Exploring Data Extraction from iOS Devices

What Data You Can Access and How

By Mattia Epifani - September 30, 2025

Source: ZENA FORENSICS - blog.digital-forensics.it

Following the previous post dedicated to Android devices, this article outlines the data available on iOS devices, depending on the different forensic acquisitions that can be made. The objective is not to propose dedicated guidelines, but to provide a comparison between the data present within different acquisitions that can be obtained from an iOS device, analyzing the specificities of Apple's operating system and related forensic implications.

iOS Data Protection: The Core of Apple Security

The two main elements that determine data extraction possibilities from an iOS smartphone are the device state (AFU or BFU) and the availability/knowledge of the access code/password.

In the iOS ecosystem, the main reference is Apple's Platform Security document. The concept of **Data Protection** is particularly relevant - Apple's proprietary technology used to protect data stored on devices with Apple SoC.

Data Protection Overview

Apple uses Data Protection technology to safeguard data stored in flash memory of devices including iPhone, iPad, Mac with Apple silicon, Apple TV, Apple Watch, and Apple Vision Pro. This technology enables devices to respond to common events (such as incoming calls) while maintaining a high level of encryption for user data.

Technical Implementation

Data Protection is implemented by constructing and managing a key hierarchy based on hardware encryption technologies integrated into Apple devices. Protection is controlled on a per-file basis by assigning each file to a specific protection class.

Every time a file is created in the data volume, Data Protection generates a new 256-bit key (per-file key) and provides it to the hardware AES Engine, which uses the key to encrypt the file during writing to flash storage:

- **A14-A18 and M1-M4 devices:** AES-256 encryption in XTS mode

- **A9-A13 and S5-S9 devices:** AES-128 encryption in XTS mode

Data Protection Classes

When a new file is created on devices with Data Protection, it is assigned to a specific class by the application that creates it. Each class uses different policies to determine when data is accessible:

Class A: Complete Protection (NSFileProtectionComplete)

The class key is protected with a key derived from the user's passcode and device UID. Shortly after the user locks the device (10 seconds if 'Require Password' is set to 'Immediately'), the decrypted class key is discarded, making all data in this class inaccessible until the passcode is entered again.

Class B: Protected Unless Open (NSFileProtectionCompleteUnlessOpen)

Some files may need to be written while the device is locked. This behavior is achieved using asymmetric elliptic curve cryptography (ECDH over Curve25519). As soon as the file is closed, the per-file key is wiped from memory.

Class C: Protected Until First User Authentication (NSFileProtectionCompleteUntilFirstUserAuthentication)

This class behaves like Complete Protection, except that the decrypted class key is not removed from memory when the device is locked. This is the default class for all third-party app data not otherwise assigned to a Data Protection class.

Class D: No Protection (NSFileProtectionNone)

This class key is protected only with the UID and is kept in Effaceable Storage. Since all keys needed to decrypt files in this class are stored on the device, encryption only provides the benefit of fast remote wipe.

The presence of 4 encryption classes makes data qualification more complex compared to Android, distinguishing between data available in BFU and AFU modes:

- **BFU mode:** only Class D files can be decrypted
- **AFU mode:** Class C and D files can be decrypted
- **With known passcode:** files from all classes (A-D) can be decrypted, obtaining a Full File System acquisition

Keychain Protection Classes

Elements within the Keychain are also protected with different protection classes. Keychain items are encrypted using two different AES-256-GCM keys: a table key (metadata) and a per-row key (secret key).

The keychain is implemented as a SQLite database stored in the file system. There is only one database, and the securityd daemon determines which keychain items each process or app can access.

Main Keychain protection classes:

- **After first unlock:** Wi-Fi passwords, mail accounts, social network account tokens, iMessage keys
- **When unlocked:** Safari passwords, Safari bookmarks, Home sharing password
- **Always:** Bluetooth keys, APNs tokens, iCloud certificates

iOS Data Extraction Scenarios

Scenario 1: BFU Device Without Passcode

This scenario requires a tool that allows the extraction of data whose protection does not depend on the user-set passcode (Class D files) and the possible ability to perform a cracking attack on the access code. The result will be a **BFU acquisition**. The availability of free or open-source tools for this type of acquisition is limited to outdated, vulnerable models.

