Naif Arab University for Security Sciences
Arab Journal of Forensic Sciences and Forensic Medicine
المجلة العربية لعلوم الأدلة الجنائية والطب الشرعي
https://journals.nauss.edu.sa/index.php/AJFSFM

Forensic Approaches for End-to-End Encryption Cloud Storage Services: MEGA as a Case Study

منهجيات للأدلة الجنائية للتعامل مع خدمات التخزين السحابي التي تستخدم التشفير الشامل: MEGA كدراسة حالة

Jeongyoon Kang[1], Jieon Kim[1], Seokhee Lee[2], Jungheum Park[1]*

[1]School of Cybersecurity, Korea University, 145 Anam-Ro, Seongbuk-Gu, Seoul, Republic of Korea.
[2]Center for Cybercrimes and Digital Forensics, Department of Forensic Sciences, Naif Arab University for Security Sciences, Riyadh, Saudi Arabia.

Received 05 Dec. 2023; Accepted 24 Mar. 2024; Available Online 18 June 2024.

Abstract

The advancement of cloud-based data storage technology allows users to conveniently access and manage files using end-point devices without being constrained by their environment. While cloud storage services have improved the efficiency of performing our daily tasks, they have also become a medium for criminals to distribute illegal materials. Services that support end-to-end encryption (E2EE), cannot decrypt data even when it's stored on their servers, attracting users who require high security. There are some existing studies related to cloud-based services using E2EE, but they only deal with local artifacts, which makes it difficult to analyze when local devices cannot be found or when there are changes to local artifacts. This study identifies the mechanisms by which MEGA, a cloud-based file hosting service, operates to obtain user authentication, explore metadata, and collect files while applying end-to-end encryption. Furthermore, we propose a forensic investigation methodology to explore various metadata and selectively acquire cloud resources relevant to an incident through an understanding of E2EE algorithms. Also, we apply MEGA to the existing framework to suggest improving the framework that encompasses E2EE cloud-based services. The findings of this study serve as a valuable reference for dealing with cloud-based services with E2EE from the perspectives of computer security and digital forensics.

Keywords: Forensic sciences, digital forensics, cloud forensics, end-to-end encryption, MEGA.

المستخلص

يتيح التقدم في تكنولوجيا تخزين البيانات السحابية للمستخدمين الوصول بسهولة إلى الملفات وإدارتها باستخدام أجهزة نقطة النهاية دون التقيد ببيئتهم. في حين أن خدمات التخزين السحابية حسنت كفاءة أداء مهامنا اليومية، فقد أصبحت أيضًا وسيلة للمجرمين لتوزيع المواد غير القانونية. لا تستطيع الخدمات التي تدعم التشفير الشامل (E2EE)، فك تشفير البيانات حتى عندما يتم تخزينها على خوادمها؛ مما يجذب المستخدمين الذين يحتاجون إلى أمان عالٍ. وهناك بعض الدراسات الموجودة المتعلقة بالخدمات السحابية باستخدام E2EE، لكنها تتعامل فقط مع المصنوعات المحلية؛ مما يجعل من الصعب التحليل عندما لا يمكن العثور على الأجهزة المحلية، أو عندما تكون هناك تغييرات في المصنوعات المحلية. وتحدد هذه الدراسة الآليات التي تعمل من خلالها شركة MEGA، وهي خدمة استضافة الملفات السحابية، للحصول على مصادقة المستخدم واستكشاف البيانات الوصفية وجمع الملفات أثناء تطبيق التشفير الشامل. علاوة على ذلك، نقترح منهجية التحقيق الجنائي لاستكشاف البيانات الوصفية المختلفة والحصول بشكل انتقائي على الموارد السحابية ذات الصلة بحادث ما من خلال فهم خوارزميات E2EE. كما أننا نطبق MEGA على الإطار الحالي لاقتراح تحسين الإطار الذي يشمل الخدمات المستندة إلى السحابة E2EE. تعتبر نتائج هذه الدراسة بمثابة مرجع قيم للتعامل مع الخدمات السحابية مع E2EE من منظور أمن الكمبيوتر والطب الشرعي الرقمي.

الكلمات المفتاحية: علوم الأدلة الجنائية، الأدلة الجنائية الرقمية، الأدلة الجنائية السحابية، التشفير الشامل، خدمة استضافة الملفات السحابية MEGA.

Production and hosting by NAUSS

* Corresponding Author: Jungheum Park
Email: jungheumpark@korea.ac.kr

doi: 10.26735/TMPV1812

1. Introduction

With the recent advancements in information and communication technology, the proliferation of personal digital devices such as smartphones has led to the widespread adoption of storing and managing data in cloud-based storage. This cloud storage service, commonly referred to as Storage as a Service (STaaS), is experiencing continuous growth in the relevant market. TechNavio forecasts the global cloud storage service market to grow from $29.79 billion in 2019 to an estimated $103 billion by 2024, with a CAGR of 28.16% [1].

