FORENSIC ANALYSIS OF ANDROID APPLICATIONS (IMO & TINDER) USING ANDRILLER, DBBROWSER AND DCODE

1.0 Background Research

1.1 Related Tools or Research

Mobile device forensics has emerged as a fundamental element of contemporary digital investigations because of the widespread use of technologies especially smartphones and messaging applications in both legal and illegal activities. Numerous forensic tools are available to assist with Android extraction and analysis, including:

- **Andriller (Android Forensic Toolkit)**: A well-known forensic suite capable of ADB extraction, decoding of common Android apps, data carving, password pattern decoding, and automated reporting. It is widely referenced in academic and professional mobile forensics research due to its ability to process Android backups (.ab), TAR archives, and emulator extractions.
- **Autopsy / Sleuth Kit**: Offers broader digital forensics capabilities, including mounting dd images, file system analysis, timeline correlation, and keyword searching.
- **MobSF & MOBEXLER**: Mobile security and forensic analysis platforms useful for static/dynamic analysis of APKs and file systems.
- **Androbug Framework**: A mobile application vulnerability and analysis tool used to inspect APK internals, permissions, and security posture.

Academic studies consistently emphasize that communication applications, particularly those featuring encrypted or ephemeral messaging, harbor forensic traces such as message metadata, cached media, account identifiers, timestamps, location data, authentication tokens, and remnants of deleted information. Research indicates that despite messages being end-to-end encrypted, additional files and metadata can still be retrievable on the device

This project builds on past research by applying **Andriller** to a forensic extraction of two widely used communication/social networking applications: **IMO** and **Tinder**.

1.2 Purpose of the Tool / Forensic Examination

Finding out what kinds of digital data can be recovered and examined from IMO and Tinder using Andriller in a controlled Android emulator environment is the main goal of this forensic investigation. The analysis focusses on:

- Extracting **application data directories**, including databases, logs, shared preferences, and cached media.
- Recovering **communication artefacts**, such as message metadata, contact associations, timestamps, and potential remnants of deleted conversations.
- Identifying **user account information**, device identifiers, and app-specific tokens present in the extracted files.
- Analyzing **app usage behaviour**, including login patterns, message interactions, match activities (Tinder), and voice/video communication traces (IMO).
- Evaluating **metadata and file-system artefacts**, which may persist even when app-level encryption protects message content.

Using Andriller allows the examiner to automate extraction, decode supported artefacts, generate HTML reports, and rapidly inspect structured database information. This aligns with forensic goals such as evidence preservation, timeline reconstruction, and correlation of user activities across multiple applications.

1.3 Forensic Scenarios Where the Tool/App Will Be Useful

Scenario 1: Suspected Online Grooming / Harassment Case (IMO)

Investigators might obtain a mobile device from a minor or victim who has communicated via chat or video with an unidentified person on IMO, a widely used app for instant messaging and video calls. Even if the suspect removes messages, the device might still hold:

- Residual chat database entries
- Contact associations
- Timestamps showing when conversations occurred
- Cached media files or call logs
- Metadata revealing the suspect's IMO ID

With Andriller, an investigator can retrieve the IMO data directory, maintain evidence in a forensically valid way, and reconstruct communication trends that either support or challenge allegations

Scenario 2: Infidelity, Stalking, or Identity Fraud Case (Tinder)

Swipes, matches, profile changes, login timestamps, and message exchanges are all recorded by the match-based social networking app Tinder. A forensic examiner may need to gather information in a civil or criminal inquiry (such as cyberstalking, impersonation, extortion, or harassment):

- Match history and metadata
- Profile information
- User IDs and tokens
- Message timestamps (even if the content is encrypted or partially deleted)
- Geolocation hints from logs or cached files

Andriller enables systematic extraction and decoding of Tinder artefacts stored within SQLite databases and app-specific directories, helping investigators correlate online behaviour with real-world events.

2.0 Forensic Plan

2.1 How the Tools Work

Andriller

Andriller is an Android forensic toolkit that automates the extraction, decoding, and reporting of digital evidence from Android devices. It works by leveraging:

- **ADB (Android Debug Bridge)** to communicate with an Android device or emulator
- **Backup extraction** (AB → TAR → file system reconstruction)
- **App-specific decoders** for popular apps
- **SQLite parsing** to read databases
- **Automated HTML report generation** for examiners

Andriller operates in three primary phases:

1. **Device Interaction:**
   Connects to the Android device/emulator through ADB, enabling file pulls, application data access, and querying of system properties.
2. **Extraction Phase:**
   Creates an Android backup (.ab) or extracts directories directly using TAR/AirGap extraction. The tool parses the resulting archive to recover application folders such as:
   - /data/data/com.imo.android.imoim/
   - /data/data/com.tinder/
3. **Decoding & Reporting:**
   Andriller examines extracted databases (e.g., messages.db, cache.db, shared_prefs) and converts them into readable formats. It generates:
   - HTML tables
   - CSV files
   - Parsed JSON artefacts
   - Timeline summaries

This automated workflow reduces manual parsing time and ensures consistency across examinations.

2.2 How the Examination Will Be Conducted

The forensic examination for this project follows a **structured, repeatable process** to ensure evidence integrity and academic reproducibility.

Step 1: Environment Setup

- Use **Android Studio Emulator** to create a test Android device.
- Install IMO and Tinder APKs.
- Confirming device connection using “adb devices” command to confirm the right device is connected to the Andriller.

Step 2: Evidence Preservation

Before analysis, a clean extraction of the device state is performed using Andriller's ADB extraction features.

- Generate an Android backup (.ab file) OR
- Pull the /data/data/ directories using Andriller's file extraction module.

This ensures forensic integrity by capturing the device in its current state without modifying inner artefacts.

Step 3: Data Extraction

Using Andriller:

- Load the .ab or TAR archive into the tool
- Allow Andriller to extract and reconstruct the file system
- Observe generated folders such as:
    - com.imo.android.imoim/ (chat data, logs, prefs, media)
    - com.tinder/ (user metadata, matches, messages, tokens)

Step 4: Database & Artefact Analysis

Using other tools (DB browser, Dcode. etc) to Inspect the results generated by Andriller, including:

- SQLite databases (message logs, contacts, chat metadata, match information)
- Shared preference XML files (authentication tokens, user IDs, timestamps)
- Cached media (images, profile photos, video thumbnails)
- Logs and metadata
- Decoded HTML reports

This phase includes aligning timestamps, recognizing communication trends, and examining extracted artefacts for forensic significance

Step 5: Documentation & Reporting

All findings will be:

- Compiled into a coherent forensic report
- Supported with screenshots, extracted tables, and parsed outputs

2.3 Underlying Implementation / Architecture Needed

To successfully demonstrate the forensic analysis, the following architecture is required:

1. Technical Setup

- **Windows Host Machine** running: Android Studio, ADB tools, IMO/Tinder APKs.

2. Connectivity Architecture

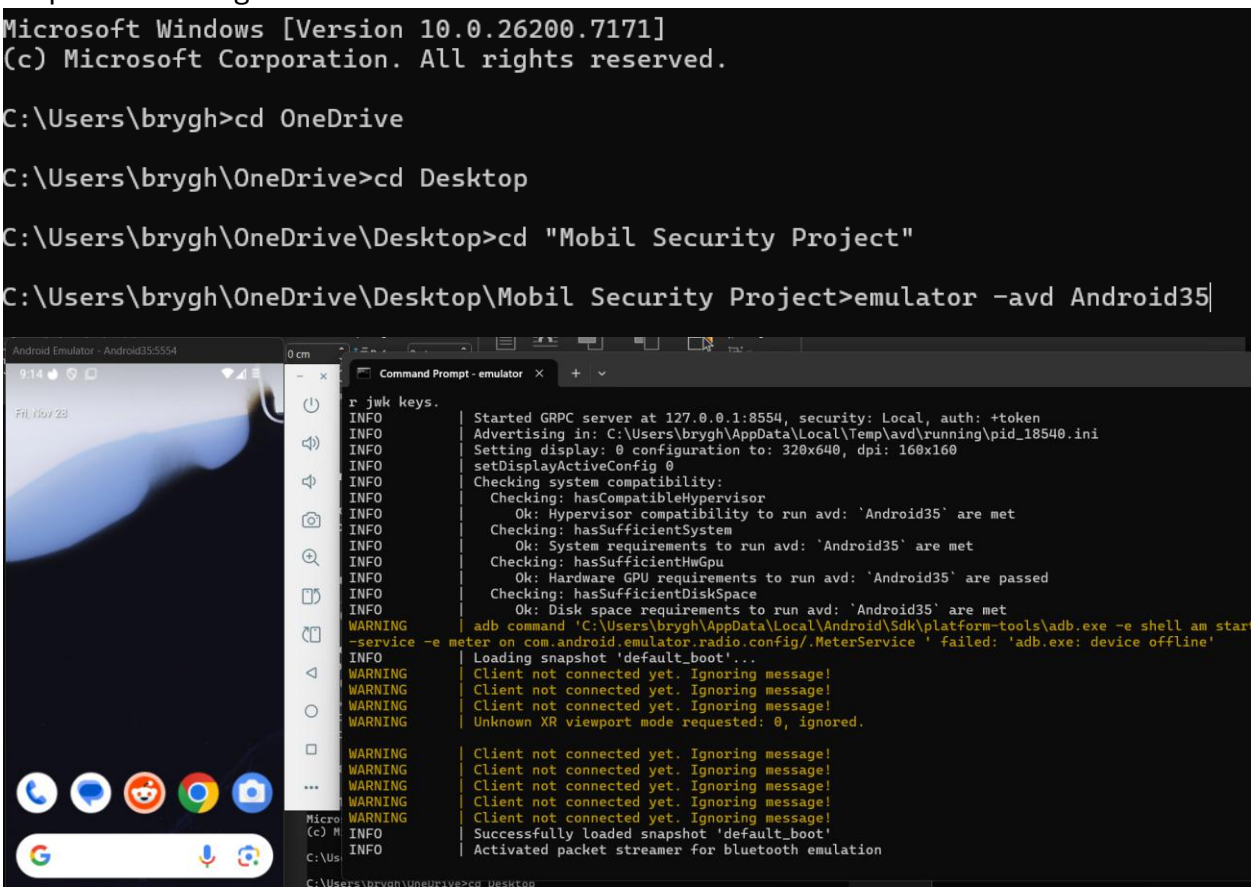
The Android emulator is connected to the forensic workstation (Andriller) via ADB bridge

