thumbnail_cache':
            Extractors.extract_thumbnail(art, artifacts)
        elif art.type == 'sqlite':
            Extractors.extract_sqlite(art)
        elif art.type == 'log':
            Extractors.extract_log(art)

    # Обчислити індикатори
    for art in artifacts:
        compute_exif_fs_indicator(art)
        compute_thumbnail_mismatch(art)

```

```

compute_sql_row_diff(art)
if art.type == 'backup_manifest':
    compute_backup_mismatch(art, artifacts)
    # Додатково: проставити hash_mismatch для окремих файлів на основі
маніфесту
    for file_path in art.indicators.get('backup_mismatch',
{}).get('hash_mismatches', []):
        # Знайти відповідний артефакт за шляхом
        for a in artifacts:
            if file_path in a.path:
                compute_hash_mismatch(a, expected_hash=None) # можна було
б передати очікуваний хеш, якщо відомо
                break
# Перевірити глобальні часові конфлікти
timeline_result = compute_timeline_conflicts(artifacts)
timeline_flag = True if timeline_result['value'] == 1.0 else False
# Скоринг кожного артефакта
for art in artifacts:
    score_artifact(art, timeline_flag)
# Вивести звіт
report_text = generate_report(artifacts)
print(report_text)

```