The growth of the cloud storage market can be interpreted as an increase in demand for data synchronization and accessibility among individual-owned terminals, which means that data storage locations have expanded from local storage devices (such as hard disk drives and SD cards) to cloud servers. In particular, cloud storage service providers also offer free storage of tens of gigabytes for free. As a result, the importance of cloud-based storage is increasing as a potential source of digital evidence related to the suspect's behavior during an investigation.

Cloud storage services generally provide features that enable easy sharing of content with specific or multiple users, and due to these characteristics, they have frequently been utilized as a medium for sharing crime-related content [2, 3, 4, 5, 6, 7, 8]. In cases where cloud storage services are used for criminal activities, the functionalities offered by each service, communication mechanisms between clients and servers and content encryption mechanisms, can significantly impact forensic activities. Accordingly, research from a security and digital forensics perspective has been actively conducted on services such as Google Cloud, MS One Drive, and Dropbox, which occupy a high share in the STaaS market. However, research on services applied with end-to-end encryption techniques in the process of user authentication and data access has been conducted on a limited basis.

Among various STaaS providers, the MEGA, which supports end-to-end encryption and offers larger free storage space than other services, is widely used for criminal activities. Many researchers studied other cloud-based services such as Google Drive or OneDrive, but there are insufficient studies conducted on the MEGA. In November 2019, criminals of the infamous ‘Nth Room’ case in Korea distributed numerous child pornographic materials through MEGA's link-sharing [9]. While MEGA implemented policies to prevent link sharing of illegal video files by comparing their hash values with those uploaded files, it is merely a post-response measure limited to the known files.

In addition, it can be seen that creating and easily distributing web URIs that can share illegal content using 'Link' functions through various reported events [2, 4, 5, 7] is a way to use MEGA Cloud for cybercrime. This is a function that makes it easy to download content even for non-member users who do not subscribe to MEGA Cloud, and it is important for digital forensic investigators to identify various types of metadata provided by MEGA Cloud to find the first distributor of illegal content circulating at a rapid pace and analyze events. In other words, it can be seen that it is necessary to identify the encryption and decryption algorithm of MEGA Cloud, which has not been studied much in the past, and to obtain various metadata.

This study proposes a method of systematically collecting data stored on remote servers through software whitepapers and SDK analysis provided by MEGA, assuming that investigators obtained user credentials or recovered cached auto-login credentials. The findings could help digital forensic experts to analyze encrypted cloud-based services since existing studies are mainly conducted on cloud-based services that do not support data encryption.

The contributions of this study are as follows:

- This study comprehensively examines the data collection capabilities and scope on MEGA required by the recently proposed framework for data collection in cloud storage, suggesting specific methods to support the framework, including authentication, discovery, filtering, and collection.
- This study identifies keys used for web API requests and responses through source code debugging and packet monitoring to collect meaningful data from a digital forensic perspective.
- This study discovers processes of decrypting E2E data to collect all available data from MEGA servers. Based on our findings, we have also developed tool that can be used to collect encrypted cloud resources.

2. Background and related works

In this section, we describe features provided by MEGA including E2EE and introduce previous studies on several cloud storage services.

2.1. Background

2.1.1. MEGA

MEGA is an open-source cloud storage service provided by Mega Limited, based in New Zealand [10]. MEGA users can access their cloud resources using a web browser or dedicated (desktop, mobile) applications. As a service that offers end-to-end encryption, only encrypted data is stored on the cloud servers, making it impossible for the service provider to decrypt the uploaded data without a user's encryption key due to the system design [11]. Consequently, MEGA ensures its high level of security for personal data, that has been utilized to distribute illegal contents [2, 3, 4, 5, 6, 7, 8].

In addition to its basic file storage functionality, MEGA offers additional features compared to other services. Users can upload files with the same name but different content to manage versions of the files. They can also engage in personal or group chats, make voice and video calls, and participate in video conferences. Furthermore, MEGA provides various sharing capabilities, including 'Incoming share', 'Outgoing share', 'Link', and 'File Request (a file upload request functionality that can also be used by non-MEGA users)'.

2.1.2. End-to-End Encryption

End-to-end encryption (E2EE) is a technology where a user sends an encrypted message, file, audio, or video stream on their device, and the other user receives and decrypts data on their devices using encryption key known only to the endpoints. This ensures that users third-party, including MEGA, cannot access the content of the encrypted data. E2EE guarantees the privacy and security of all communication, safeguarding the confidentiality and integrity of the exchanged data [12].