3.0 Detailed Analysis

This part outlines the complete practical forensic procedure conducted on the IMO and Tinder apps with Andriller, accompanied by screenshots, command results, file-system proof, and insights from both successful and failed extraction efforts. This assessment mimics an actual forensic process and records the difficulties faced during extraction and decoding

3.1 Hands-On Demonstration: Connecting and Preparing the Android Emulator

Step 1: Launching the Android Emulator



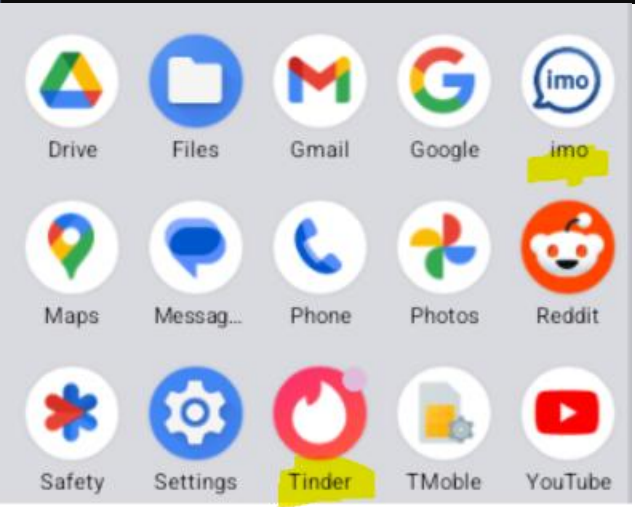
Step 2: Verifying ADB Connection

The connection between the Windows host and the emulator was validated

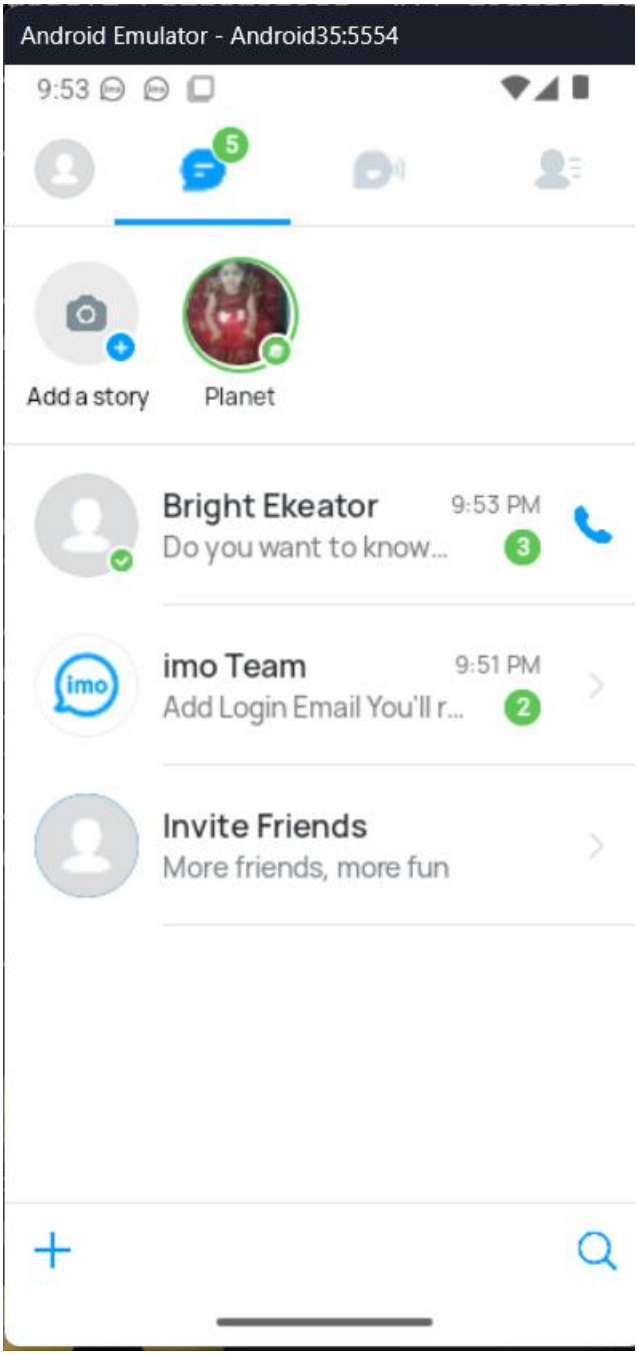
```
C:\Users\brygh\OneDrive\Desktop\Mobil Security Project>adb devices
List of devices attached
emulator-5554   device


C:\Users\brygh\OneDrive\Desktop\Mobil Security Project>
```

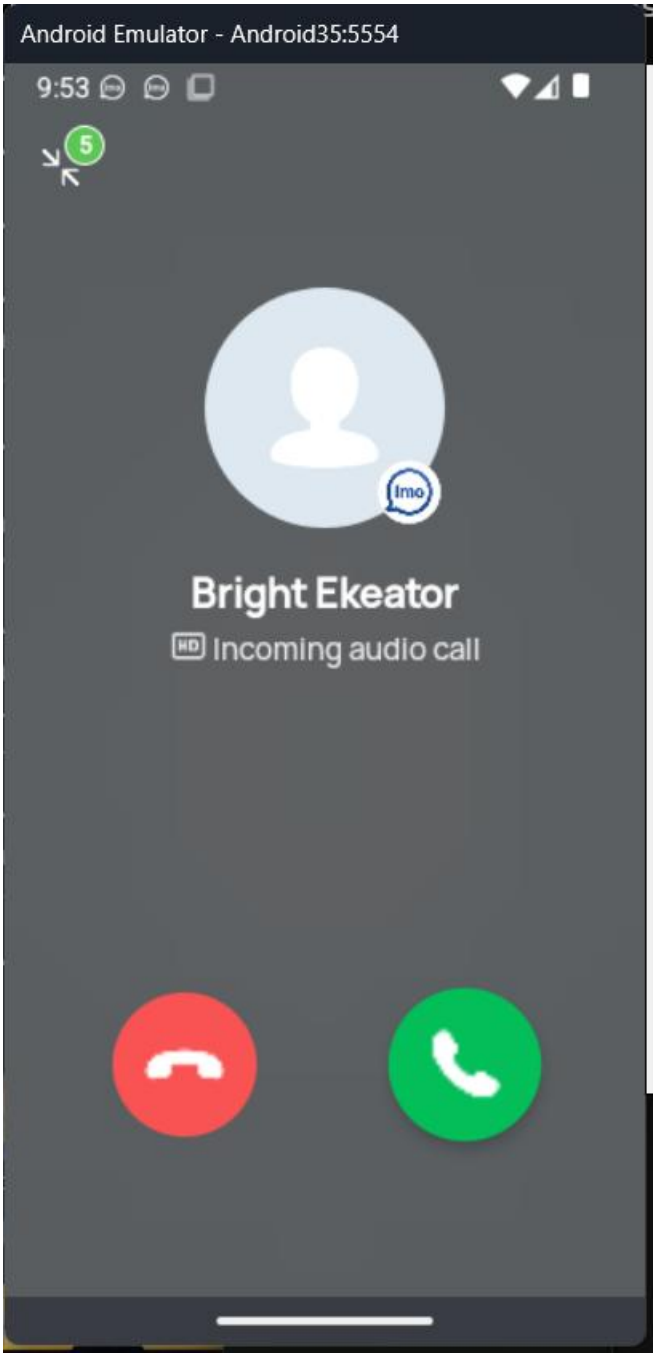
Step 3: Installing the Apps (IMO and Tinder)

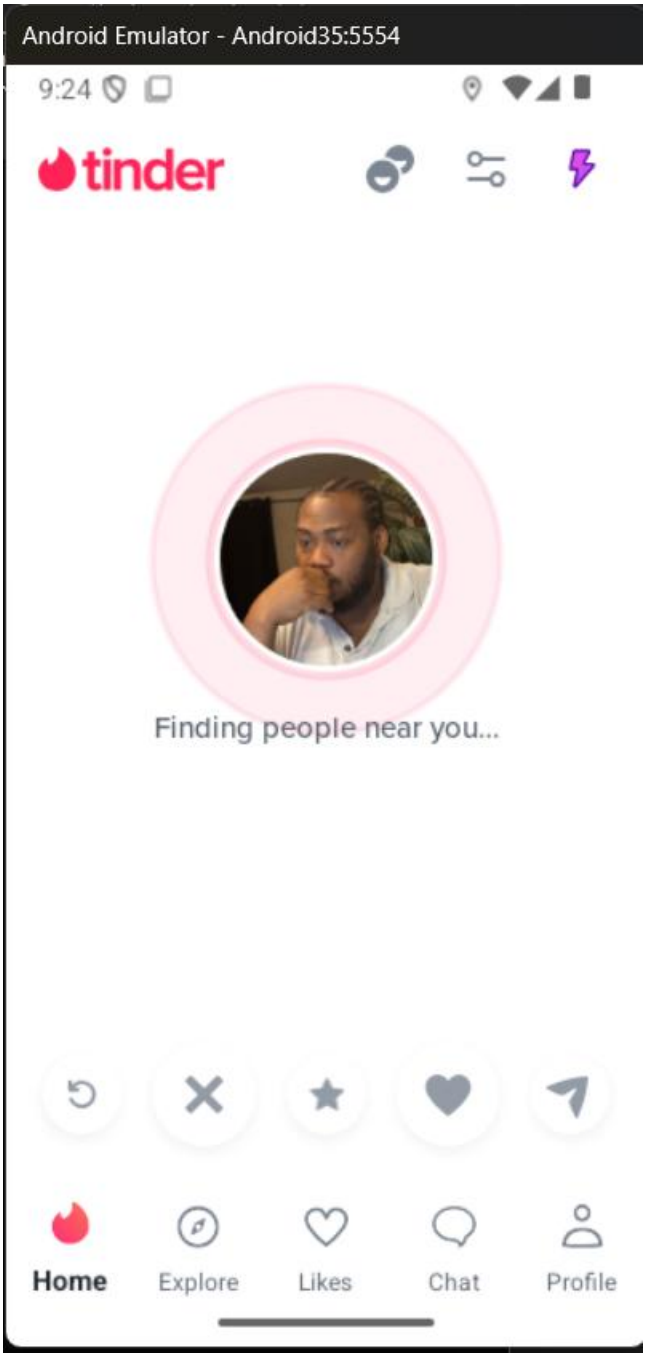
```
C:\Users\brygh\OneDrive\Desktop>cd "Mobil Security Project"

C:\Users\brygh\OneDrive\Desktop\Mobil Security Project>adb install imo-2025-09-1021.apk
Performing Streamed Install
Success
```