Scenario 2: AFU Device Without Passcode

This requires a tool that allows AFU acquisitions, i.e., accessing Class C and D data without needing passcode cracking. The resulting acquisition is an **AFU** type.

Scenario 3-4: Device With Known Passcode

When the passcode is known, data accessibility depends on the availability of a tool that can bypass iOS restrictions by obtaining 'root' level access. If available, the result will be a **Full File System (FFS)** acquisition; otherwise, it will be a **Logical** acquisition.

iOS Logical Extraction Techniques

iOS logical extraction consists of interacting with the device using natively available protocols. Main techniques include:

iTunes Backup

iTunes backup can be extracted in encrypted or unencrypted mode. An encrypted backup contains significantly more data than an unencrypted one (i.e., Call Log, Apple Health, and Safari History). Critical aspects to consider:

- **Backup password:** if already set by the user, cracking may be necessary
- **Password reset:** since iOS 11, reset possible via 'Reset all settings' (with consequent loss of some configurations)
- **SQLite Vacuuming:** Apple has modified backup behavior by implementing automatic SQLite database 'vacuuming' activities

AFC Service (Apple File Connection)

Used to extract the /private/var/mobile/Media/ folder, containing mainly multimedia files and Apple Photos application data.

Shared 'Documents' Folders

'Documents' folders shared by each app that chooses to enable this functionality.

Lockdown Service

Device information obtained through the Lockdown daemon: device name, iOS version, IMEI, UDID, Serial Number, and installed applications details.

Crash Logs

Extracted from /private/var/mobile/Library/Logs/CrashReporter/ path. They typically don't contain user data but are useful for determining application execution timelines and malware analysis activities.

Sysdiagnose

A native application that generates an archive containing the results of specific commands and system file collection. The generated file is stored in the 'CrashReporter' folder.

iOS Unified Logs

A centralized logging system was introduced in iOS 10. Unified Logs provide detailed information on:

- System and application activities
- Security and authentication events
- User interactions with the device
- Diagnostics and performance

Syslog

Live system syslog that records events in real-time until recording interruption. Useful for creating test data on comparison devices and examining processes currently occurring on the device.

Free and Open Source Tools

For about a year, the **UFADE** tool (https://github.com/prosch88/UFADE) has been available, allowing exploitation of all mentioned communication channels to extract all accessible data without requiring root exploits. Functionality details are available on the author's blog (https://cp-df.com/en/ufade/).

iTunes Backup Operational Procedure

iTunes Backup

iTunes backup is a native iOS device functionality that allows the owner to perform a local device backup. The backup content should be considered a partial File System extraction: for operating system and native application configurations and logs, the operating system manages which data will be included in an iTunes backup; for third-party applications, each individual application defines which files should be included in an iTunes Backup.

It should also be noted that an iTunes backup can be encrypted with a password: an encrypted backup contains more data than an unencrypted one. Therefore, it is essential during the acquisition phase to verify whether a backup password has been previously set on the device by the user.

Password Management Scenarios

If the password is not set, you can proceed by setting it to a known password and then creating an encrypted backup using that password. At the end of the process, it is possible to disable the password if needed.

If the password is already set by the user, two scenarios are possible:

1. The backup password is known
2. The backup password is not known, which opens two further scenarios:

• The backup password can be cracked

• The time required to crack the password is too high

Backup Password Reset Procedure

To extract an iTunes Backup in any case, even when facing a device with an already set password, starting from iOS 11, Apple has provided a method to reset the backup password (Settings → General → Transfer or Reset iPhone → Reset → Reset all settings). It is important to highlight that, in case of a backup password reset, other data will also be deleted. In particular:

• Display brightness/NightShift settings

• Home screen layout

• Set wallpaper

• Stored Wi-Fi networks and passwords

• Known Bluetooth devices

• VPN settings

• Keyboard settings

• Apple Pay information

• HomeKit configuration

For this reason, when proceeding with an 'iTunes backup' acquisition where the backup password stored on the phone is not known, the following operational procedure can be evaluated:

1. **Create an iTunes backup** with the unknown password
2. **Generate and extract a 'sysdiagnose'**
3. **Reset the unknown backup password** and acquire a new backup, ensuring to set a known password

SQLite Vacuuming: A Critical Change in iOS 17.4

It has been observed over time that Apple has modified the backup procedure behavior in relation to SQLite database 'vacuuming' activities. This change has significant implications for forensic data recovery.