If the cloud storage service does not provide E2EE, even with TLS (Transport Layer Security) communication, the encrypted data is decrypted as it passes through the server and re-encrypted just before it is received Figure-1a. As a result, cloud service providers can access the decrypted data. Without E2EE, it is also vulnerable to man-in-the-middle attacks, which means that someone can decrypt and leak the data you upload to the cloud.

MEGA supports E2EE that when a user logs into their account, the password value is used for generating a one-way hash and encrypting data as well. The encrypted data then will be stored on cloud servers Figure-1b. This data protection mechanism ensures that only authorized users can access cloud resources and the service provider cannot view the

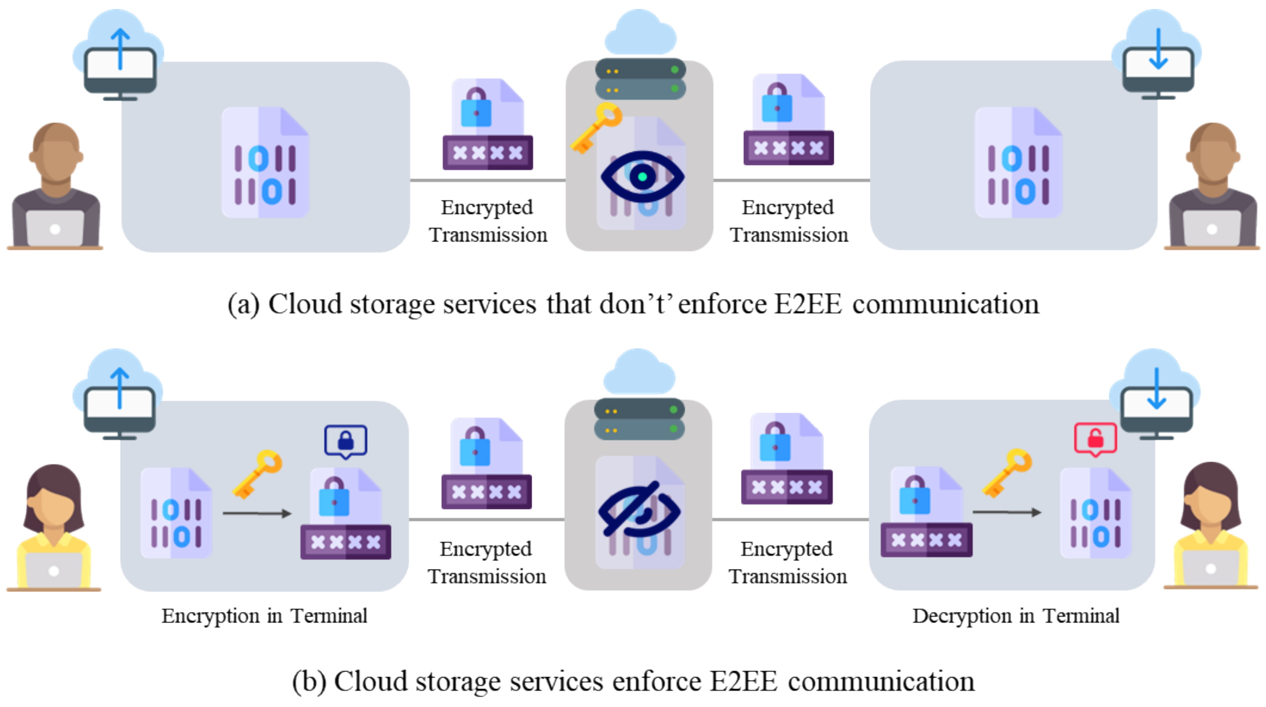


Figure 1 - *Communication types of cloud storage services*

contents uploaded by users. While this enhanced security strengthens data protection, it means that a user should remember the password not to lose access to their contents. Therefore, MEGA recommends users to manage a password recovery key separately.

Due to these characteristics, malicious users who engage in sharing illegal content utilize cloud services that employ E2EE techniques, taking into consideration the potential for leakage of crime-related data and the identification of accounts [2, 3, 4, 5, 6, 7, 8]. As a result, digital forensic experts may experience difficulties by not acquiring sufficient information regarding the suspect's criminal activities.

2.2. Related works

2.2.1. Digital forensic approaches for cloud storage services

Previous studies on cloud-based services have primarily focused on remote data collection using open and internal APIs, as well as investigating local artifacts in mobile and PC environments.

Chung et al [13] proposed a method for investigating cloud storage-related information by extracting artifacts found on PCs and mobile devices, targeting applications such as Amazon S3, Dropbox, and Evernote, which provide cloud storage.