3.2 Populating data into the database by using the apps on the emulator

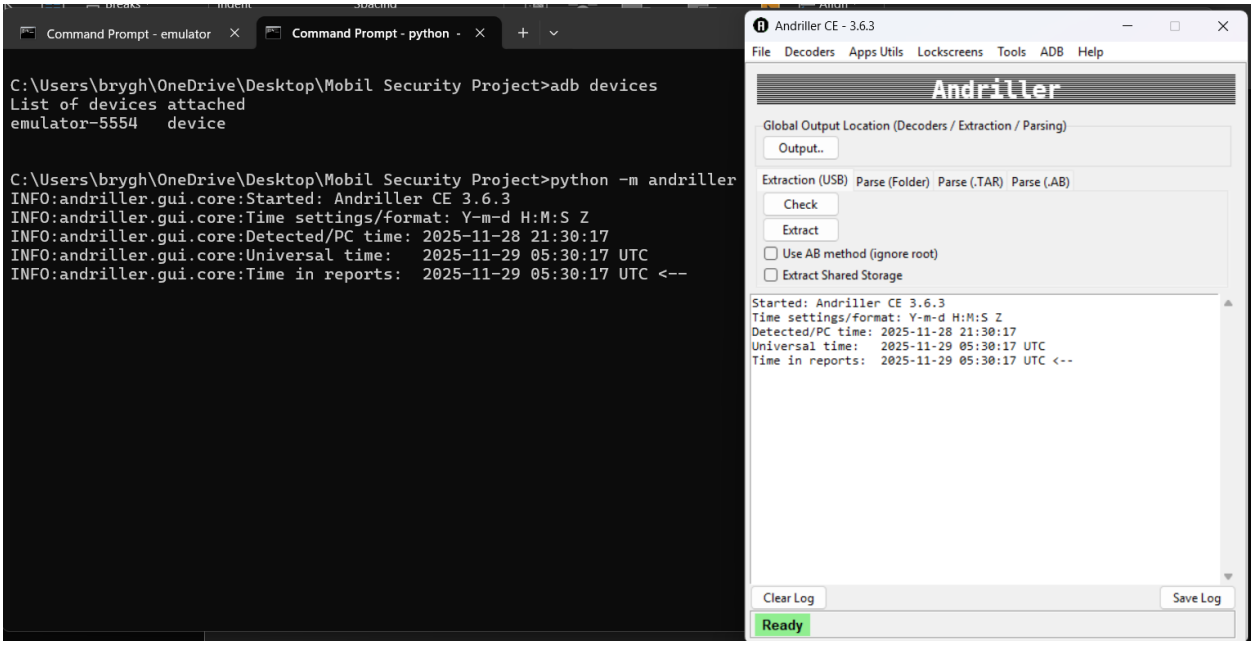

Android Emulator - Android35:5554
9:53
5
Add a story
Planet
Bright Ekeator
9:53 PM
Do you want to know...
3
imo Team
9:51 PM
Add Login Email You'll r...
2
Invite Friends
More friends, more fun


Android Emulator - Android35:5554
9:53
5
Bright Ekeator
Incoming audio call

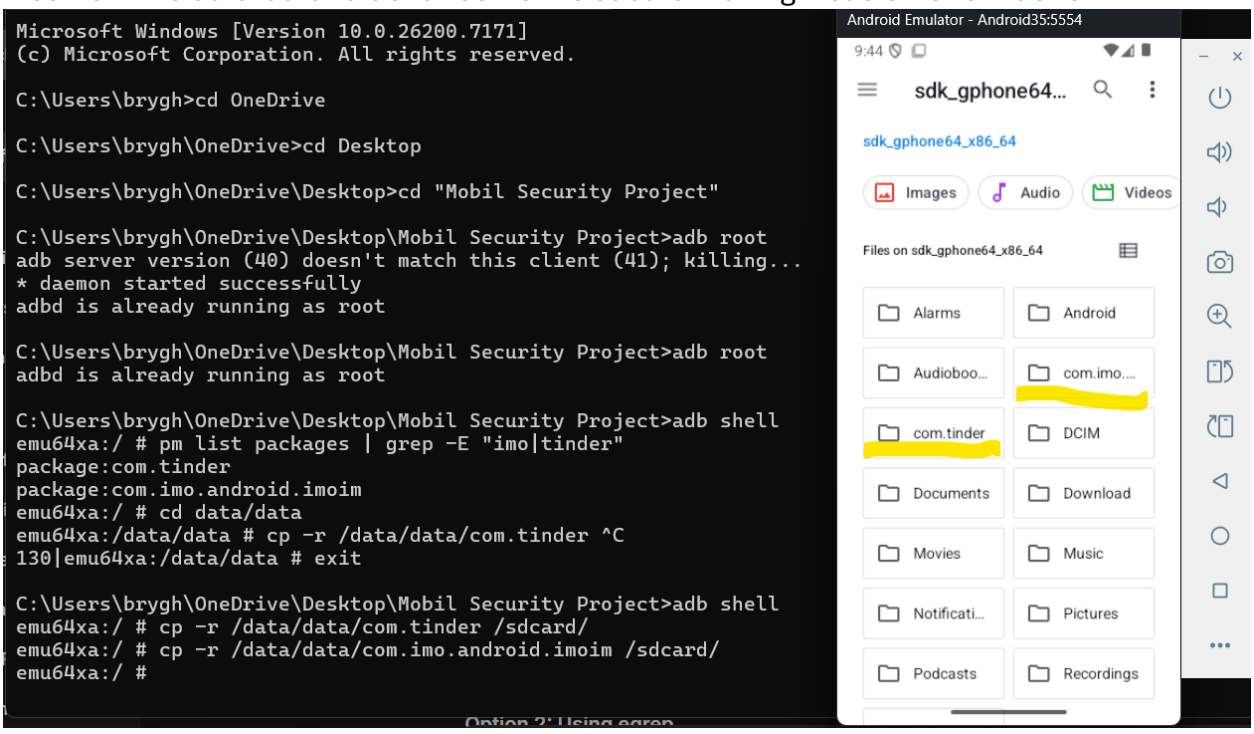


3.3 Extraction Using Andriller

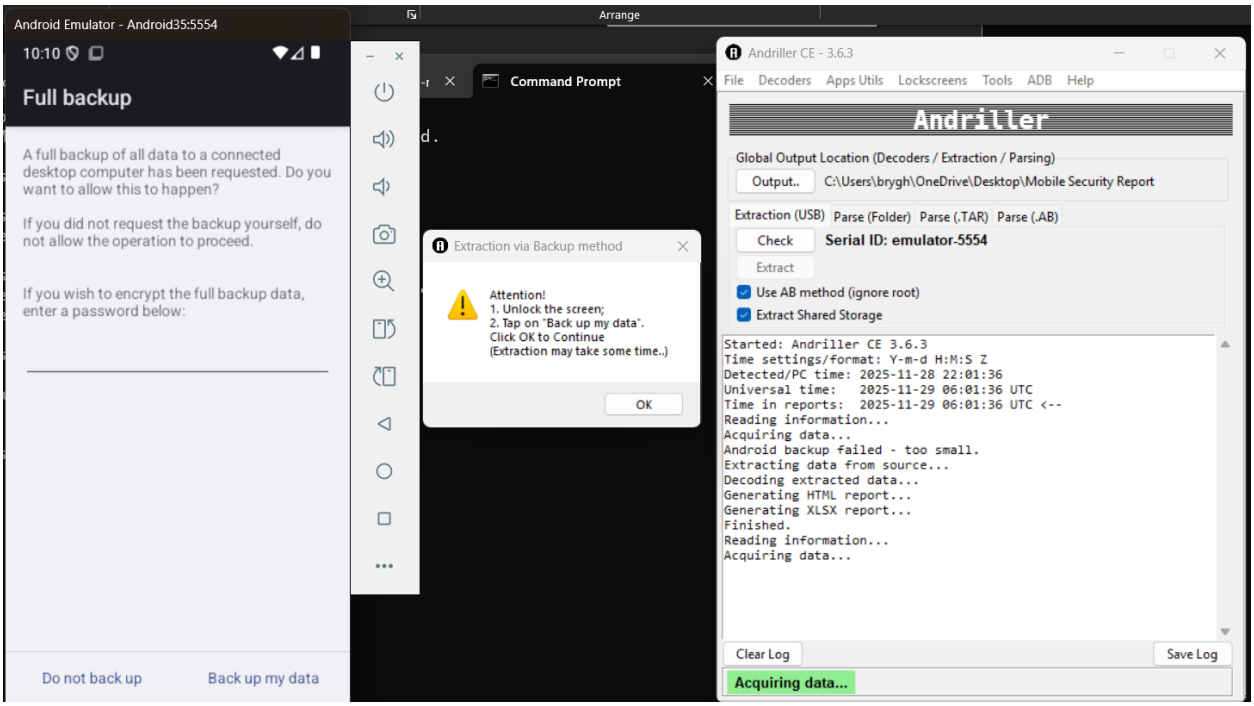
Step 1: Launching Andriller



Step 2: Data Extraction.
Due to the difficulties for Andriller to extract as root, I found a way around it by copying the files from the data/data folder direct to the sdcard making it easier for extraction



