From sync to seizure: A binary instrumentation-based evaluation of the iCloud backup process

By:
Julian Geus, Jan Gruber, Jonas Wozar, Felix Freiling

From the proceedings of
The Digital Forensic Research Conference
DFRWS APAC 2025
Nov 10-12, 2025

Forensic Science International: Digital Investigation 54 (2025) 301978

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

DFRWS APAC 2025 - Selected Papers from the 5th Annual Digital Forensics Research Conference APAC

From sync to seizure: A binary instrumentation-based evaluation of the iCloud backup process

Julian Geus *, Jan Gruber, Jonas Wozar, Felix Freiling

Department of Computer Science, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Erlangen, Germany

ARTICLE INFO

Keywords:
Mobile forensics
Cloud forensics
Cloud acquisition

ABSTRACT

Mobile phone data is crucial for gathering investigative leads and solving cases in most criminal investigations. An increasingly common method for collecting mobile data as evidence is acquiring phone backups stored in manufacturer cloud services. However, the reliability of this evidence source compared to the original device has yet to be thoroughly assessed. In this work, we investigate the accuracy and completeness of iOS backups stored in iCloud. We propose a novel evaluation methodology based on dynamic binary instrumentation, enabling precise tracking of backup contents during the restore process. Using this approach, we compare a full file system extraction and a local backup of an iOS device to a backup downloaded from iCloud and restored on a test device. Our analysis reveals significant discrepancies in timestamp information and minor differences in user data—both critical considerations when analyzing iOS backups in criminal investigations.

1. Introduction

Mobile device data is now highly important in investigations of many kinds to such a degree that some even consider it "today's equivalent to yesterday's DNA evidence" (Reiber, 2016, p. 7). However, investigators often face challenges accessing smartphone data due to hardware damage, theft, or security barriers.

Since smartphones are usually integrated into a cloud software ecosystem that is used to backup and synchronize data between devices, evidence acquisition from the cloud is an increasingly promising approach (Roussev and McCulley, 2016). If the target has activated the cloud backup and synchronization functions offered by Google or Apple, it opens up the possibility of obtaining backed-up data from their smartphone. This data set could in fact contain a copy of the desired information, and therefore, we require reliable methods that enable its extraction. Just as important, however, is knowledge of the reliability of the acquisition process and the acquired evidence, i.e., its authenticity, accuracy, and completeness, which are critical attributes for remotely acquired digital evidence (Sommer, 1997, p. 139). Surprisingly, the reliability of cloud-acquired evidence and methods for the assessment, to rule out contamination effects (Gruber et al., 2023), are poorly investigated.

In the present work, we investigate the scope and volume of device backups of Apple iPhones in iCloud. While the official information provided by Apple (Apple, 2025j,h) suggests that local backups and cloud backups comprise a similar, or equal set of data, the accuracy and completeness of restored iCloud backups in comparison with local backups from the originating device have never been assessed.

As illustrated in Fig. 1, there are multiple ways to access data stored in the cloud. The first is to directly access iCloud data using the publicly known API or the web interface of iCloud (Ⓐ in Fig. 1). However, as discussed below, this enables access only to data that is regularly synchronized by Apple's own applications (e.g., Photos, Calendar, Contacts). While there must exist an API to access further data such as phone backups or app folders by restore software on an iDevice, this API is not public. The alternative therefore is to use a technique we call *cloud backup restore* in which we restore a cloud backup onto an *intermediate device* and extract the data by using a local backup (Ⓑ in Fig. 1). If we are able to gain privileged access on the intermediate device we can directly access its full file system after the cloud restore (Ⓒ in Fig. 1).

In this paper, we seek to determine the accuracy and completeness of cloud backup restores. In general, we want to answer the following questions about the scope and the quality of the data extracted:

- How much data can be extracted using the cloud backup restore process?
- How does the scope compare to a local backup from the originating phone?

* Corresponding author.
E-mail addresses: julian.geus@fau.de (J. Geus), jan.gruber@fau.de (J. Gruber), jonas.jw.wozar@fau.de (J. Wozar), felix.freiling@fau.de (F. Freiling).

https://doi.org/10.1016/j.fsidi.2025.301978

Available online 3 November 2025

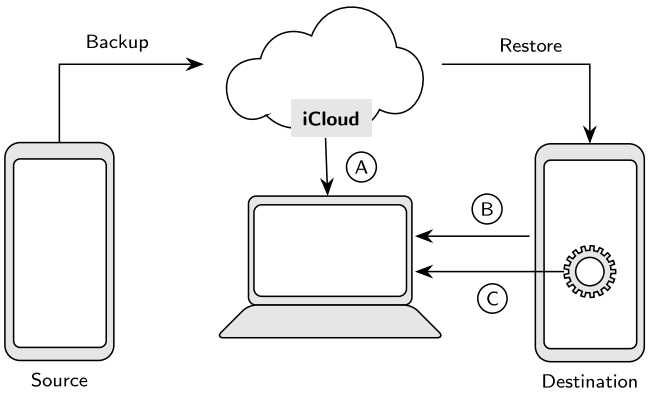

Fig. 1. Access points for forensic data extraction of private iCloud data. Access point *A* represents direct API acquisition, *B* a local backup from a cloud restore device, and *C* a full file system copy from this device.

- Is the extracted data altered during the extraction process?

1.1. Related work

A quality assessment of cloud data acquisition was carried out by Oestreicher (2014), who analyzed the data synchronization feature from iCloud on MacOS for several applications and concluded that, while the contents are identical, the hash values of some of the acquired data did not match. Additionally, Geus et al. (2023) analyzed the forensic soundness of local backups of Android and iOS devices. They concluded that this method is certainly well-suited for forensic analyses with a minor caveat related to the merging of uncommitted database changes, i.e., the application of the write-ahead logging of SQLite databases during the backup creation process.

For evidence collection, Roussev et al. (2016) proposed an API-based cloud acquisition. For specific applications related to vehicle forensics, Ebbers et al. (2024) used this concept to acquire data from cloud services of car manufacturers. Similarly, Hilgert et al. (2021) analyzed the API and developed an acquisition framework for cloud services of micro-mobility solution apps (e.g., ebikes and escooters). However, this procedure is only partially applicable to mobile device forensics since a major part of smartphone data in the cloud is not accessible by a public API, but only by a smartphone using proprietary interfaces.

From a more practical perspective, there are a few blog posts from vendors of forensic tools. Oftentimes, their descriptions deliver valuable information and technical insights, even though they are focused on marketing their solutions. While the articles by Katalov (2020) and Afonin (2022) contain background information about iCloud acquisition and an interesting comparison of the scope compared to local backups, others are mostly focused on the usage of their tool (Magnet Forensics, 2022; Belkasoft, 2025). These examples, however, underpin the practical importance of iCloud backups as a source of evidence.