Martini and Choo [14] introduced a method for collecting user data by analyzing the open-source code of ownCloud. They suggested utilizing server log files and metadata when the forensic experts cannot collect digital evidence thoroughly on the client side.

Han et al [15] proposed and implemented a method for efficiently selecting and collecting forensically significant metadata using Dropbox and OneDrive APIs from digital forensic perspectives.

Kim et al. [16] proposed a method to track user behavior based on collected cloud data using Google Takeout.

Yang et al. [17] developed a systematic approach for selectively collecting data from cloud storage, specifically OneDrive. Their method utilizes web APIs to gather metadata and contents from remote servers. Additionally, they introduced a framework to address limitations in existing commercial digital forensic tools.

While the methods proposed in previous research help to collect cloud resources from general cloud services, they may not apply to services such as MEGA, which employs E2EE techniques. In such services, the data is stored in an encrypted state within the servers, making it difficult to expect complete data collection by invoking required APIs.

2.2.2. Studies on MEGA service from a security and forensics perspective

Thamburasa et al. [18] conducted a study on the traces generated by the MEGA and IDrive cloud storage services when used in Windows 7. They identified these traces from local artifacts such as physical memory, registry, and web browser logs.

Daryabar et al. [19] studied the MEGA client app on both Android and iOS platforms, in order to identify a range of artifacts arising from user activities, such as login, uploading, downloading, deletion, and sharing of files.

Ji et al. [20] analyzed the security policy of MEGA and identified the structure of download URIs (Uniform Resource Identifiers) that require passwords. Also, they proposed attacks that can guess valid passwords. This research focused on analyzing the cryptographic vulnerabilities of publicly shared download links and explaining methods to access the data without the owner's account information.

The existing digital forensic research focusing on MEGA has mainly centered around identifying artifacts generated on local client systems (PCs, mobiles). Therefore, these studies assume that devices used with MEGA are available for acquisition and data collection. However, if access to the PC or mobile device is difficult or if the device's state has been altered, such as artifact deletion, conducting smooth analysis of MEGA usage history becomes challenging. Moreover, using the method of collecting files through download links that require password input has limitations, as it is difficult to obtain publicly shared download links that the investigation target storage may contain, and selective collection of desired data is not possible. In this paper, we aim to address these limitations of previous studies by proposing methods to effectively explore and selectively collect user data stored on remote MEGA servers.