Step 2: Package Extraction using Andriller

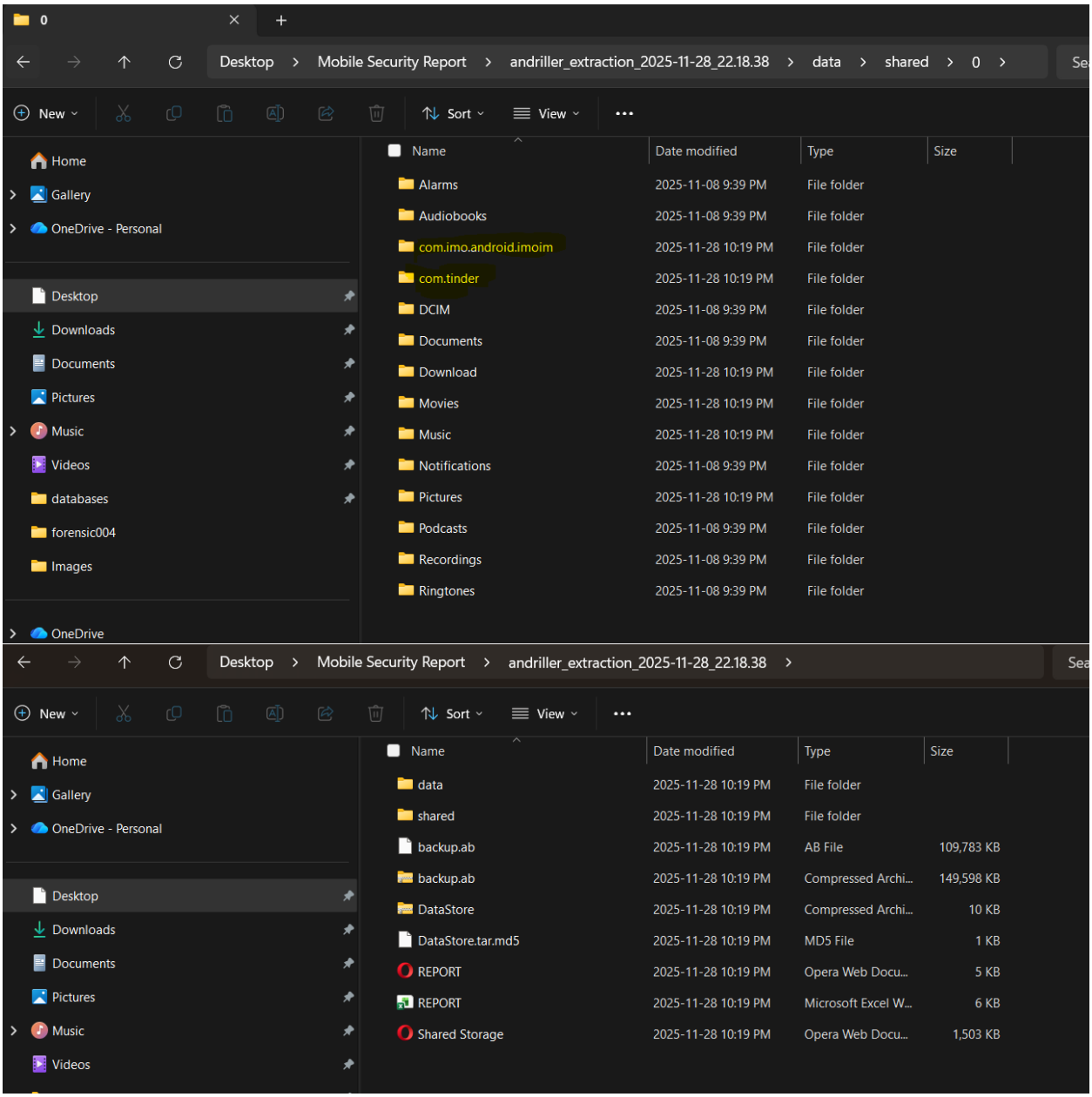


 # This report was generated using Andriller CE # (This field is editable in Preferences)

[Andriller Report]

Type	Data
Serial	emulator-5554
Status	device
Permisson	root
Ro.Build.Version.Release	15
Ro.Build.Display.Id	sdk_gphone64_x86_64-userdebug 15 AE3A.240806.043 12960925 dev-keys
Wifi Mac	02:15:b2:00:00:00
Local_Time	2025-11-28 22:18:37 Pacific Standard Time
Device_Time	2025-11-28 22:18:37 PST
Accounts	• com.imo.android.imoim: imo • com.google: mobikeaecurity@gmail.com • com.reddit.account: Reddit for Android • com.reddit.account: Akunaatakasi
Application	Shared Storage (2572)

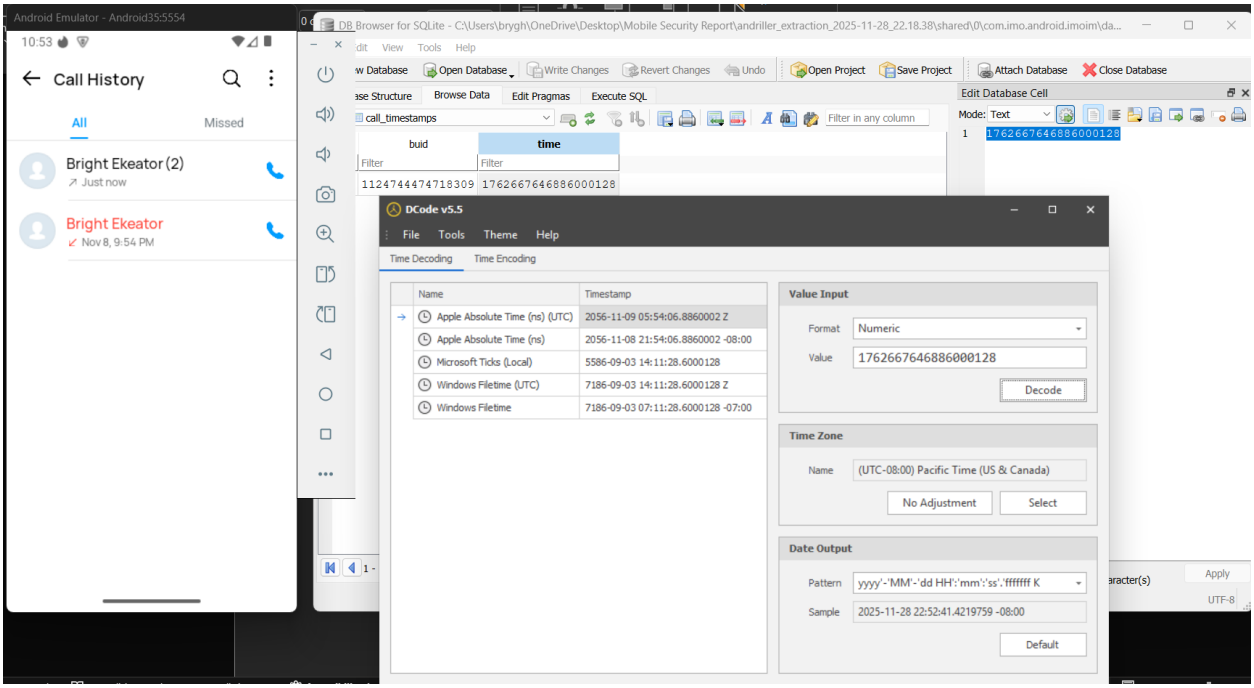
 # andriller.com # (This field is editable in Preferences)



4.0 Results and Insights (Revised with DB Browser & DCode Analysis)

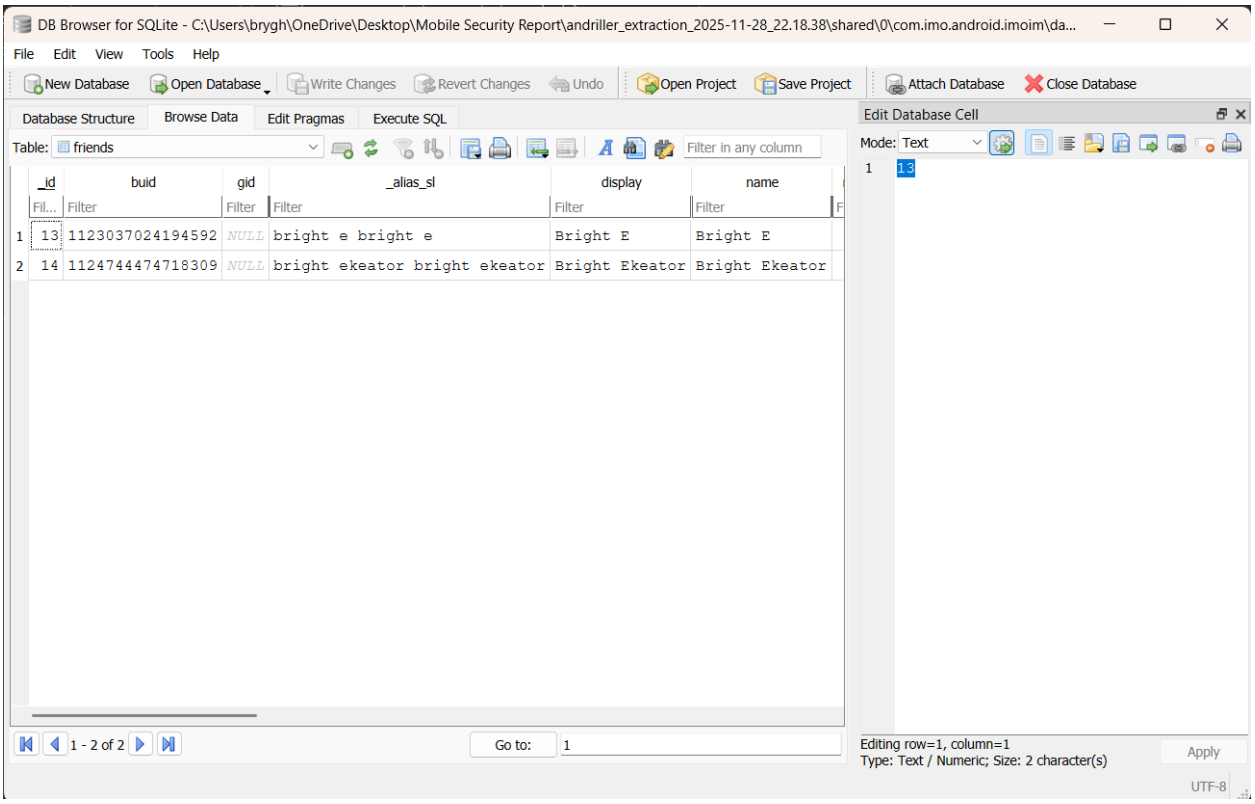
This section presents the results of the forensic investigation performed on IMO and Tinder using a combination of **Andriller**, **DB Browser for SQLite**, and **DCode**. These additional tools provided deeper insight into timestamps, metadata interpretation, hidden fields in databases, and validation of extracted artefacts.