1.2. Contributions

We survey existing methods of device data acquisition in Apple's iCloud, which has, to the best of our knowledge, not been analyzed from a forensic perspective yet. To improve the reliability of cloud-acquired device data, we develop a novel approach to utilize binary instrumentation for the evaluation of cloud data acquisition processes. We then evaluate the reliability of the cloud backup restore mechanism of iOS by these means and determine the accuracy and completeness of iCloud backups in comparison to the data of the source device. In particular, we make the following contributions:

1. We provide an overview and practical guidelines for forensic data acquisition from Apple's iCloud ecosystem.
2. We propose a novel evaluation method of iCloud data acquisitions leveraging binary instrumentation to observe the inner workings of the processes responsible for downloading and restoring the cloud backup.
3. We evaluate the forensic soundness of the cloud backup restore mechanism over iCloud on iOS and assess the completeness of local backups and restored cloud backups in comparison to the original device data.

Using our evaluation methodology, we are able to determine that the scope of an iCloud backup is smaller compared to a local backup. Thus, data is lost during the cloud backup restore. Furthermore, we were able to show the scope of data alterations and could attribute most of these to concurrent system behaviour and changed metadata due to the data restore onto a different device. However, even though data alterations are quite extensive, most parts of the forensically relevant traces are kept intact.

Due to privacy concerns that are caused by the involvement of multiple parties and the immense quantity of potential traces, the data set is not publicly available but can be obtained from the authors upon request.

2. Background: Apple's backup ecosystem

iOS offers two main backup options: *iCloud* (enabled by default if an account is linked) and *local backups* via USB onto a connected computer. Both methods are similar in scope and generally store most user data, excluding content already synced to iCloud or other cloud services, as well as some device settings. Third-party app data is usually included, except for temporary files (e.g., `tmp/`, `Caches/`), though developers can opt out of backups using the `isExcludedFromBackup` flag (Apple, 2025e). The details and differences of both backup options are further described in the following.

2.1. Local backups

iOS devices can be backed up locally via macOS, iTunes on Windows, or the cross-platform open-source library *libimobiledevice*, which offers greater control and scriptability—making it especially suitable for forensic purposes.

Backups may be encrypted or unencrypted. Encrypted backups require a user-defined password and are more forensically valuable, as they include sensitive data such as health records and stored passwords. If a backup is encrypted, decryption can be achieved by using the information from the included `Manifest.plist` file combined with the user-defined password. There are multiple open-source projects available for parsing and decrypting local backups. We use the Python library *pyiosbackup* (Perelman, 2024) since it is easy to install and still actively supported.

2.2. Cloud backups

iOS devices support data synchronization and device backups into a data store associated with an iCloud account by default. Users can enable continuous cloud synchronization for various data categories (e.g., images, notes) to ensure they remain updated across devices (Apple, 2025h). Furthermore, iCloud can be used as a backup location, which eases the transition to a new device and keeps unsynchronized data safe. Data that is already synchronized will be omitted from device backups (Apple, 2025j). Since multiple devices can be connected to the same iCloud account, one such account can host backups from different devices, which can be chosen on a restore operation. Apple provides the *CloudKit* (Apple, 2025b) developer API as a way for app developers to store and synchronize private or public app data in an app-specific

container of the iCloud account. The stored data cannot only be accessed by an iOS device through synchronization or backup features but also—at least partially—on iCloud's web interface,[1] by using the integrated iCloud feature in MacOS or iCloud for Windows. These options serve as an easy extraction point for specific categories of the stored data. However, a forensically relevant volume of the data, like cloud backups, is not accessible by those means. To access an iCloud account, the authentication by using a second factor is usually required besides the user's credentials. *Two-factor authentication (*2FA*)* is activated by default on all Apple accounts if the requirements are met and is also mandatory to use security-sensitive Apple services like Apple Pay (Apple, 2025g).

2.3. Acquisition options

Given the tight integration and widespread use of cloud services as the preferred backup location for user data on smartphones, the data stored there is of utmost importance in the field of digital forensic science. However, the acquisition also yields some major challenges: As mentioned above, the data is always protected by the user's login credentials, which is oftentimes accompanied by 2FA. Still, even if we gain access to a desired account, its data is not simply accessible. For example, device backups, which may have special importance in an investigation, can only be restored onto a device, on which they are protected by the device's security measures. Therefore, we categorize cloud data into two distinct classes: *Accessible data*, which is easily extractable given account access, and *private data*, which can only be accessed by an iOS device or a specific app.

We illustrated the separation of these two data categories in Fig. 2. An examiner who wants to access cloud data has different options. They can utilize a phone connected to the account to access both, accessible data ① (e.g., photos, reminders, etc.) and private data ② (e.g., a device backup), indirectly. This data has to be restored or synchronized onto a device from which it can be extracted subsequenty—considering access to the device's data is possible. The direct acquisition of the data onto an examiner's computer poses a greater challenge. Accessible data can be directly downloaded ③, given access to the account (e.g., by using a public API or the web interface). Private data ④, however, is not accessible that easily. We analyze these options in greater detail in the following subsections. We focus on publicly available acquisition options while omitting commercial tools since we cannot trace the methods used for data acquisition due to their black-box nature. Thus, we cannot make statements about their forensic soundness.

2.4. Acquisition of accessible data

Accessible data on iCloud comprises of parts of the continuously synchronized data from Apple's own applications (e.g., Photos, Calendar, Contacts). The synchronization of all data categories is activated by default, but the user can choose to opt out. Third-party app containers are also synchronized but not easily accessible. Accessible data includes a good amount of forensically relevant information (see Section 2.2) and there are several options to extract it. These range from using the official iCloud clients over unofficial open-source tools (Evans, 2024; Conrad, 2023) to Apple's user data requests. All have their merits and downsides. Generally, this kind of data is more easy to recover and does not provide the full picture. Therefore, we focus on the acquisition of private data.

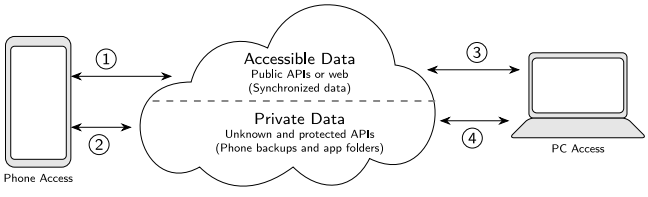

Fig. 2. Separation of cloud data and their access methods using a phone or a PC. Using a phone, both accessible and private data can be obtained. With a PC, official interfaces are restricted to accessible data.

2.5. Acquisition of private data

On iCloud, private data includes device backups, which can only be accessed by restoring the data back onto a device, as well as app containers that are accessible solely through their respective applications. This data might be crucial for forensic investigations, since it could contain valuable information such as third-party app data (e.g., of messenger apps). As described in Section 2.2, the scope of the data depends on the synchronization and backup settings. Therefore, data that is not synchronized and hence inaccessible by the means discussed before might still be accessible through an iCloud device backup. In this work, we mainly focus on device backups, since third party app containers are application specific and cannot easily be accessed or restored. However, these apps might download their data from their iCloud container onto the device, from where we are able to access it with the methods discussed below. This heavily depends on the application's data handling.

2.5.1. Option 1: API acquisition

Direct API access for extracting private cloud data is forensically sound due to minimal data processing. However, unlike the option to obtain accessible data, access to the cloud backup API is officially only available on an iOS device. Additionally, since iPhones must be registered with Apple, there may be access control measures or other hardware bound security mechanisms associated with using these APIs.