When data is deleted from a SQLite database, the deletion does not necessarily occur immediately. Rather, the area that contained the deleted data is marked or flagged as 'available space' to be used by the database to store new data when it arrives. The VACUUM command cleans up any 'available space' areas by rebuilding the database file and repacking the data inside the database so it can occupy the smallest amount of required space.

iOS Acquisition Types

Based on the described scenarios, the analyst may encounter one of the following acquisitions:

1. **BFU Acquisition:** from a locked and powered-off device
2. **Logical Acquisition:** from an unlocked device with native techniques
3. **AFU Acquisition:** from a locked device in AFU state
4. **Full File System (FFS) Acquisition:** complete file system access
5. **FFS with KeyChain Acquisition:** includes decryption keys from the keychain

Data Analysis by Acquisition Type

iOS BFU Acquisition

On iOS devices, most files are stored with protection class C or higher. The set of Class D files is limited but may contain useful information for:

Information Available in BFU:

- Device setup and installation date
- Apple ID (iCloud email)
- Other email addresses in use
- Keychain Authentication Tokens not protected by the passcode
- Phone number and SIM information (CellularUsage.db)
- List of installed applications (MobileInstallation logs)
- Application Usage (DataUsage.sqlite)
- Connected Wi-Fi networks
- Bluetooth devices
- Images and videos (application cache)
- Shutdown logs and last shutdown date

BFU Acquisition Utility:

- Device user profiling
- Wordlist generation for dictionary attack
- Usage timeline determination
- Assessment of the opportunity to invest in passcode cracking

Limitations:

- Limited access to user's personal data

- Most files in /private/var/mobile encrypted with Class A, B, or C
- Variable behavior per individual application

iOS Logical Acquisition

The result strongly depends on the tool used and the functionalities implemented during extraction.

iTunes Backup - Operating System Configurations:

- **Device Model Number and Device Name**

+ /private/var/preferences/SystemConfiguration/preferences.plist

- **Serial Number**

+ /private/var/root/Library/Caches/locationd/consolidated.db

- **Last Known ICCID**

+ /private/var/wireless/Library/Preferences/com.apple.commcenter.plist

- **SIM Card information**

+ /private/var/wireless/Library/Databases/CellularUsage.db

Native Applications:

- **Accounts:** /private/var/mobile/Library/Accounts/Accounts3.sqlite
- **Address Book:** /private/var/mobile/Library/AddressBook/AddressBook.sqlitedb
- **Calendar:** /private/var/mobile/Library/Calendar/Calendar.sqlitedb
- **Call History:** /private/var/mobile/Library/CallHistoryDB/CallHistory.storedata
- **Files:** /private/var/mobile/Library/Application Support/CloudDocs/
- **Maps:** /private/var/mobile/Containers/Shared/AppGroup/Library/Maps/
- **Media (Photos/Videos):** /private/var/mobile/Media/DCIM/*, Photos.sqlite
- **Notes:** /private/var/mobile/Library/Notes/notes.sqlite
- **Safari:** Bookmarks.db, History.db, SafariTabs.db
- **SMS/MMS/iMessage:** /private/var/mobile/Library/SMS/sms.db
- **Shortcuts:** /private/var/mobile/Library/Shortcuts/Shortcuts.sqlite
- **Voicemail:** /private/var/mobile/Library/Voicemail/voicemail.db
- **Weather:** group.com.apple.weather.plist

Networks and Connected Devices:

- **Bluetooth:** com.apple.MobileBluetooth.ledevices.paired.db
- **Wi-Fi Networks:** com.apple.wifi.known-networks.plist

- **Home Kit (IoT):** /private/var/mobile/homed/datastore3.sqlite

Application Usage:

- **Application State:** /private/var/mobile/FrontBoard/applicationState.db
- **iOS Screen Layout:** /private/var/mobile/Library/SpringBoard/IconState.plist
- **TCC (Transparency, Consent and Control):** /private/var/mobile/Library/TCC/TCC.db
- **Data Usage:** /private/var/wireless/Library/Databases/DataUsage.sqlite

Pattern of Life:

- **Health:** healthdb.sqlite, healthdb_secure.sqlite
- **Keyboard Usage Stats:** user_model_database.sqlite
- **InteractionC:** interactionC.db
- **Personalization Portrait:** PPSQLDatabase.db
- **Recents:** /private/var/mobile/Library/Recents/Recents

iOS AFU Acquisition

AFU state device with an appropriate exploit allows access to all files with protection classes C and D. AFU acquisition is a **'Partial File System'** that extracts more data than logical acquisition, but cannot access files with protection class A or B.

Additional AFU Data:

- Device usage-related artifacts

+ Notifications (DuetExpertCenter folder)

+ CurrentPowerLog.PLSQL (complete)

+ Biome

+ knowledgec.db

+ Network Usage (netusage.sqlite)

+ Aggregated Dictionary (ADDataStore.sqlitedb)

+ Screen Time (RMAdminStore-Local.sqlite / RMAdminStore-Cloud.sqlite)

iOS Full File System Acquisition

Full File System acquisition is the most complete for FBE devices. It includes all data from previous acquisitions plus:

FFS-Exclusive Files:

- **Emails**
- **Location-related artifacts:**

+ Cached locations (Cache.sqlite)

+ Wi-Fi/Cell towers locations (cache_encryptedB.db)

+ Motion History (cache_encryptedC.db)

+ Significant Location (local.sqlite/cloud-V2.sqlite)

+ FindMy Devices

- **Calendar**
- **Apple Health data**

KeyChain Importance:

The ability to access keys stored in iOS keychain is crucial for decrypting data from specific applications:

Applications Requiring Keychain Keys:

- **Signal** (org.whispersystems.signal)
- **Session** (com.loki-project.loki-messenger)

- **AWS Wickr** (com.wickr.pro.prod)
- **MEGA** (mega.ios)
- **Proton Mail** (ch.protonmail.protonmail)
- **Threema** (ch.threema.iapp)

iOS-Specific Limitations and Considerations

Deleted Data Recovery

- **Deleted files:** practically impossible to recover (encryption keys eliminated immediately)
- **Some artifacts persist temporarily:**

+ Deleted photos/videos: 30 days

+ Safari history: 30 days

+ Deleted SMS/iMessage: 30 days

+ KnowledgeC/Biome databases: approximately 28 days

+ Location Service cache: 7 days

Wiped Devices

- Impossible to recover data from factory reset devices
- Local or iCloud backups might be the only source

Modern Devices

- For iPhone 13/14/15/16 devices powered off/battery depleted, at the moment, the only hope is obtaining the passcode from another device of the same owner

Full File System Extraction Tools

FFS extraction is very complex without commercial tools. Historically, jailbreaking techniques were available:

- **checkm8:** usable with checkra1n and palera1n on compatible devices
- **Dopamine:** last jailbreak on a wide range of devices (iOS 15-16.5.1)
- No iPhone/iPad device with iOS 26 is vulnerable to checkm8

No public jailbreak available for post-iPhone X devices with iOS 17, 18, or later versions. Only commercial tools support FFS extractions.

Conclusions and Best Practices

Conclusions are analogous to those for Android devices: there are no perfect guidelines or absolute models, but fundamental guiding principles:

Core Principles

1. **Pre-shutdown evaluation:** always think before powering off an active phone, even with full access. Some data might be irretrievably lost.

2. **Volatility order:** acquire data as soon as possible according to their volatility order. Some data disappears automatically over time.

3. **BFU acquisition utility:** when the device is powered off with an unknown passcode, BFU is always a good starting point for profiling and possible cracking hints.

4. **Technique complementarity:** even with full access and FFS tools, always evaluate extraction with logical tools for data whose interpretation is simpler (e.g., Unified Logs).

5. **Procedural flexibility:** always adapt and re-evaluate the workflow on a case-by-case basis, using available tools and skills.