4.1 What Did You Find?
Figure 1: IMO Call History Displayed on the Emulator



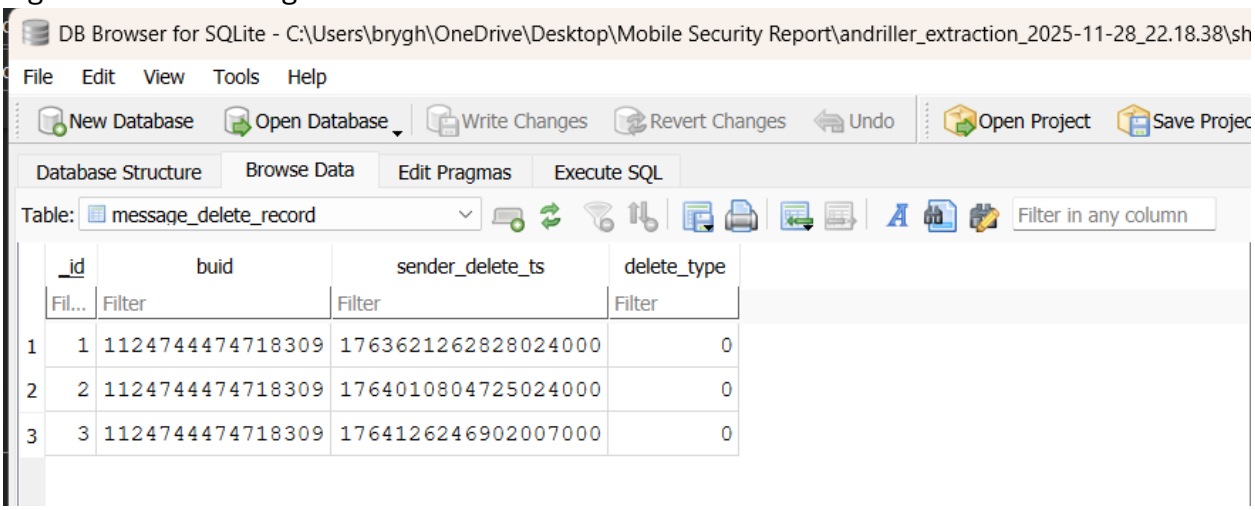
This is a result of a call log one is a missed call log on the 8th of November from the call timestamps table in the database file which the date can correlate on Dcode as well. On Dcode it shows the year is 2056 that is because it is not showing on unix time instead of Apple absolute time. To get the exact date, the time code must be divided by 1000000 before pasting on DCode for the exact time info.

Figure 2: IMO Friends



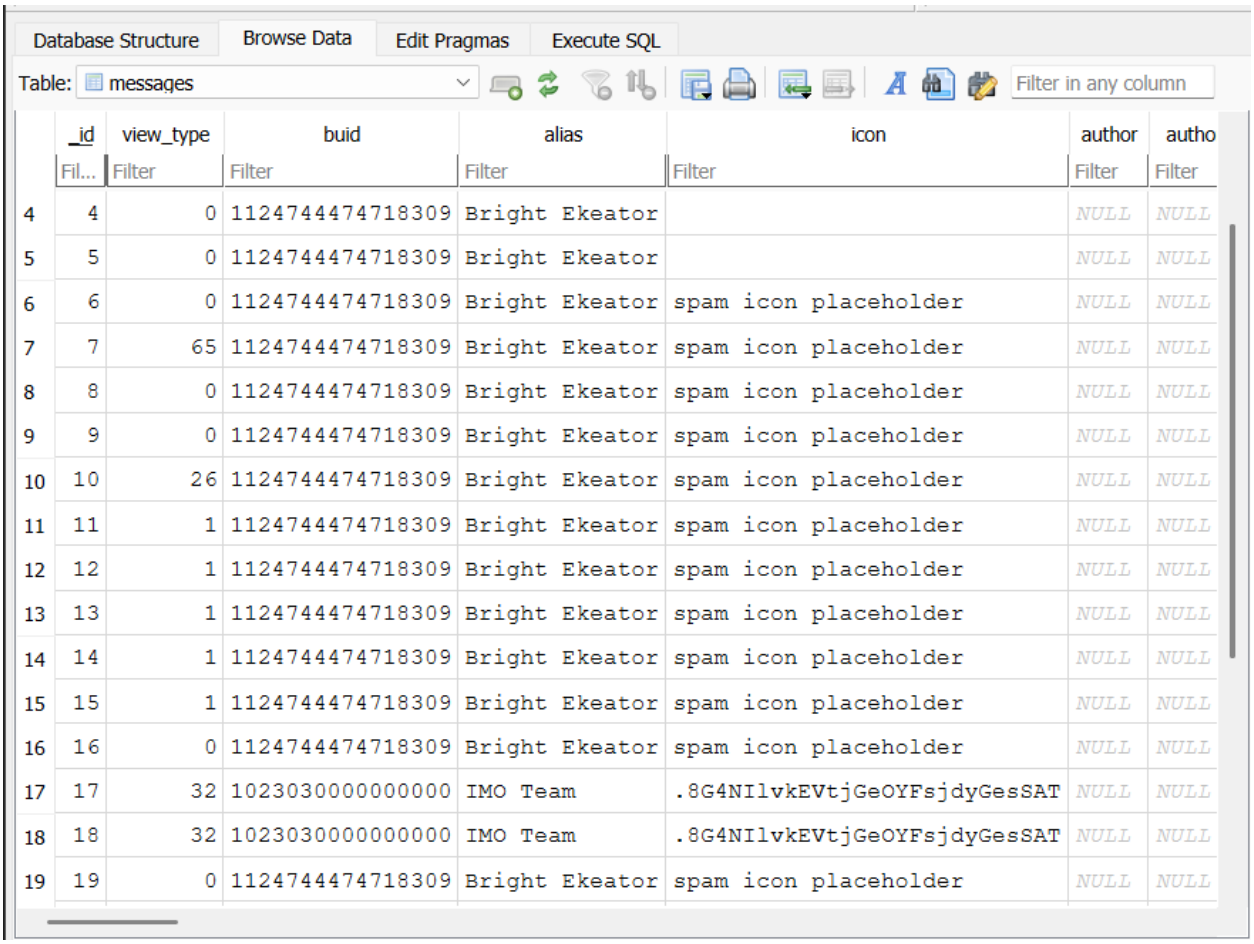
This shows 2 list of friends but in reality, the first name is the name of the profile from the extracted device, and the second name is the profile on the other end of conversation