Consequently, direct utilization of the API would necessitate extensive reverse engineering, and Apple retains the ability to change their iCloud communication protocol anytime, which could render such efforts futile. In the past, publicly available tools reimplemented those communication protocols and enabled the acquisition of iCloud backups (HackApp, 2016; Horrorho, 2018); however, they ceased to function due to changes made by Apple. As discussed in the corresponding repository issues, community efforts to update the projects apparently came to a halt or turned out to be too demanding. As a result, we currently deem the efforts of acquiring the backup data directly through the API as unfeasible for our work.

2.5.2. Option 2: Intermediate device with unprivileged access

The general idea of using an intermediate device for gaining access to the backup data is to leverage the already existing implementation for accessing the cloud backups from Apple themselves; employed on their devices. This negates the need for manually reverse-engineering the process and also ensures that the implementation works correctly. This way, we can acquire the cloud backup by restoring it to an iPhone. However, backups can only be restored to the same or newer iOS versions and the devices storage capacity must be sufficient. Since no publicly available exploits for current iPhone models are available, a full file system acquisition cannot be executed that easily. Still, the local backup mechanism allows examiners to indirectly acquire the data, which was originally restored from an iCloud backup. This approach is basically *a cloud backup restore* acquisition. The major advantage of this procedure is its availability and the ease of operation. However, it is based on the assumption that local backups have a similar, or equal set of data.

2.5.3. Option 3: Intermediate device with privileged access

Having a device available that allows for privileged access enables us

[1] See https://www.icloud.com/.

to collect the restored cloud data as a full file system acquisition. It requires so-called jailbreaking, which describes the process of removing restrictions on iOS, thereby granting privileged access. With these privileges all data restored during the process becomes accessible. However, it presents challenges in distinguishing what has really been restored versus what data was previously on the device or was created by the setup process. Additionally, the data may be subject to changes due to the data restore, as well as to the concurrently running processes.

Jailbreaking had a once active community, which changed drastically in the last couple of years. As a result, the development efforts are in sharp decline. The *palera1n* (Palera1n Team, 2024) jailbreak-tool, as one of the most recent ones as of the time of writing, uses the infamous *checkm8* vulnerability. It therefore supports only devices up to the iPhone X because it depends on an unpatchable vulnerability in the BootROM code. In consequence, there are no jailbreak tools for the latest iOS devices available publicly at the present moment.[2] Since iCloud backups need to be restored onto a device that supports the iOS version of the originating device (Apple, 2025f), the unavailability of jailbreaks renders this approach practically futile but it is still helpful for evidence assessments, as we will later show.

2.5.4. Option 4: Government information request

Apple offers a process for governments, law enforcement, and private entities to request stored user information. These requests are assessed for legal validity before approval (Apple, 2025i). For instance, in the first half of 2024, 48 % of device data requests in Germany were approved (Apple, 2025c).

In their *Legal Process Guidelines* (Apple, 2025d), Apple outlines the data categories that can be requested, which also include iCloud device backups. However, accounts protected with *Advanced Data Protection* (Apple, 2025a), mostly store end-to-end encrypted data in iCloud, which cannot be decrypted by Apple.

Compared to the previously discussed options, the data request does not require the account credentials or technical knowledge. It might also be considered the most forensically sound approach, as it neither involves manual access to the account—which could alter the data—nor requires any additional processing steps. However, Apple may reject such a request, and even if they comply, the process might be lengthy. This poses challenges for time-critical investigations or cases where data may be deleted from the account in the interim. Since we have no means to obtain such a data set, we can neither describe nor evaluate the exact process or the resulting data. Depending on the investigation, accessing iCloud data without Apple's support may be preferable, as evidenced by the adoption of such methods in commercial forensic tools (see Section 1.1).

3. Evaluation methodology

Considering the vast amount of *private* data and the inherent difficulties of retrieving it, this paper will concentrate on the possibilities of accessing such data via an iCloud account. The *cloud backup restore acquisition* using an intermediate device with unprivileged access is the practically most relevant method, nevertheless a comparison of the scope, accuracy, and completeness of iCloud backups has not been conducted yet. Thus, the exact scope of the data that can be acquired in this way and its forensic soundness is unknown.

To approach the goals of our evaluation we will now describe the process that enables us to answer those questions. Therefore, we first provide an overview of the general procedure, before we discuss details about the generic evaluation process. This process consists of the device preparation and the comparison procedure, which focuses on both, low-level technical measurements and semantic equality.

[2] Due to hardware limitations iPhone X only supports iOS version 16.

3.1. Evaluation setup

Fig. 3 illustrates our setup, with the source device on the left and the intermediate restoration device on the right. We marked the different "measurement points" we used with characters enclosed in circles. The measurements at Ⓛ and Ⓡ (local backups) do not require special privileges, measuring at the points Ⓕ (full file system extraction) and Ⓘ (binary instrumented extraction) is only accessible on jailbroken phones.

The procedure for the evaluation is as follows: The source device is filled with test data. Afterward, we initiate the cloud backup mechanism, which transfers the phone data into the cloud. If a jailbreak is available, a full file system extraction is conducted (Ⓕ in Fig. 3). In any case, we create a local backup (Ⓛ in Fig. 3). In the next step, the cloud backup is restored to an intermediate device. After the cloud restore completed we create a local backup from the intermediate device in any case (Ⓡ in Fig. 3). With this, we facilitate the evaluation of the cloud backup restore process, described in Section 2.5. We present the comparisons of both local backups from recent iPhones in Section 4. Since we aim to precisely keep track of the data that is downloaded from the cloud that comprises the cloud backup, we will extend this method and consider an earlier measurement point (Ⓘ in Fig. 3) utilizing a jailbroken device and dynamic binary instrumentation in Section 5.

3.2. Evaluation process

3.2.1. Process modeling

If local and cloud backups from iOS would behave exactly identical, regarding both their scope and data handling, we expect to acquire congruent data sets through the cloud backup restore acquisition on the source and destination device. Practically, concurrent processes might interfere, but still, the majority of acquired data should be identical.

As set out above, the goal of our evaluation is to determine how much data overlaps and if there are low-level or even semantic mismatches. To precisely define our calculations, we introduce a symbolic notation: We refer to data acquisition from the source iPhone by $\mathscr{S}_X$ and from the destination iPhone by $\mathscr{D}_X$. The subscript X defines the measurement point, as defined in Fig. 3. A data acquisition is a set of file objects. We refer to a set of files by F and a set of full file names including their paths by N. A file object $f \in F$ is a tuple (n, d) composed of a string $n \in N$ representing the unique *file name* and a byte array d representing its associated *data*. We adopt a record accessor notation where, e.g., $f.n$ is the name of file f.