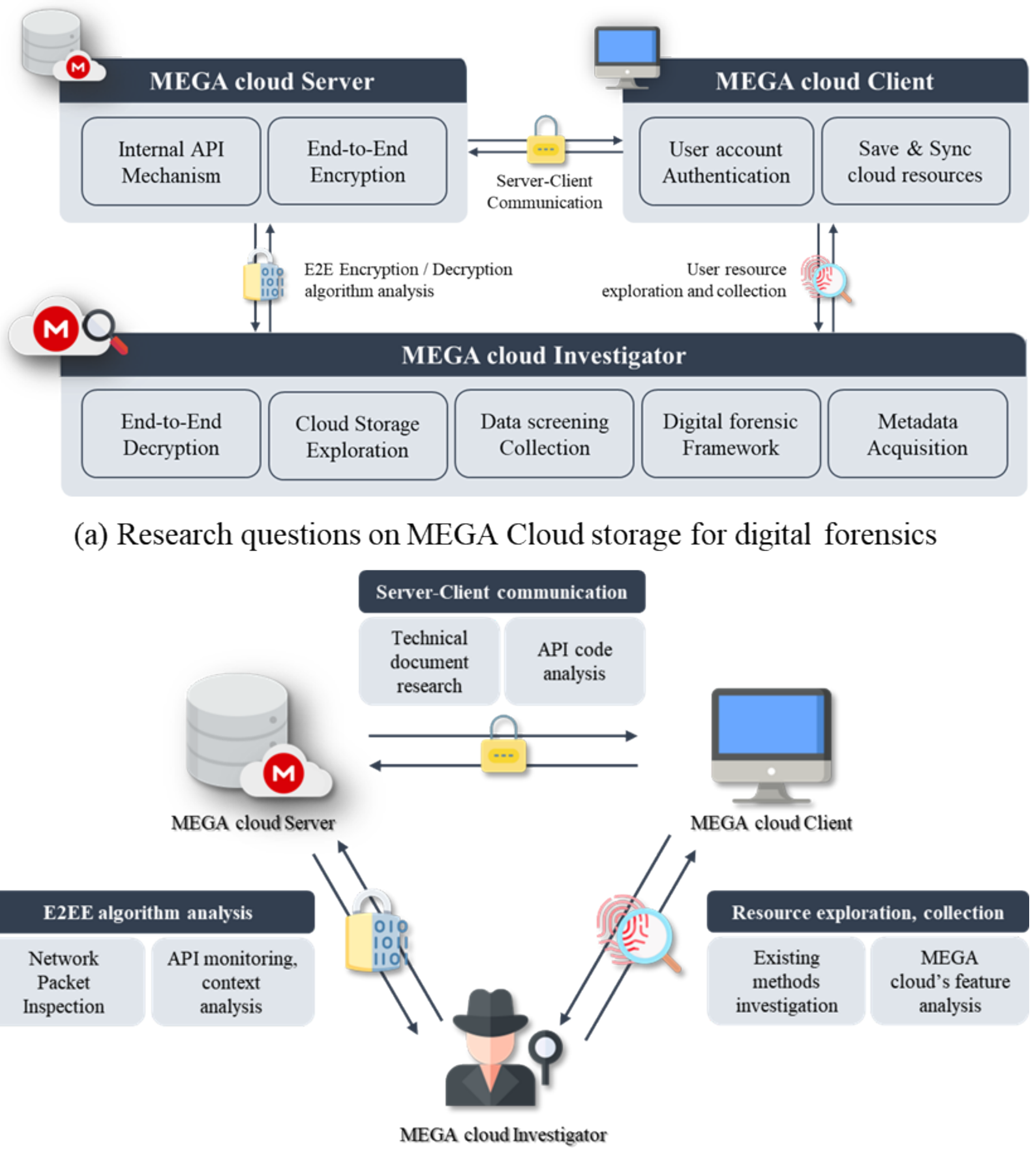
3. Research questions and methods

3.1. Research questions

In this section, we discuss research questions regarding digital forensic activities when using MEGA. As depicted in Figure-2a, it is essential to understand a request and response structure of MEGA APIs and the underlying encryption algorithms to gain access to remote data on MEGA. Additionally, we examine whether we could apply existing cloud forensic frameworks to MEGA.

The main research questions of this paper are as follows:

- **RQ1:** How does the communication mechanism between the MEGA server and the client (dedicated app or web browser) operate?
- **RQ2:** Does MEGA provide web APIs for user authentication, authorization, data exploration, and selective collection? If so, how are their request and response packets specifically structured?
- **RQ3:** Does MEGA's security policies, such as E2EE, affects digital forensic activities?
- **RQ4:** Can digital forensic experts apply existing methods and cloud forensic tools to MEGA? If not, what aspects are required?


MEGA cloud Server
Internal API Mechanism
End-to-End Encryption
MEGA cloud Client
User account Authentication
Save & Sync cloud resources
Server-Client Communication
E2E Encryption / Decryption algorithm analysis
User resource exploration and collection
MEGA cloud Investigator
End-to-End Decryption
Cloud Storage Exploration
Data screening Collection
Digital forensic Framework
Metadata Acquisition
(a) Research questions on MEGA Cloud storage for digital forensics
Server-Client communication
Technical document research
API code analysis
MEGA cloud Server
MEGA cloud Client
E2EE algorithm analysis
Network Packet Inspection
API monitoring, context analysis
Resource exploration, collection
Existing methods investigation
MEGA cloud's feature analysis
MEGA cloud Investigator

(b) Research methods on MEGA Cloud storage for digital forensics

Figure 2 -*Research questions and methods on MEGA Cloud storage for digital forensics.*

- **RQ5:** What data does MEGA apply to E2EE, and how can a user see decrypted data?

3.2 Research methods

Figure-2b illustrates the research methods and approaches to answer the research questions.

- **Utilizing state-of-the-art forensic frameworks for cloud storage services:** This study examines the characteristics of MEGA to identify effective selective collection methods for remote MEGA data. We develop our method based on the CATCH framework [17], which supports investigative activities at each stage.

- **Literature review including official technical documentation:** This study conducts a thorough investigation on official documents, SDK codes, and other publicly available materials to understand communication and E2EE mechanisms of MEGA. The research was conducted from 2022 to 2023, and the official technical documentation that could be obtained from MEGA during that period is based on the ‘MEGA Security Whitepaper Third Edition [21]’ updated in June 2022. Further experiments and analyses were also tested in MEGA SDK v4.28.0 [22] and MEGA Cloud 4.10.0 versions provided as web services.
- **Reverse engineering communication mechanisms through network packet inspection:** This study sets up an experimental environment to simulate communication between MEGA clients and servers to examine the operation of REST (REpresentational State Transfer) APIs. We reverse-engineer to understand the syntax of API invocations and responses for user authentication and data selective collection.
- **Reverse engineering E2EE mechanisms through dynamic analysis of open-source information and code:** This study documents and validates the overall process of decrypting sample data encrypted with E2EE through experiments.