Figure 3: IMO Message delete Record



Shows 3 messages has been deleted so far in this account

Figure 4-5 IMO Messages Record

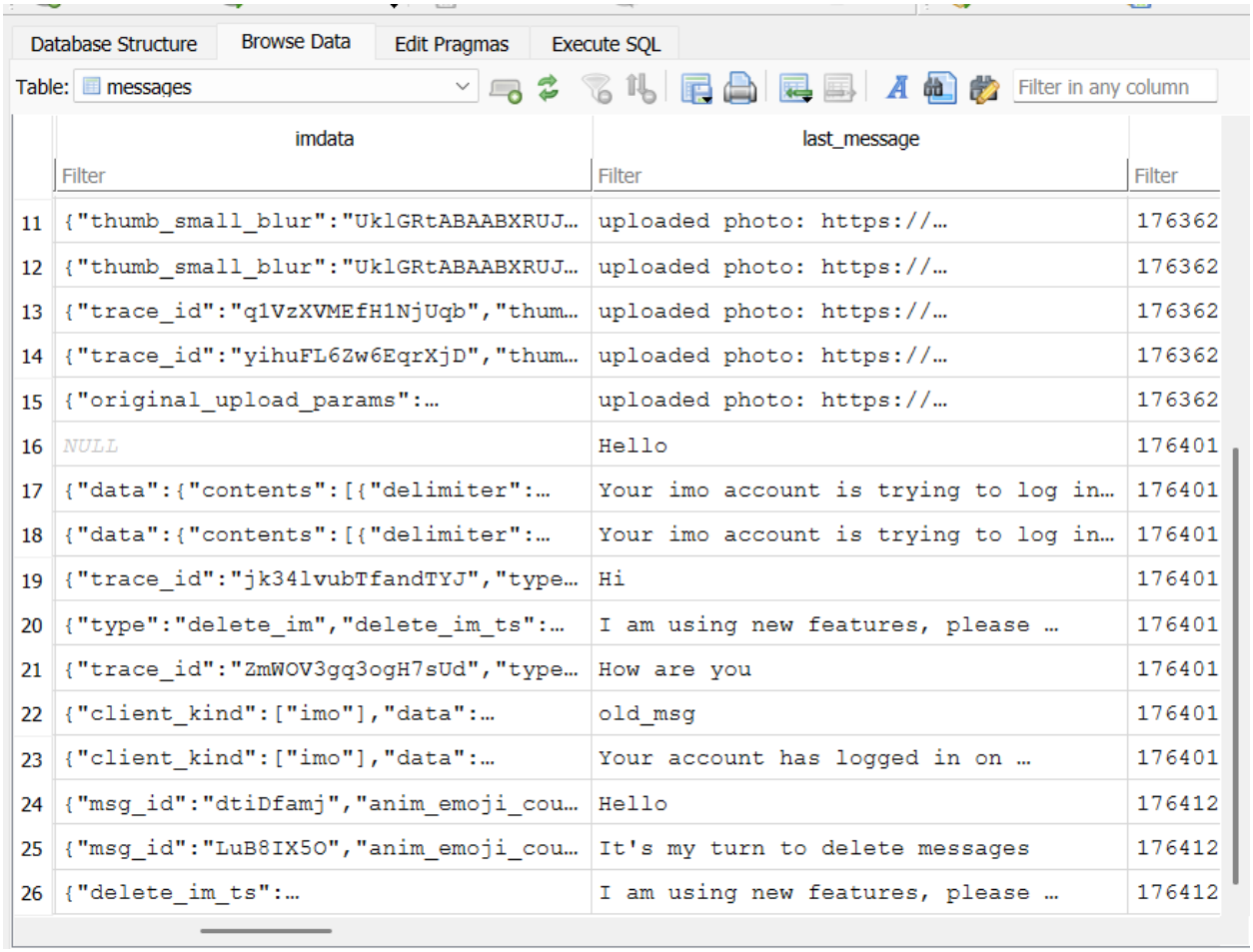


Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: messages

Filter in any column

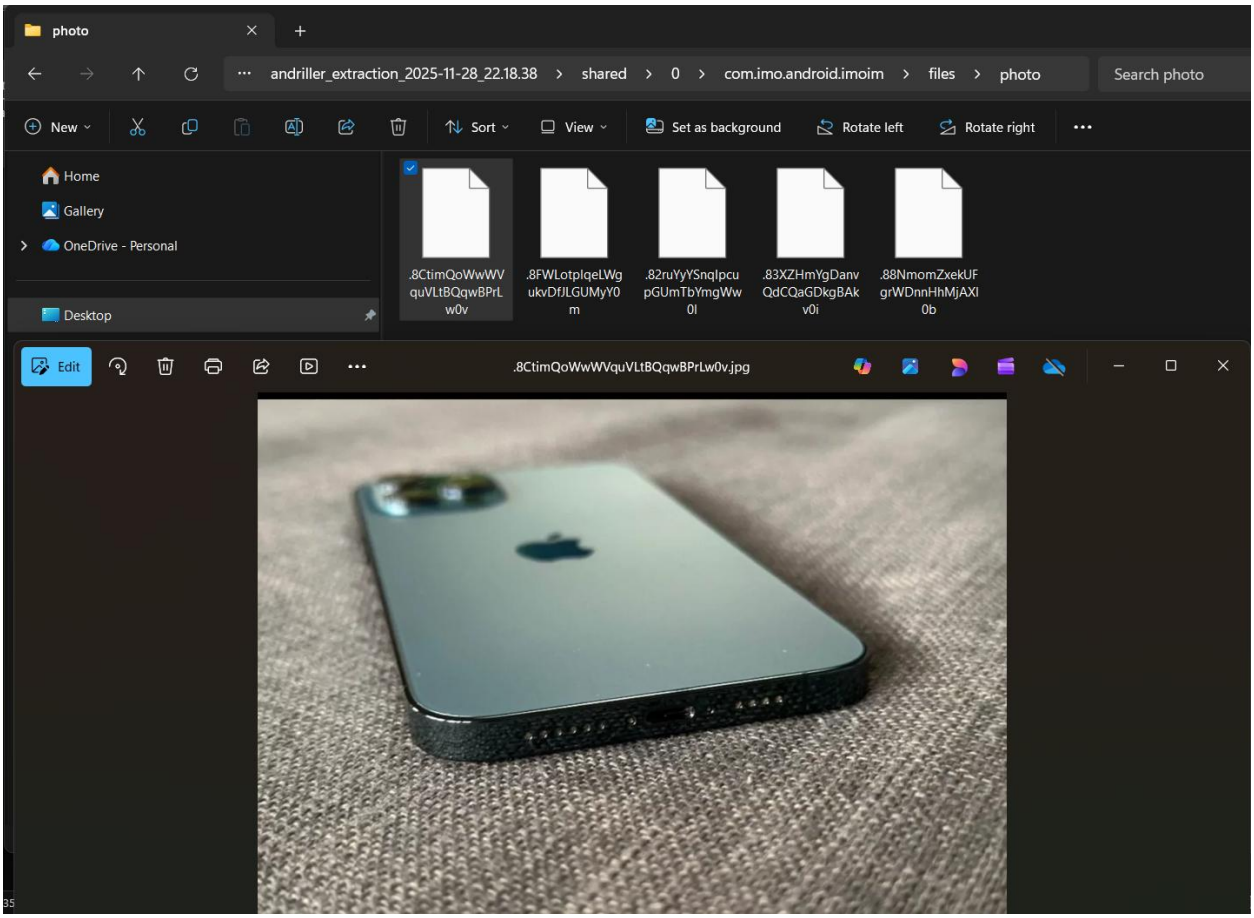
	_id	view_type	buid	alias	icon	author	autho
	Fil...	Filter	Filter	Filter	Filter	Filter	Filter
4	4	0	1124744474718309	Bright Ekeator		NULL	NULL
5	5	0	1124744474718309	Bright Ekeator		NULL	NULL
6	6	0	1124744474718309	Bright Ekeator	spam icon placeholder	NULL	NULL
7	7	65	1124744474718309	Bright Ekeator	spam icon placeholder	NULL	NULL
8	8	0	1124744474718309	Bright Ekeator	spam icon placeholder	NULL	NULL
9	9	0	1124744474718309	Bright Ekeator	spam icon placeholder	NULL	NULL
10	10	26	1124744474718309	Bright Ekeator	spam icon placeholder	NULL	NULL
11	11	1	1124744474718309	Bright Ekeator	spam icon placeholder	NULL	NULL
12	12	1	1124744474718309	Bright Ekeator	spam icon placeholder	NULL	NULL
13	13	1	1124744474718309	Bright Ekeator	spam icon placeholder	NULL	NULL
14	14	1	1124744474718309	Bright Ekeator	spam icon placeholder	NULL	NULL
15	15	1	1124744474718309	Bright Ekeator	spam icon placeholder	NULL	NULL
16	16	0	1124744474718309	Bright Ekeator	spam icon placeholder	NULL	NULL
17	17	32	1023030000000000	IMO Team	.8G4NIlvkEVtjGeOYFsjdyGesSAT	NULL	NULL
18	18	32	1023030000000000	IMO Team	.8G4NIlvkEVtjGeOYFsjdyGesSAT	NULL	NULL
19	19	0	1124744474718309	Bright Ekeator	spam icon placeholder	NULL	NULL



	imdata	last_message	
	Filter	Filter	Filter
11	{"thumb_small_blur":"UklGRtABAABXRUJ…	uploaded photo: https://…	176362
12	{"thumb_small_blur":"UklGRtABAABXRUJ…	uploaded photo: https://…	176362
13	{"trace_id":"q1VzXVMEfH1NjUqb","thum…	uploaded photo: https://…	176362
14	{"trace_id":"yihuFL6Zw6EqrXjD","thum…	uploaded photo: https://…	176362
15	{"original_upload_params":…	uploaded photo: https://…	176362
16	*NULL*	Hello	176401
17	{"data":{"contents":[{"delimiter":…	Your imo account is trying to log in…	176401
18	{"data":{"contents":[{"delimiter":…	Your imo account is trying to log in…	176401
19	{"trace_id":"jk34lvubTfandTYJ","type…	Hi	176401
20	{"type":"delete_im","delete_im_ts":…	I am using new features, please …	176401
21	{"trace_id":"ZmWOV3gq3ogH7sUd","type…	How are you	176401
22	{"client_kind":["imo"],"data":…	old_msg	176401
23	{"client_kind":["imo"],"data":…	Your account has logged in on …	176401
24	{"msg_id":"dtiDfamj","anim_emoji_cou…	Hello	176412
25	{"msg_id":"LuB8IX5O","anim_emoji_cou…	It's my turn to delete messages	176412
26	{"delete_im_ts":…	I am using new features, please …	176412

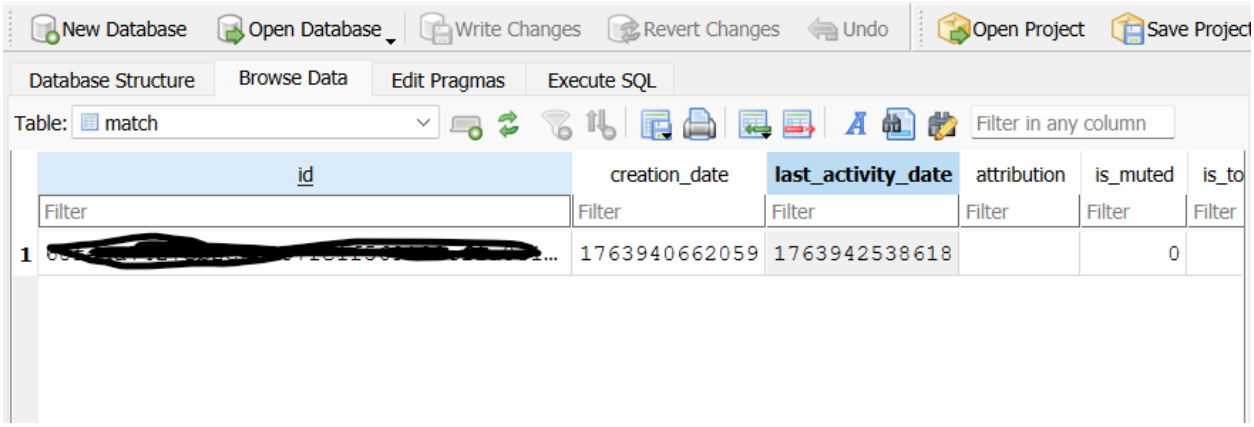
In the Messages table, you can see all the messages exchanged between contacts including the ones deleted as well. You can distinguish a atext message from a picture message from the imdata colum which clearly specifies which is which

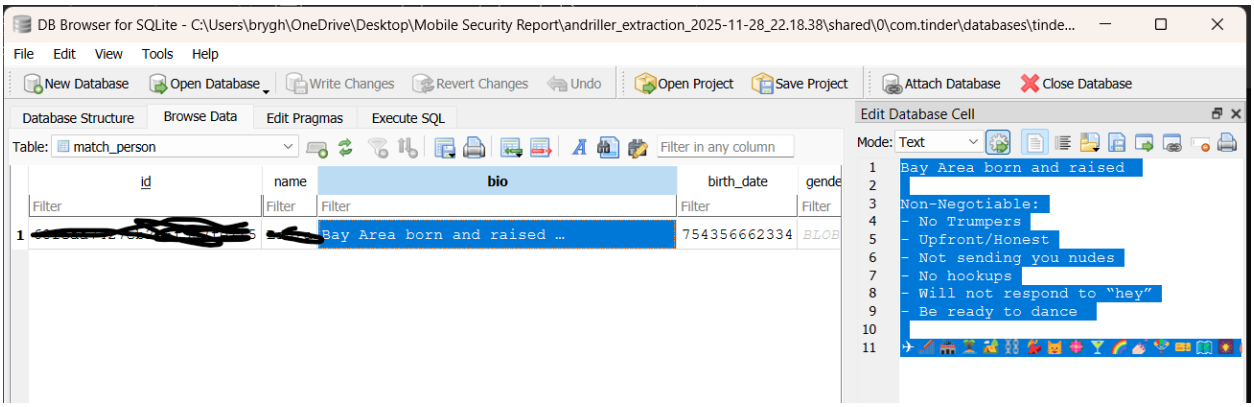
Figure 6: IMO Files folder



From this folder you can see a photo subsection which all the images transferred between contacts in a conversation is stored in.