3.2.2. Dataset creation

We now define the steps necessary for the data set creation. These are

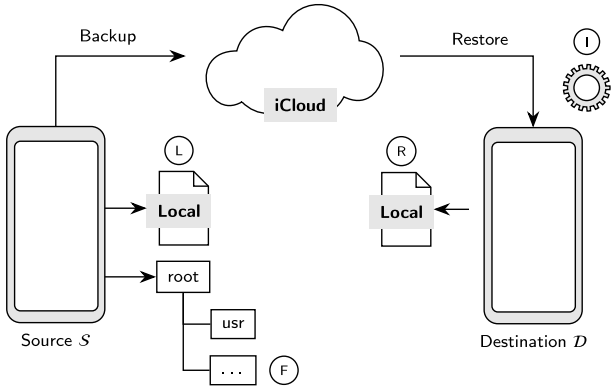

Fig. 3. Overview of data access points for the evaluation process. F denotes the full file system copy, L the creation of a local backup on the source device, I a binary instrumented data acquisition, and R a local backup creation on the destination device.

identical for both device generations we use in our experiments.

On the source device a set of user data is generated to account for an appropriate set of digital traces relevant to a forensic investigation. The destination device is left in a factory new state or is reset to that state, since it is necessary to execute the initial setup process to restore a cloud backup.

On the initial setup of the source device, the default settings were chosen and a newly created iCloud account is registered. We disabled the synchronization of all data in the iCloud settings to maximize the amount of information stored in a device backup. In our scenario, we specifically want to evaluate the data scope and integrity of cloud backups and therefore minimize the influence of synchronization. If, in a practical scenario, data synchronization was activated, these data categories can easily be accessed with one of the options described in Section 2.4. After the setup, a SIM card is inserted into the source device, which is needed to activate some of the chosen applications, and for creating typical communication traces.

The following data is created on the phone:

- *Personal data:* photos, documents, calendar entries, notes, mail account, browser usage, stored passwords
- *Communication data:* contacts, calls, SMS
- *App data:* WhatsApp, Telegram, Signal, TikTok, Instagram, YouTube, Firefox

3.2.3. Dataset comparison

In our evaluation, we compare technical file-based identifiers, such as file names and hashes on the one hand and we look at semantic equality of the data on the other.

3.2.3.1. Technical metrics. Our comparison metrics for the technical comparison are inspired by and partly taken from Geus et al. (2023). Similar to their methodology, we use the function $names(D)$

$$names \subseteq \mathscr{P}(F) \times \mathscr{P}(N)$$

which is defined to retrieve the set of unique file names including their path from a given data set D,

$$names(D) := \{f.n | f \in D\}.$$

We first compare the set of file names from the data extractions $\mathscr{S}_X$ and $\mathscr{D}_Y$ to identify the overlapping files (N_{both}) and determine the files that are only in data set $\mathscr{S}_X$ (N_{Xonly}) or $\mathscr{D}_Y$ (N_{Yonly}):

$$\begin{aligned} N_{both} &:= names(\mathscr{S}_X) \cap names(\mathscr{D}_Y) \\ N_{Xonly} &:= names(\mathscr{S}_X) \backslash names(\mathscr{D}_Y) \\ N_{Y\ only} &:= names(\mathscr{D}_Y) \backslash names(\mathscr{S}_X) \end{aligned}$$

To ensure that the data's integrity is kept intact during the process, we compare all files in N_{both} between $\mathscr{S}_X$ and $\mathscr{D}_Y$ by their content. Intuitively, if the files paths and values equal it is assigned to V_{eq} and if only the paths match to V_{ch}:

$$\begin{aligned} V_{eq} &:= \{f | f \in \mathscr{S}_X \wedge f \in \mathscr{D}_Y\} \\ V_{ch} &:= \{f | (n,d) \in \mathscr{S}_X \wedge (n,d') \in \mathscr{D}_Y \wedge d \neq d'\} \end{aligned}$$

In practice, this is achieved by using hash comparison.

Using these metrics, we can quantify the overlap between the two data sets and see which files are altered during the backup process. Furthermore, these calculations allow us to infer which data comprises the cloud backup.

3.2.3.2. Semantic metrics. Besides the technical metrics, we estimate the semantic equivalence of the compared data sources. In the first part, we compare the file content of SQLite databases, which are highly relevant for mobile forensic analyses. A hash mismatch on such complex file types can have multiple reasons that must not be related to the actual content. We, therefore, determine content equality despite hash mismatches for SQLite databases using a filetype-specific interpretation function I. $I(f)$ interprets binary content $f.d$ of file f according to its type in a meaningful way. This is the validation that all tables (and the data stored in there) are identical. The interpretation function I allows us to define an equivalence relation $\sim$ on the set of files F as follows:

$$f_i \sim f_j \iff I(f_i.d) \subseteq I(f_j.d)$$

Utilizing this content-based equivalence relation, we can determine the set of content-wise matching (or mismatching) files:

$$V_{\sim eq} := \{f | (n,d) \in \mathscr{S}_X \wedge (n,d') \in \mathscr{D}_Y \wedge d \sim d'\}$$

To carry out a more general comparison of forensically relevant traces we use iLEAPP (iOS Logs, Events, And Property Parser) (abrignoni, 2025, v2.1.3) to gather a more meaningful representation of the data sets. iLEAPP is a popular tool that extracts and presents relevant artifacts from iOS backups and full file system acquisitions. The tool enables us to transform the data of the acquisition according to their artifact types (e.g., LocationHistory, SafariHistory, etc.) into tab-separated values (TSV) files and then compare the TSV files using the TSV specific interpretation function I, as described above. In particular, we create a reference iLEAPP report of the data on the source device and check each TSV file whether it is a subset of the corresponding TSV file of the comparison report. If any line from the reference file is not found in the corresponding file of the other report, a diff is generated based on the sorted contents of both files.

A semantic comparison can be arbitrarily complex. Thus, a suitable approximation is necessary, which we believe our approach provides. It is tailored to the specific requirements of forensic investigations, while remaining manageable in terms of scope and complexity.

4. Evaluation of the cloud backup restore process

Now, we present the first concrete experiment using the methodology outlined in the previous section. In particular, we compare data acquisitions gathered at the measurement points Ⓕ and Ⓡ of Fig. 3. We compare a local backup, which has been acquired immediately after creating a cloud backup and a local backup after restoring the iCloud backup.

4.1. Practical setup

For the analysis of the *cloud backup restore* acquisition we use two *iPhone 16e* models. As of the time of writing, those devices are part of the latest generation of iPhones and we also updated both to the latest iOS version (*18.4.1*). As described above, one serves as the source and the other as the destination device. The actual data sets for the evaluation are created as follows:

1. We manually execute a cloud backup on the source device using the settings menu.
2. As reference set we create an encrypted local backup from the source device. This results in the data set $\mathscr{S}_L$.
3. We now restore the cloud backup created in step 1 onto the destination device. During this process all apps previously installed on the source device are automatically installed on the destination phone.
4. After step 3 completed, we create an encrypted local backup of the destination device (data set $\mathscr{D}_R$).

This process was repeated three times to ensure comparability between multiple iterations. Before each run the destination device was reset to factory default. The resulting data sets are presented in the following.

4.2. Results

The results from the file-based comparison can be observed in Table 1. From there, we can determine that the total file count in both backup processes slightly fluctuates. However, this is expected since the devices are in a running state during the data set creation, which leads to data alterations due to concurrent processes. In general, however, the results of all three evaluation runs differ only slightly and are therefore comparable.