4. Experiments and results

In this section, we introduce the methods to access cloud resources stored in MEGA by calling APIs and describe available cloud resources. Building upon the existing framework named CATCH [19], we divide the process into three main steps: authentication, exploration, and filtering & collection: (1) Obtaining values such as 'SID,' 'User Hash,' and 'User Private Key' required for MEGA

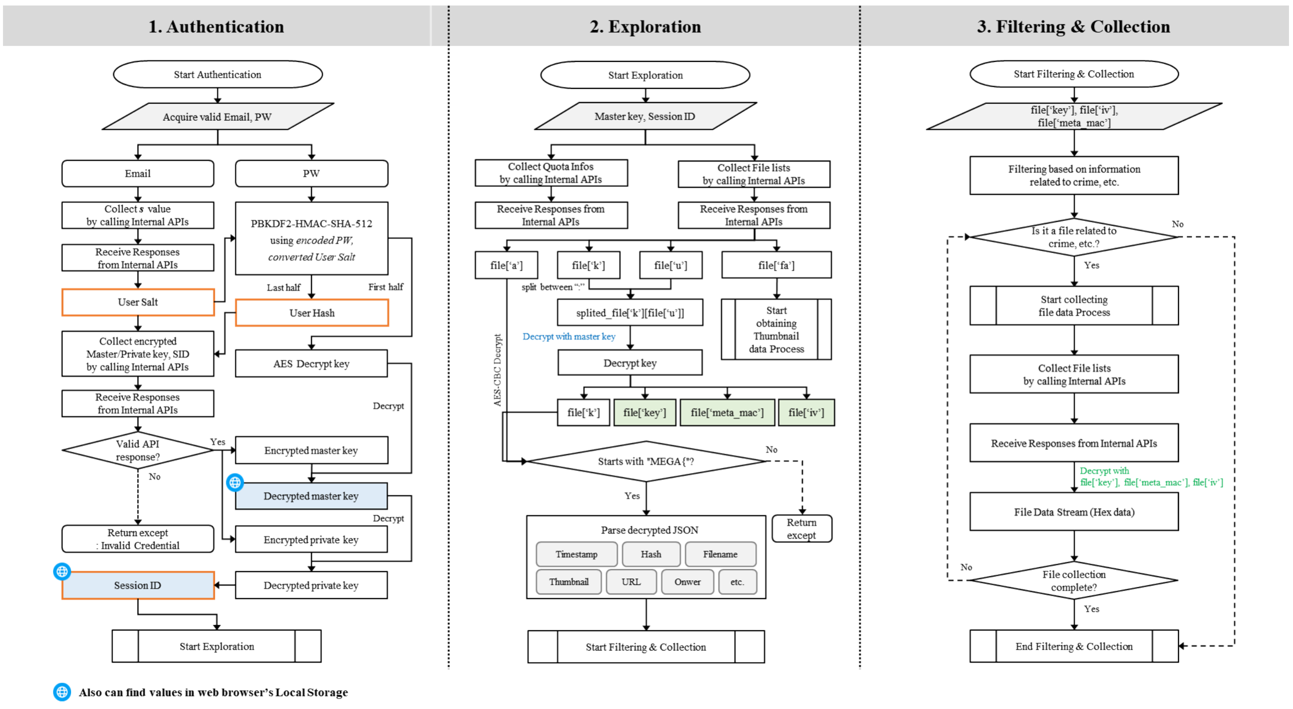
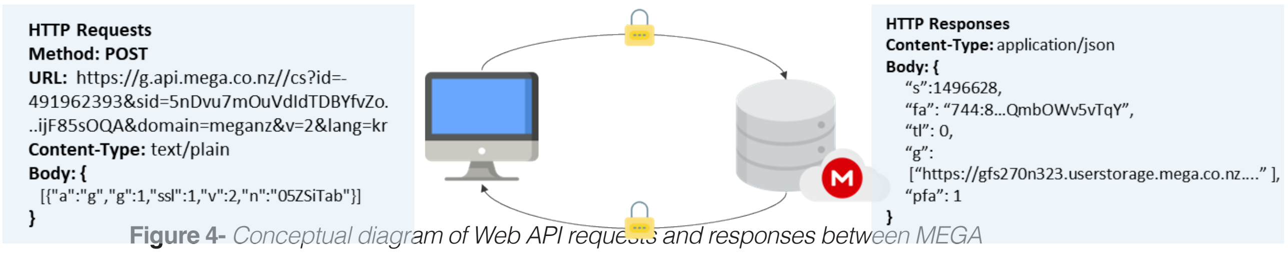