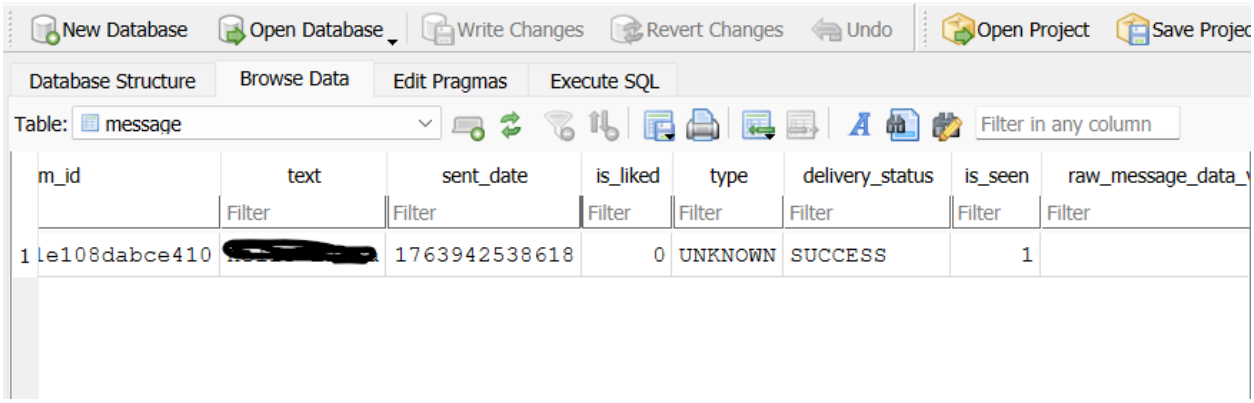
Figure 7: Tinder Match Record





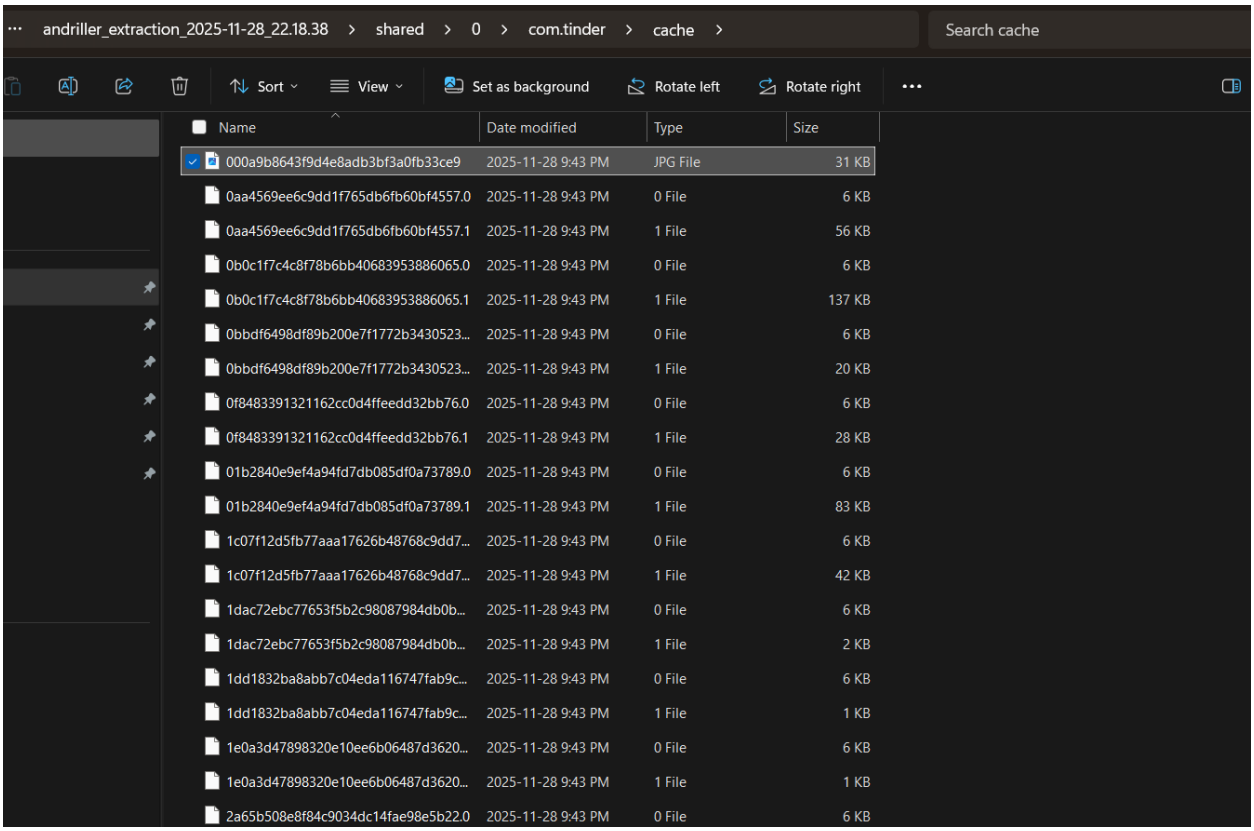
This record shows the number of matched contacts including their user id which I have blotted out for privacy reasons. You can also see the time the contact exchanged an activity with you. In the Person Matched table you can see their bio, names birthdate information including user id.

Figure 8: Tinder Message



This shows the id of the sender and receiver, the content pf the message, the date it was sent, the delivery status and if the message was seen or not.

Figure 9: Tinder Cache



The cache folder contains pictures temporary stored in the device from swipes and profiles visited during a session.

4.2 How Was the Exploration Useful?

1. Manual Verification Improved Accuracy

Andriller's automated parsing is beneficial, but the manual use of DB Browser allowed:

- Verification of parsed values
- Inspection of raw tables and BLOB fields
- Discovery of deleted entries that HTML reports did not show
- Clear view of normalized data structures and relationships between tables

This reflects a real-world forensic workflow where no examiner depends solely on automated reports.

2. Timeline Reconstruction Using DCode

The combination of SQLite timestamps + DCode produced:

- Accurate event timelines
- Distinction between message send time, received time, and call start/end time
- The ability to correlate events across IMO and Tinder
- A chronological understanding of user activity

This is vital in forensic cases involving harassment, stalking, fraud, or communication-pattern analysis.

3. Strengthened Evidence Validation

Using two additional tools allowed:

- Cross-tool validation of artefacts
- Greater confidence in recovered metadata
- Ability to interpret fields Adriller couldn't decode
- Recovery of partial data from Tinder that automated tools largely overlooked

This demonstrates multi-tool forensic methodology.

4.3 Limits of the Research and Exploration

While the forensic investigation yielded meaningful insights, several limitations impacted the extent and completeness of the analysis. These limitations arose from **app design decisions**, **data storage models**, and **technical extraction constraints**.

1. Andriller's Tool Limitations

Although powerful, Andriller has limitations:

- Some app-specific decoders are outdated
- Newer Tinder/IMO schema changes are unsupported
- JSON and XML files occasionally triggered parsing errors
- Long filenames and Windows path issues sometimes caused extraction failures

2. Tinder's Swipe-Based Communication Model Restricts Evidence Collection

Tinder is inherently a **match-based, swipe-driven communication app**, which imposes a structural limitation on the amount of evidence the investigator can retrieve:

- A user **cannot communicate** with another profile **unless both parties swipe right** (a mutual match).
- If the examiner does **not receive a "swipe back,"** Tinder does not generate:
  - A message thread
  - A match ID
  - A conversation record
  - Any metadata indicating attempted communication
- The only forensic artefacts available for unmatched profiles are **cached profile images** temporarily stored in the /cache/ directory.

However:

There is **no database table or SQLite record** that stores:

- The list of profiles **the user swiped right on**
- The list of profiles **the user swiped left on**
- The set of profiles presented to the user but not engaged with

This creates a significant visibility gap because:

Tinder maintains no local record of all viewed users or all outgoing swipes.

Only successful matches appear in local storage.

This limits the forensic examiner's ability to reconstruct:

- The full history of potential contacts the user interacted with
- Behavioral patterns such as swipe frequency, selection tendencies, or rejected profiles

This is an inherent design limitation and not a failure of the forensic tool.

5.0 Conclusions

The forensic analysis of IMO and Tinder with Andriller, DB Browser for SQLite, and DCode showed what modern mobile app forensics can and cannot do. By following a real investigative process from device setup to data extraction and analysis, this project offered useful insights into how different apps handle user data on devices.

5.1 Future Work

Although the investigation successfully extracted and analyzed key artefacts, several opportunities exist to extend this work:

1. Expand Analysis to Physical Android Devices

Using actual devices instead of emulators would reveal:

- Additional artefacts stored in hardware-backed secure storage
- Cached files and logs that emulators may not generate
- Device-specific metadata such as SIM information, GPS history, and network logs

A real device would give a richer dataset and a more realistic forensic scenario.

2. Explore Rooted Device Extraction

Root access would enable:

- Full access to /data/data/ without permission restrictions
- Acquisition of protected files such as encrypted keys or hidden logs
- Deeper recovery of deleted or unallocated SQLite pages
- Direct collection of app-specific secure storage

This could yield more artefacts, especially for apps like Tinder with restrictive data storage.

3. Test Additional Mobile Forensic Tools

Comparing Andriller with tools such as:

- **Cellebrite UFED**
- **Magnet AXIOM**
- **MOBEXLER**
- **Autopsy (Mobile module)**

could highlight differences in extraction capability, timestamp interpretation, and artefact completeness.

4. Explore More Social/Communication Apps

Analyzing other apps like WhatsApp, Telegram, Instagram, Messenger, would help generalize patterns in mobile app data retention and privacy models.