By observing $|N_{both}|$ (overlapping files), we see that most files have a counterpart. However, surprisingly some files do not. By analyzing those files we account these observations, again, to concurrent system behaviour. Interestingly, at least parts of the non overlapping files are always the same in each execution. This might be due to some differences in the devices states, or caused by the fact, that the destination phone is always reset shortly before each acquisition, while the source device was running for quite some time.

Most interestingly, however, is the value comparison. From $|V_{eq}|$ (value and paths equal) and $|V_{ch}|$ (path match only), we can deduce that only roughly two thirds of all file pairs with matching names are actually identical. This leads to the assumption that there are quite substantial differences between the local backup and the cloud backup process, which needs to be investigated.

By not only focusing on a file's hash, but also on its content, we observe that from all SQLite databases with a hash-mismatch, around 70 % actually are equal in their content. This is especially interesting, since SQLite databases oftentimes store relevant traces for forensic purposes. This observation somewhat attenuates the result of the value-based analysis.

The comparison of the TSV files generated by iLEAPP from the local backups of the devices showed that in two out of three runs 30 and in one run 31 TSV files of this report were fully contained in the local backup created after the completion of the iCloud backup restore process. These were mainly related to the phone services, such as SIM and subscriber info, SMS, interactionC call history, as well as core services of the phone, such as the account data, preferences, and configuration, timezone information, control center configuration, application permissions, keyboard usage stats, *Find My iPhone* settings, address book data, and calendar entries. Besides that, browsing data, in the form of recent web searches and Safari bookmarks were retained. For third party application data, WhatsApp messages and contacts are available, while for all other installed applications either iLEAPP has no parsing module or their data was not included in the local backup. We also observed concurrency abnormalities: In one run of the experiment, the *zAsset-Analysis State Modification Date* of one asset listed in `PhotoData-Photos.sqlite` has changed. This timestamp is not related to explicit user interaction, but only to background processes.

Five TSV files in the iLEAPP report from the source data set were missing in $\mathscr{D}_R$. They were related to the connected device information, the step recordings from the Health app and the Safari web search and browser history.

Between 29 and 32 TSV files were not fully contained in the data collected after the iCloud backup restoration. Primarily, these changes were related to backup settings and data usage as well as the expected deviations of the IMEI and IMSI. Additionally, we observed changes to the plist-files related to the `locationd` and `routined` daemons as well as the `com.apple.purplebuddy.plist` file, which contains settings for the setup app.

5. Binary instrumented cloud backup evaluation

Given the discrepancies resulting from our comparisons, there is a need to inspect the backup restoration process in more depth to infer the causes and the extent of the observed differences. Hence, we develop and apply a novel binary instrumentation-based evaluation method of iCloud's backup restore process.

Following the methodology outlined in Section 3, we are also able to acquire data at the measurement points Ⓛ and ⓘ of Fig. 3 now. This means, besides the local backups, we can now use a full file system dump, which has been acquired immediately after creating a cloud backup from the source device and the data, which is downloaded during the iCloud restore process for the evaluation.

5.1. Practical setup

5.1.1. Device selection

For the analysis of the acquisition process we use two similar *iPhone 8* models. The iPhone 8 is—alongside the iPhone X—the last model which is susceptible to the `checkm8` vulnerability and therefore, can be jailbroken independently of the installed iOS version. Since the support for these devices already ended, both are restricted to *iOS 16.7.10*. iOS 16 was originally released in September 2022. Although we use an outdated iOS version in this second experiment, it is reasonable to assume that the iOS's backup and restore mechanisms are still operating in a similar way, as later results indicate. So, we argue that the gained insights are still valid and valuable.

The first iPhone 8 is jailbroken and serves as the source device from which the data in iCloud originates. We use the data set from this device as the ground truth and acquire it as a full file system dump. The other iPhone 8 is also jailbroken and is used as an intermediate device with privileged access to restore the data from the iCloud account. The iCloud data will be acquired indirectly from this phone using the proposed method.

5.1.2. Phone preparation

Both phones need to be set up to allow for privileged access. Therefore, the *palera1n* jailbreak tool is used, which also enables SSH access by default. Due to the jailbreak's limitations no passcode can be configured on either devices. The user data on the source device is created as described in Section 3.2.2. The destination phone requires different handling, since it needs to be in a factory reset state to restore the cloud backup. The jailbreak is reset after a reboot, but all jailbreak files and installed packages even survive a factory reset. The jailbreak just needs to be reapplied to restore its full scope, which can also be done during the initial setup. Therefore, we first jailbreak the phone, then reset it to factory default and reapply the jailbreak in the initial setup screen. For the analysis of this device, we use the binary instrumentation toolkit *Frida*.[3] Frida enables us to inject JavaScript code into processes running on the device. It is installed from *Sileo*, a third-party package manager for jailbroken devices. Using the elevated privileges and Frida, we can acquire all data sets of Fig. 3 for each evaluation run, using the following steps:

1. We manually execute a cloud backup on the source device from the iCloud settings.
2. We then create a full file system copy (data set $\mathscr{S}_F$) and an encrypted local backup (data set $\mathscr{S}_L$).
3. We now restore the cloud backup created in step 1 onto the destination phone and record the processes using Frida (data set $\mathscr{D}_I$). We wait until all apps included in the cloud backup are installed on the device.
4. After step 3 completed, we create an encrypted local backup from the destination phone (data set $\mathscr{D}_R$).

Again, the process is repeated three times to ensure comparability between multiple iterations. Before each run the destination device was reset to factory default and the jailbreak was reapplied.

[3] See https://frida.re.

Table 1
Results of the file-based comparison of the local backups acquired from both iPhone 16e devices.

	Filecount		Name/Value Comparison					Semantic Comparison				
								SQLite		iLEAPP (TSV)		
	$\lvert\mathscr{S}_L\rvert$	$\lvert\mathscr{D}_R\rvert$	$\lvert N_{both}\rvert$	$\lvert N_{Sonly}\rvert$	$\lvert N_{Tonly}\rvert$	$\lvert V_{eq}\rvert$ ($\%N_{both}$)	$\lvert V_{ch}\rvert$ ($\%N_{both}$)	$\lvert V_{ch}\rvert$	$\lvert V_{\sim eq}\rvert$	*full*	*partial*	*missing*
1	1230	1235	1210	20	25	817 (67.5%)	393 (32.5%)	125	87	30	29	5
2	1228	1231	1202	26	29	813 (67.6%)	389 (32.4%)	124	86	30	32	5
3	1234	1228	1208	26	20	816 (67.5%)	392 (32.5%)	126	89	31	31	5
$\bar{\Sigma}$	1231	1231	1207	24	25	816 (67.5%)	391 (32.5%)	125	87	30	31	5

5.2. Results

To gain deeper insight into the internal workings of the restore process and to better understand the discrepancies observed earlier, we developed the binary instrumentation method. This allowed us to identify files originating from the cloud and capture their contents before they were restored to the device, thereby avoiding interference from the operating system or applications.