1. Authentication
2. Exploration
3. Filtering & Collection
Start Authentication
Acquire valid Email, PW
Email
PW
Collect s value by calling Internal APIs
Receive Responses from Internal APIs
PBKDF2-HMAC-SHA-512 using encoded PW, converted User Salt
Last half
First half
User Salt
User Hash
Collect encrypted Master/Private key, SID by calling Internal APIs
AES Decrypt key
Receive Responses from Internal APIs
Decrypt
Valid API response?
Yes
No
Encrypted master key
Decrypted master key
Decrypt
Return except : Invalid Credential
Encrypted private key
Session ID
Decrypted private key
Start Exploration
Also can find values in web browser's Local Storage
Start Exploration
Master key, Session ID
Collect Quota Infos by calling Internal APIs
Collect File lists by calling Internal APIs
Receive Responses from Internal APIs
Receive Responses from Internal APIs
file['a']
file['k']
file['u']
file['fa']
split between ":"
splited_file['k'][file['u']]
Start obtaining Thumbnail data Process
Decrypt with master key
Decrypt key
AES-CBC Decrypt
file['k']
file['key']
file['meta_mac']
file['iv']
Starts with "MEGA{"?
No
Yes
Return except
Parse decrypted JSON
Timestamp
Hash
Filename
Thumbnail
URL
Onwer
etc.
Start Filtering & Collection
Start Filtering & Collection
file['key'], file['iv'], file['meta_mac']
Filtering based on information related to crime, etc.
Is it a file related to crime, etc.?
No
Yes
Start collecting file data Process
Collect File lists by calling Internal APIs
Receive Responses from Internal APIs
Decrypt with file['key'], file['meta_mac'], file['iv']
File Data Stream (Hex data)
No
File collection complete?
Yes
End Filtering & Collection

Figure 3- *Workflow for authentication, exploration, and filtering & collection.*



```
HTTP Requests
Method: POST
URL:  https://g.api.mega.co.nz//cs?id=-
491962393&sid=5nDvu7mOuVdIdTDBYfvZo.
..ijF85sOQA&domain=meganz&v=2&lang=kr
Content-Type: text/plain
Body: {
  [{"a":"g","g":1,"ssl":1,"v":2,"n":"05ZSiTab"}]
}
```

```
HTTP Responses
Content-Type: application/json
Body: {
    "s":1496628,
    "fa": "744:8...QmbOWv5vTqY",
    "tl": 0,
    "g":
     ["https://gfs270n323.userstorage.mega.co.nz...." ],
    "pfa": 1
}
```

Figure 4- Conceptual diagram of Web API requests and responses between MEGA

Figure 4 - *Conceptual diagram of Web API requests and responses between MEGA Cloud clients and servers.*

Table 1- MEGA internal APIs and their required parameters

Category	REST API URL using POST
(a) Account Information	https://g.api.mega.co.nz/cs?id={variable}sid={variable}
(b) Storage Information	https://g.api.mega.co.nz/cs?id={variable}ak={variable}sid={variable}
(c) User Salt	https://g.api.mega.co.nz/cs?id={variable}ak={variable}
(d) Session ID	https://g.api.mega.co.nz/cs?id={variable}ak={variable}
(e) List of Files with Metadata	https://g.api.mega.co.nz/cs?id={variable}ak={variable}sid={variable}
(f) Thumbnails of Files	https://g.api.mega.co.nz/cs?id={variable}ak={variable}sid={variable}
(g) Encrypted File Content	https://g.api.mega.co.nz/cs?id={variable}ak={variable}sid={variable}

internal API requests and data decryption with user ID (email) and password, (2) Exploring metadata such as file names, paths, content hash values, thumbnails, download URLs, and timestamps, and (3) Selectively collecting data related to the investigation using the metadata. Figure-3 summarizes three steps of authentication, exploration, and filtering and collection proposed in this paper.

4.1 Internal APIs and required parameters for communicating with MEGA server

To access cloud resources from MEGA, forensic experts need to properly set key-value pairs in JSON format within the request header body of each API. Therefore, in this section, we introduce in-depth packet inspection of internal APIs. In addition, we provide essential keys for each API to collect potential digital evidence from MEGA.

Figure-4 illustrates the conceptual diagram of API requests and responses between the MEGA client and server. When invoking the identified MEGA APIs, Table-1, it is necessary to include the appropriate key-value pairs in the JSON format. For instance, to collect metadata and download URL information of a specific file, the API request should include the JSON data with the value of ‘g’ in the key ‘a’, the value of ‘1’ in the key ‘g’, and the handle value of the target file in the key ‘n’ within the HTTP message body. For the parameters listed in Table-1, MEGA changes the 'id' for each request based on the current time and sets the 'ak' parameter to the MEGA application version (app key).