5.2 Additional Scenarios Where These Insights Could Apply

The insights gained from this investigation can be adapted to several real-world forensic and cyber investigations:

1. Harassment, Cyberbullying, or Threat Cases (IMO)

Recovered call logs, contact metadata, and timestamps can help:

- Establish communication timelines

- Identify involved users
- Prove repeated or unwanted contact
- Reconstruct patterns of interaction

2. Romance Fraud, Impersonation, or Catfishing Investigations (Tinder)
Metadata from matches, profile IDs, and cache images may support cases involving:

- Fraudulent identity claims
- Fake profiles used to manipulate victims
- Misrepresentation during online dating
- Behavioural profiling of the suspect's app usage

3. Digital Behavioural Analysis
The contrast between IMO's rich local logs and Tinder's minimal on-device data provides insight for:

- Understanding which apps store what types of evidence
- Prioritizing extraction strategies
- Predicting the likelihood of data recovery from different apps

4. Corporate or Internal Investigations
Organizations examining inappropriate communications or device misuse may apply the extracted insights to:

- Trace calls and messages on corporate-managed devices
- Recover deleted or hidden interactions
- Validate activity timelines through timestamp decoding

5.3 Broad Conclusions
1. IMO Stores Extensive Local Evidence
IMO maintains substantial on-device artefacts including:

- Message metadata
- Call logs
- Contacts
- Configuration files
- Authentication tokens

This makes IMO a **high-value target** for forensic investigations.

2. Tinder Stores Limited Local Evidence Due to Its App Design
Tinder's design philosophy limits local storage:

- Messages stored in the cloud, not on-device
- No logs of swiped-left users
- Only matches and minimal metadata stored locally
- Cached images present but not linked to database entries

This restricts the depth of forensic recovery but still allows partial timeline reconstruction.

3. Multi-Tool Analysis Strengthens Forensic Conclusions

Using Andriller alone would not have revealed:

- Deleted or hidden SQLite entries
- Precise timestamp conversions
- Cross-database correlations

DB Browser and DCode significantly enhanced the accuracy and depth of the investigation.

4. Forensic Workflow Matters as Much as the Tool

This project demonstrated that successful mobile forensics depends on:

- A structured extraction plan
- Verification of artefacts using multiple tools
- Understanding of app architecture
- Critical evaluation of tool limitations

Automation helps, but manual validation is essential.

5. App Architecture Directly Influences Forensic Recoverability

Messaging apps (IMO) retain extensive user traces, while swipe-based apps (Tinder) intentionally minimize local artefacts.

This distinction is crucial for:

- Setting realistic expectations of recoverable data
- Choosing extraction techniques
- Evaluating evidentiary value in investigative scenarios

5.4 Final Statement

This project provides a detailed and practical look at mobile application forensics on Android devices. It highlights the importance of using both automated tools and manual analysis, understanding how each app works, and adjusting forensic methods for today's privacy focused apps.

6.0 AI Use Section

Table of AI Tools and Specific Use

AI Tool Name	Version, Account Type	Specific feature for which the AI tool was used
Gemini (LLM)	Pro, Free	Drafting the initial project proposal structure and outlining the novelty sections for each student
ChatGPT (GPT-5.1)	Free Version	Used for structuring report and formatting. Solutions to tinder APK debugging

ChatGPT (Earlier sessions)	Free Version	Clarified ADB and forensic workflow assisted in andriller installation and use
Microsoft Word Editor	Local	Grammer and formatting
ChatGPT (OpenAI)	GPT-5.1, *ChatGPT Plus (Premium)*	Assisted in drafting sections of the forensic report, generating structured explanations, refining grammar and clarity, and helping format results into academic style.
ChatGPT Code Interpreter / Analytical Mode	Built-in to GPT-5.1, *ChatGPT Plus*	Helped interpret database structures, explain timestamp formats, and provide reasoning for SQLite, JSON, and metadata analysis steps.

6.1 Value Addition

Although AI tools assisted with wording, formatting, and explanations, **all practical forensic work, experimentation, screenshots, and analysis were performed manually** by the student.

Specific value-added contributions include:

- **Manually running ADB commands** to connect, extract, and verify the Android emulator.
- **Performing real forensic extractions** using Andriller, not AI.
- **Investigating IMO and Tinder databases** using DB Browser for SQLite.
- **Independently decoding timestamps** using DCode and correlating them with app UI data.
- **Troubleshooting extraction failures** (empty AB backups, connection issues, unsupported formats).
- **Generating real screenshots** from emulator, DB Browser, and DCode as evidence.
- **Critically interpreting findings**, including identifying Tinder's metadata limitations and IMO's richer local storage.
- **Making all technical decisions**, including methods, tools, and forensic workflows.

AI provided **writing support**, but **all digital forensics procedures, results, verification, and insights were personally executed** by the student.

6.2 Appendix

- How can I solve the sign in problems of my tinder
- How to install xapk in place of apk on android studio emulator via command line

- Step by step guide on how to use andriller
- Explain in detail why I cannot view the extracted message sent between contacts on my IMO without decryption key
- Debugging my tinder app issues
- Guides on comparative analysis between andriller backup files and adb backup files
- Fix grammar and adjust academic tone.
- How do I fix ADB connection timeouts
- How else can I view dd files besides mounting in Linux?
- Troubleshoot Andriller unroot on emulator

7.0 References

- Andriller: Android Forensics Tool. (2024). Retrieved from https://github.com/BlackArrow/andriller
- Imo Application. (2025). Retrieved from https://imo.im/
- Tinder Application. (2025). Retrieved from https://tinder.com/
- GitHub Documentation. (2025). Using repositories for team collaboration. Retrieved from https://docs.github.com/
- OpenAI ChatGPT. (2025). Used for conceptual planning and drafting sections of the proposal. Retrieved from https://openai.com
- Tinder. (2023). *Tinder Privacy Policy & Data Handling*. https://www.gotinder.com/privacy
- DB Browser for SQLite. (2023). *DB Browser for SQLite* [Software]. https://sqlitebrowser.org

8.0 Work Date/Hours Log

Date	Student Name	Number of Hours	Description of Work
2025-10-20	Bright Ekeator	30 Minutes	Met with instructor to discuss project scope, tool selection, and proposal feedback.
2025-10-20	Bright Ekeator	1 hour	Team meeting to discuss project objectives, divide tasks, and plan analysis workflow for selected apps.

2025-10-24	Bright Ekeator	2 hours	Team meeting to set up the project proposal structure and created the GitHub repository for collaborative work.
2025-10-24	Bright Ekeator	3 hrs	Installed Andriller and set up test environment for Imo and Tinder APK extraction.
2025-10-27	Bright Ekeator	1 hour	Installed and configured Android Studio and AVD emulator with rooted image.
2025-10-28	Bright Ekeator	30 mins	Installed IMO app on emulator and confirmed package path
2025-10-29	Bright Ekeator	30 mins	Attempted adb backup on IMO; backup produced 1 KB file.
2025-10-30	Bright Ekeator	1 hour	Pulled internal IMO app data using adb pull.
2025-10-31	Bright Ekeator	30 mins	Organized project folders for analysis outputs
2025-11-01	Bright Ekeator	1 hour	Installed Python environment and Andriller; verified modules
2025-11-02	Bright Ekeator	1 hour	Extracted .ab backup using Andriller and decoded data
2025-11-03	Bright Ekeator	1 hour	Opened SQLite databases and reviewed message tables.

2025-11-04	Bright Ekeator	1 hour	Identified encrypted BLOB message content.
2025-11-05	Bright Ekeator	1 hour	Analyzed shared preferences for configuration data.
2025-11-06	Bright Ekeator	1 hour	Documented folder sizes and data extraction process.
2025-11-07	Bright Ekeator	1 hour	Drafted initial analysis report.
2025-11-07	Bright Ekeator	1 hour	Re-verified data integrity and repeated extraction steps
2025-11-08	Bright Ekeator	3 hours	Researched runtime key extraction and alternative decryption methods
2025-11-08	Bright Ekeator	2 hours	Reviewed all databases and performed additional test extractions.
2025-11-12	Bright Ekeator	30 mins	Tried Installing another APk file for Tinder and tried sign in but unsuccessful
2025-11-14	Bright Ekeator	30 mins	Prepared Progress report and compared previous analysis
2025-11-15	Bright Ekeator	2 hours	Attempted multiple Tinder APK installations (v14–v18), tested compatibility on emulator, collected crash logs.
2025-11-16	Bright Ekeator	1.5 hours	Research on Tinder root/jailbreak detection; attempted patching

			via apktool. **(Debugging time: 1 hr included)**
2025-11-17	Bright Ekeator	2 hours	Set up new emulator images (Android 11 & 12) to test APK compatibility.
2025-11-18	Bright Ekeator	1.5 hours	Used DCode to decode additional IMO timestamps; manually correlated values with call logs
2025-11-19	Bright Ekeator	2 hours	Deep dive into IMO shared preference XML files; recorded authentication token behaviour and storage paths.
2025-11-20	Bright Ekeator	3 hours	Full extraction test using Andriller TAR method; compared AB backup vs TAR output. **(Debugging time: 1 hr included — Andriller extraction errors)**
2025-11-21	Bright Ekeator	2 hours	SQLite DB Browser manual inspection of IMO call logs and message metadata; exported tables to CSV for report.
2025-11-22	Bright Ekeator	2 hours	Mapped Tinder cache directory contents; identified profile thumbnails and unmatched user artefacts.
2025-11-23	Bright Ekeator	3 hours	Attempted ADB-over-TCP connection between

			Windows host and Kali VM; resolved network bridging issues. **(Debugging time: 2 hrs)**
2025-11-24	Bright Ekeator	2 hours	Re-extracted IMO databases after emulator reset; validated consistency of timestamps across runs.
2025-11-25	Bright Ekeator	3 hours	Drafted Detailed Analysis section with screenshots; wrote correlation for call timestamps (UI → DB → DCode).
2025-11-26	Bright Ekeator	3 hours	Tinder forensic gap assessment: tested API-based message absence, confirmed no storage of unmatched swipes.
2025-11-27	Bright Ekeator	2 hours	Re-tested Andriller on Tinder TAR extraction; confirmed partial metadata recovery; documented database schema.
2025-11-28	Bright Ekeator	3 hours	Built Results & Insights section, focusing on manual DB Browser analysis and timestamp decoding. **(Debugging time: 1 hr — inconsistent timestamp formats)**

2025-11-29	Bright Ekeator	3.5 hours	Final report assembly
2025-11-29	Bright Ekeator	1 hour	Work Integration
2025-11-29	Bright Ekeator	1 hour	Video recording for submission
2025-11-29	Bright Ekeator	30 mins	Submission and push to GitHub.