5.2.1. Reverse engineering iCloud's backup internals

To conduct the black-box analysis, we relied on a combination of two complementary techniques: In addition to binary instrumentation, we utilized iPhone logs and system monitoring utilities.

5.2.1.1. iCloud backup internals. First, we identified the processes related to iCloud's backup restore procedure using `htop` and `ps` by monitoring the activity during the process. Then, we collected unredacted iOS logs (EthanArbuckle, 2023) via the `idevicesyslog` command from *libimobiledevice* to better understand the activity of system services. Lastly, we leveraged the command-line tool `fs_usage`, available by default on Apple devices, to track file system activity and analyze the origin of files.

The orchestration of these tools enabled us to trace the backup restoration process at a technical level: Initially, files are downloaded in chunks by *cloudd* and stored in temporary directories. From there, *backupd* takes over and further processes the files as they pass through various stages and temporary folders. Before final restoration, most of the backup data are collected in `/var/.backup.i/`, where the original file system structure is reconstructed. Subsequently, the data are simply copied to their final destinations in the file system.

Third-party app data, however, follow a different path: after downloading, the data are copied directly from a temporary location with scrambled file names to their corresponding locations in the file system. Once the temporary data—whether of system or third-party origin—are copied, they are immediately deleted. This behavior necessitated the development of a reliable method to capture these data during the restoration process.

5.2.1.2. Binary instrumented iCloud process observation. Insights into the internal workings of the procedure enabled us to develop a Frida script to precisely observe the content and scope of the downloaded cloud backup at measurement point in Fig. 3. In particular, the script operates as follows: Since iCloud-related communication is handled by the daemon process *cloudd*, and subsequent data handling is performed by *backupd*, we hooked into these processes to monitor system calls related to file operations. Each time a file is opened by one of these processes, its file descriptor is stored to track subsequent file operations. When the file descriptor is about to be closed, the process execution is paused at the end of the `close` system call to create a copy of the file. Using this method, we are able to acquire all downloaded backup data in their final state—that is, fully complete with the restored files, their names, and their file system locations. Thus, we consider this data acquisition to be essentially as close as possible to the original data stored in iCloud. In the following, this data set is essential for improving the accuracy of our evaluation procedure. Since it can provide further insights into the origins of the data alterations observed in Section 4.2.

However, there are multiple system calls that can be used to open or move a file. Since there is no iOS equivalent to the Linux utility `strace` for identifying all system calls issued by a process, we used Frida to attach to the process and monitor all executed machine instructions. On every `svc` (supervisor call), we inspected the registers to extract the syscall number and its parameters. This approach allowed us to determine the required function hooks for acquiring all relevant files.

Although we tested our scripts multiple times and are confident that we capture all important aspects, we cannot guarantee the completeness of our procedure due to the black-box approach and the required reboot during the restore process.

5.2.2. Technical metrics

Table 2 shows the results of the comparisons of the local backups (Ⓛ vs. Ⓡ) as well as the comparison of the full file system extraction and the data acquired by binary instrumentation (Ⓕ vs. Ⓘ). Each experiment has been run three times to account for sporadic effects and ensure repeatability.

To begin, we compare the data acquisitions from the local backup mechanism to establish a baseline for comparability with the results for the iPhone 16e devices presented in Section 4.2. The total file count of the backups is smaller compared to these devices. We attribute this difference to their broader functional scope. More interestingly, the number of files in $\mathscr{S}_L$ drastically decreases in the second and third data set, while it is relatively consistent in $\mathscr{D}_R$. This surprising behaviour is caused by TikTok's application data, which is no longer available on the source device after the first measurement. The exact reason for this remains unknown. However, otherwise, we do see a similar trend compared to the newer devices: most files are present in both sets, but a significant portion of them do not match in the hash comparison. The higher percentage of mismatches observed in this evaluation may be due to the smaller data set.

Since we observed similar trends as in Section 4.2, we now dig deeper to identify the root cause of this behaviour. To do so, we compare the data set $\mathscr{S}_F$ with $\mathscr{D}_I$ as the acquisition via binary instrumentation provides a close approximation of the actual cloud backup content. In the destination set $\mathscr{D}_I$, we observe many files without a counterpart in the reference data set $\mathscr{S}_F$. However, this is not surprising, as temporary files created during the backup restore are included in the process interception data set. The significant difference in the amount of matching data during the first run is again caused by the missing files from the TikTok application. Although the app remains installed and unchanged on the source device, its data appears to have vanished from the phone. There is again a significant number of data alterations, but compared directly to the local backup results we observe a significant improvement ($\sim$ 40% vs. $\sim$ 18%). As noted above, most database files exhibiting hash mismatches are actually identical in content, which further increases the similarity between the sets. N_{both}, however,

Table 2
Results of the file-based comparison of the data sets acquired from both iPhone 8 devices.

	Filecount		Name/Value Comparison					Semantic Comparison				
								SQLite		iLEAPP (TSV)		
	Source	Dest.	$\lvert N_{both}\rvert$	$\lvert N_{Sonly}\rvert$	$\lvert N_{Tonly}\rvert$	$\lvert V_{eq}\rvert$ ($\%N_{both}$)	$\lvert V_{ch}\rvert$ ($\%N_{both}$)	$\lvert V_{ch}\rvert$	$\lvert V_{\sim eq}\rvert$	*full*	*partial*	*missing*
	$\lvert\mathscr{S}_L\rvert$	$\lvert\mathscr{D}_R\rvert$	*local backup vs. local backup*									
1	954	945	925	29	20	560 (60.5%)	365 (39.5%)	95	68	28	31	13
2	872	947	843	29	104	499 (59.2%)	344 (40.8%)	85	59	28	31	13
3	872	945	843	29	102	495 (58.7%)	348 (41.3%)	85	58	27	32	13
$\bar{\Sigma}$	899	946	870	29	72	518 (59.5%)	352 (40.5%)	88	62	28	31	13
	$\lvert\mathscr{S}_F\rvert$	$\lvert\mathscr{D}_I\rvert$	*full file system vs. binary instrumentation*									
1	13965	1125	774	13191	351	648 (83.7%)	126 (16.3%)	76	72	32	5	79
2	12897	1125	717	12180	408	596 (83.1%)	121 (16.9%)	68	64	32	7	75
3	12907	1119	704	12203	415	560 (79.5%)	144 (20.5%)	67	58	31	8	76
$\bar{\Sigma}$	13256	1123	732	12525	391	601 (82.1%)	130 (17.9%)	70	65	32	7	77

decreased compared to the local backup comparison. The most plausible explanation is that the cloud backup comprises a smaller data set compared to the local backup. This hypothesis could account for the large number of hash mismatches observed in the local backup comparisons: if certain data are not restored from the cloud backup, new files with different contents may be created in their place.

5.2.3. Semantic metrics

To establish a baseline for comparability with the findings presented in Section 4.2, we begin by analyzing both local backup data sets at a semantic level. Here, we observe 27–28 identical, 31–32 changed and 13 missing TSV files. These numbers closely resembles the results from the iPhone 16e comparison; however, similar to the file-based results, they contain a smaller amount of data. Data categories that are now also completely absent after the cloud backup restore are all types of health data and the call history.

Comparing the full file system acquisition with the data obtained by binary instrumentation, we see that again most SQLite databases exhibiting hash mismatches actually are identical in content, with an even greater overlap than observed in the local backup comparison. The comparison of the TSV files generated by iLEAPP from the full file system extraction of the source device showed that in two out of three runs 32 and in one run 31 TSV files of this report were fully contained in the binary instrumented recording of the iCloud backup restore process. Between five to eight TSV files were not fully contained. Primarily, these changes were related to backup settings and data usage. Otherwise, the contained information closely resembles the already described local backup data set.

Between 75 and 79 TSV files parsed from the source data set were missing in $\mathscr{D}_I$. This high number is not surprising, however, since we now compare the restored backup data against the entire iOS file system, which is naturally much broader in scope. The missing TSV files were related to data from the Health app, the Biome streams (data related Apple's default apps), WiFi related information, and connected devices. Additionally, application data such as call history, Safari browsing history or Telegram messages were not included.

To examine the reliability of our binary instrumented acquisition, we compare the overlapping data of both semantic data comparisons. We would expect that the resulting data set from the $\mathscr{S}_F$ vs. $\mathscr{D}_I$ is a superset of $\mathscr{S}_L$ vs. $\mathscr{D}_R$ and that data alterations are less frequent in the former. Interestingly, the *full* and *partial* TSV files from the binary instrumented comparison are entirely contained in the local backups. However, we encountered 20–22 TSV files that are present only in local backups. This is unexpected, because data in $\mathscr{D}_R$ has to be restored to the phone, and must therefore be contained in $\mathscr{D}_I$. Most of these TSV files (12) are based on photo metadata from the *Photos.sqlite* database. This file, however, is fully present in $\mathscr{D}_I$ and also hash-identical. The same holds true for the cellular information from the *com.apple.commcenter.plist*. Therefore, we suspect an error in iLEAPP's data parsing. Still, we also observed missing WiFi information and interactionC data. In this case, the backing files are also available, but empty. Thus, we did not capture this data in the data set. We suspect this to be related to the unknown internals of the synchronization processes of iOS, where data is not included in the backups, but later synchronized from iCloud. Two other data sets are directly related to iOS's local backup metadata and are therefore only available in those.

6. Discussion and limitations

The results of the experiments provided us with comprehensive insights into iCloud's backup process and several implications for forensic fieldwork. We now discuss the main takeaways and outline the limitations of our approach.

6.1. Implications of the findings

6.1.1. Technical aspects

In the file-based analysis of the cloud backup restore procedure, we observed that a significant portion of the acquired local backup data had different hash values compared to their reference counterparts. There are various possibilities to explain this behavior: Parts of the data might undergo minor changes due to the backup and restore process or the scopes of local and cloud backups might differ and therefore data is lost in the process.

To identify the root cause of these changes, we have undertaken a second evaluation, in which we compared the full file system copy to the downloaded data from the iCloud restore process. In this comparison, we noticed that the data overlap between the sets is smaller than in the comparison of local backups but exhibits a significantly lower percentage of hash mismatches. This result confirms that the cloud backup data set is, in fact, smaller in size. Additionally, it suggests that specific files of the resulting local backup may not stem from the cloud backup.

6.1.2. Semantic aspects

By comparing the contents of the pairs of SQLite databases with a hash mismatch, we get a similar picture in all data comparisons: The majority of these pairs are indeed semantically identical, which indicates a modification of the databases due to the backup process. It is likely to be caused by merging the write-ahead logs to the copy of the backed-up database file, as previously identified by Geus et al. (2023)

for local backups; thus, these data alterations can generally be considered insignificant for most forensic purposes.

Apple states that data already synchronized to iCloud is omitted from cloud backups; therefore, we deactivated synchronization altogether to maximize the included data. Data such as health and browsing information, however, were surprisingly still not present in the resulting data set of the cloud backup restore. This finding suggests that these data categories can only be synchronized with iCloud but are not included in cloud backups. Apple might regard them as especially sensitive since they are not included in unencrypted local backups but only in encrypted ones. However, this data could answer many investigative questions; thus, their acquisition is desirable in forensic investigations. In a practical scenario, they can still be synchronized to the intermediate device via iCloud, provided the owner kept the synchronization activated. In general, we again noticed that the scope of the cloud backup data set is smaller compared to encrypted local backups, and thus conclude with a high amount of certainty that iCloud backups do not fully contain the data of encrypted local backups. The key insight is that essential content information is retained. Nevertheless, we observed that data related to autonomous and concurrently running system services were modified. The lack of understanding of the behavior of these programs hinders the analysis because examiners do not have a sufficient understanding of the relevance and expressiveness of particular facets regarding investigative hypotheses (Gruber and Humml, 2023). In this regard, we also observed timestamp changes due to the backup and restore process that need to be considered. Due to the nature of the process, which essentially creates logical copies of all data objects and accesses them multiple times, we stress that the *timestamps of files* will be altered and should not be trusted in general. However, *timestamps in files* (e.g., in photos or databases) are mostly preserved—we only identified changes due to concurrent system behaviour (see Section 4.2).

In essence, we conclude that this acquisition method is useable for digital forensic investigations; still, we observed that the data's accuracy and completeness is limited, and provenance does not uphold the high standards of forensic soundness.

6.2. Limitations

Our experiments were limited to two iPhone models and two iOS versions. For each model, we conducted only three runs due to the need for manual interaction with the physical devices. Although this allowed us to identify some consistent patterns, we also observed occasional deviations. Furthermore, the scope of the data created on the phones, which comprised the backup was rather confined. While we also examined the semantics of the data, a more in-depth analysis of the iCloud process' impact is necessary. Nonetheless, our findings and, in particular, the proposed evaluation methodology establish a basis for a larger study involving more devices, software versions, and real-world data sets.

7. Conclusion

The use of cloud-acquired evidence in mobile device forensics is gaining importance. However, the scope and volume of data retrieved via backup restore processes have remained largely opaque. To address this, we described data categories and forensic access points within Apple's iCloud ecosystem and evaluated the cloud backup restore mechanism, aiming to assess the accuracy and completeness of restored iCloud backups compared to data on the original device. To gain deeper insights, we proposed a novel binary instrumentation-based evaluation method, which allowed us to precisely track data downloaded from the cloud and restored to the phone. Our analysis showed that iCloud backups are more limited in scope than local backups. Furthermore, although hash values of the data sets differed, their semantic content—that is, content-wise consistency—was largely preserved. Nonetheless, we advise caution when interpreting timing information related to system services, as these varied across our experiments. Overall, we conclude that the iCloud backup restore process can be used for forensic evidence acquisition, but examiners must be aware of the changes introduced during this process. Finally, we identified the need for large-scale future studies utilizing the proposed evaluation method.

CRediT authorship contribution statement

Julian Geus: Conceptualization, Methodology, Investigation, Validation, Visualization, Writing - Original Draft, Writing - Review and Editing, Supervision. **Jan Gruber:** Methodology, Investigation, Visualization, Writing - Original Draft, Writing - Review and Editing. **Jonas Wozar:** Investigation, Writing - Review and Editing. **Felix Freiling:** Conceptualization, Funding Acquisition, Methodology, Supervision, Writing - Review & Editing.

Acknowledgements

We wish to thank our shepherd, Christopher Hargreaves, and the reviewers for their valuable feedback. Special thanks goes to the FAUmac-Team for providing us with iPhones for our research, and to our colleagues, Maximilian Eichhorn and Jenny Ottmann, for their helpful remarks. Work was supported by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) as part of the Research and Training Group 2475 "Cybercrime and Forensic Computing" (grant number 393541319/GRK2475/2-2024).

References

Abrignoni, 2025. iOS Logs, Events, And Plists Parser. URL: https://github.com/abrignoni/iLEAPP. (Accessed 23 July 2025).

Afonin, O., 2022. Cloud Forensics: Obtaining iCloud Backups, Media Files and Synchronized Data. URL: https://blog.elcomsoft.com/2022/11/cloud-forensics-obtaining-icloud-backups-media-files-and-synchronized-data/. (Accessed 23 July 2025).

Apple, 2025a. Advanced Data Protection for iCloud. URL: https://support.apple.com/guide/security/advanced-data-protection-for-icloud-sec973254c5f/web. (Accessed 21 July 2025).

Apple, 2025b. Designing apps using CloudKit. URL: https://developer.apple.com/icloud/cloudkit/designing/. (Accessed 15 April 2025).

Apple, 2025c. Germany - Transparency Report. URL: https://www.apple.com/legal/transparency/de.html. (Accessed 21 July 2025).

Apple, 2025d. Legal Process Guidelines. URL: https://www.apple.com/legal/privacy/law-enforcement-guidelines-outside-us.pdf. (Accessed 21 July 2025).

Apple, 2025e. Optimizing Your App's Data for iCloud Backup. URL: https://developer.apple.com/documentation/foundation/optimizing-your-app-s-data-for-icloud-backup. (Accessed 15 April 2025).

Apple, 2025f. Restore an iPhone, iPad, or iPod touch that needs a newer version of iOS or iPadOS. URL: https://support.apple.com/en-us/108811. (Accessed 17 April 2025).

Apple, 2025g. Two-factor authentication for Apple Account. URL: https://support.apple.com/en-us/102660. (Accessed 15 April 2025).

Apple, 2025h. Use iCloud on iPhone. URL: https://support.apple.com/guide/iphone/use-icloud-iphde0f868fd/ios. (Accessed 15 April 2025).

Apple, 2025i. We believe security shouldn't come at the expense of individual privacy. URL: https://www.apple.com/privacy/government-information-requests/. (Accessed 21 July 2025).

Apple, 2025j. What does iCloud back up? URL: https://support.apple.com/en-us/108770. (Accessed 15 April 2025).

Belkasoft, 2025. iCloud acquisition and analysis with Belkasoft X. URL: https://belkasoft.com/icloud-forensics-with-belkasoft-x. (Accessed 23 July 2025).

Conrad, E.M., 2023. iCloud API. URL: https://github.com/ElyaConrad/iCloud-API. (Accessed 17 April 2025). Commit: 7edb504.

Ebbers, S., Gense, S., Bakkouch, M., Freiling, F.C., Schinzel, S., 2024. Grand theft API: A forensic analysis of vehicle cloud data. Forensic Sci. Int. Digit. Investig. 48, 301691. https://doi.org/10.1016/j.fsidi.2023.301691.

EthanArbuckle, 2023. Unredacting <private> os_log( ) messages on iOS. URL: https://github.com/EthanArbuckle/unredact-private-os_logs. (Accessed 23 July 2025). Commit: 544cc65.

Evans, P., 2024. pyiCloud. URL: https://github.com/picklepete/pyicloud. (Accessed 17 April 2025). Commit: 622cd16.

Geus, J., Ottmann, J., Freiling, F., 2023. Systematic evaluation of forensic data acquisition using smartphone local backup. In: Proceedings of the Digital Forensics Research Conference USA (DFRWS USA) 2023.

Gruber, J., Hargreaves, C.J., Freiling, F.C., 2023. Contamination of digital evidence: Understanding an underexposed risk. Forensic Sci. Int. Digit. Investig. 44, 301501. https://doi.org/10.1016/j.fsidi.2023.301501.

Gruber, J., Humml, M., 2023. A formal treatment of expressiveness and relevance of digital evidence. Digital Threats. URL. https://dl.acm.org/doi/10.1145/3608485.
HackApp, 2016. iLoot. URL: https://github.com/hackappcom/iloot. (Accessed 20 April 2025). Commit: 9362ac9.
Hilgert, J.N., Lambertz, M., Mateyna, A.M., Hakoupian, A., 2021. A forensic analysis of micromobility solutions. Forensic Sci. Int.: Digit. Invest. 38, 301137.
Horrorho, 2018. InflatableDonkey. URL: https://github.com/horrorho/InflatableDonkey. (Accessed 20 April 2025).
Katalov, V., 2020. iOS acquisition methods compared: logical, full file system and iCloud. URL: https://blog.elcomsoft.com/2020/04/ios-acquisition-methods-compared-logical-full-file-system-and-icloud/. (Accessed 31 July 2025).
Magnet Forensics, 2022. Forensic Analysis of iCloud Backups up to iOS15. URL: https://www.magnetforensics.com/blog/forensic-analysis-of-icloud-backups-up-to-ios15/. (Accessed 23 July 2025).
Oestreicher, K., 2014. A forensically robust method for acquisition of icloud data. Digit. Invest. 11, S106–S113. https://doi.org/10.1016/j.diin.2014.05.006. URL: https://www.sciencedirect.com/science/article/pii/S1742287614000498. fourteenth Annual DFRWS Conference.
Palera1n Team, 2024. palera1n - Jailbreak for iPhone, iPad, Macbooks, and AppleTV's for versions 15 and higher. URL: https://palera.in/. (Accessed 23 July 2025).
Perelman, M., 2024. A python parser for iOS backups. URL: https://github.com/matan1008/pyiosbackup. (Accessed 17 April 2025). Commit: 09dfb26.
Reiber, L., 2016. Mobile Forensic Investigations: A Guide to Evidence Collection, Analysis, and Presentation, first ed. McGraw-Hill Education Group.
Roussev, V., Barreto, A., Ahmed, I., 2016. Api-based forensic acquisition of cloud drives. In: Peterson, G.L., Shenoi, S. (Eds.), Advances in Digital Forensics XII - 12th IFIP WG 11.9 International Conference, New Delhi, India, January 4-6, 2016, Revised Selected Papers. Springer, pp. 213–235. https://doi.org/10.1007/978-3-319-46279-0_11.
Roussev, V., McCulley, S., 2016. Forensic analysis of cloud-native artifacts. Digit. Invest. 16 (Suppl. ment), S104–S113. https://doi.org/10.1016/j.diin.2016.01.013.
Sommer, P., 1997. Downloads, logs and captures: evidence from cyberspace. J. Financ. Crime 138–151.