4.2 Authentication of a user account

This section describes the process of obtaining the values required in the user authentication of MEGA, such as SID, User Hash, and User Private Key. The first step in Figure-3 illustrates authentication process. Below this, the detailed process for obtaining specific values or user authentication and result collection, including examples applying the proposed methodology, are elaborated.

Table 2- Pairs of keys and values contained in a JSON-formatted message body that is transferred to MEGA server

Category	JSON-formatted message body		
	Key name	**Value**	**Description**
a) Account Information)	a	ug	-
	a	uq	-
b) Storage Information)	xfer	1	-
	strg	1	-
	r	1	-
c) User Salt)	a	us0	-
	user	{email}	User's email address
d) Session ID)	a	us	-
	user	{email}	User's email address
	uh	{value}	The second half of PBKDF2 ((Base64URL(User's password =)
e) List of Files with Metadata)	a	f	-
	c	1	-
	r	1	-
f) Thumbnails of Files)	a	ufa	-
	r	1	-
	fah	{handle}	(File handle (= a unique identifier
g) Encrypted File Content)	a	g	-
	g	1	-
	p or n	{handle}	(File handle (= a unique identifier

4.2.1. Getting a user salt value

For user authentication, the request API is called by first setting the keys listed in (c) of Table-2. The MEGA server returns a 43-character string, unique to each user ID (always the same when re-logging in), to the 's' key of the response message (refer to Table-3 (c) for details). For reference, since the corresponding response value is a Base64URL-encoded string, an array of 32 bytes must be obtained by decoding to be utilized for use in generating the user hash in the subsequent steps.

4.2.2. Generating a user hash value

In the second step, a user hash value is generated using the PBKDF2 (Password-Based Key Derivation Function 2) with HMAC (Hash-based Message Authentication Code) algorithm. As shown in Table-4, the hash function 'SHA-512' is utilized, and it involves the user's password converted to bytes format and the user salt value. Afterward, a series of steps result in the creation of a 32-byte binary key, of which the latter 16 bytes are used as the user hash value in the subsequent steps.

4.2.3. Obtaining a sessionID

The final step in authentication is to generate sessionID which is essential for collecting information about files and folders stored within cloud storage. Initially, the user includes their email address

Table 3- Pairs of keys and values contained in a JSON-formatted message body that is received from MEGA server

Category	JSON-formatted message body		
	Key name	**Value**	**Description**
a) Account Information)	u	{id}	(User handle (= a unique identifier
	since	{value}	Account creation time
	email	{email}	User email
	name	{name}	User name
b) Storage Information)	cstrg	{value}	Total account storage usage
	mstrg	{value}	Maximum storage allowance
	cstrgn	{value}	Root, shared nodes information
c) User Salt)	s	{salt}	User salt
	v	{version}	Account version
d) Session ID)	csid	{csid}	Encrypted session ID
	privk	{key}	Encrypted private key
	k	{key}	Encrypted master key
e) List of Files with Metadata)	f	{value}	Directory/File-related information p: Parent ID, u: Owner ID, t: Type, s: File size, ts: Uploaded timestamp
	f2	{value}	File versioning information
	s	{value}	Outgoing share information
	u	{value}	Contacts information
	ph	{value}	Shared link information
f) Thumbnails of Files)	p	{url}	Downloadable URL for a file's thumbnail
g) Encrypted File Content)	g	{url}	Downloadable URL for a file's content
	s	{size}	File size
	at	{value}	(File attribute (name, label, favorite
	fa	{value}	(File attribute (thumbnail

and the generated user hash from the previous step in the message body of the request API, Table-2 (d). In response, encrypted master key and private key to be used in the RSA algorithm, along with the encrypted sessionID, are returned Table-3 (d).

Subsequently, an operation is performed to decrypt the encrypted RSA private key. The required master key can be decrypted using the first 16 bytes of the return value of the PBKDF2 function used in the previous step. As a result, utilizing the master key along with AES-CBC (Cipher Block Chaining) mode allows obtaining the RSA private key. Lastly, the encrypted sessionID can be obtained by decoding Base64URL and then decrypting it with RSA private keys. The master key and sessionID obtained in this way are utilized as essential elements of API calls to explore and collect data in the next step.

In addition, MEGA supports automatic login, and upon logging into MEGA via a web browser, it can be observed that the sessionID and master key are stored in the Local Storage under the respective keys sid and k. Although this sessionID changes with each login, a significant characteristic is the previous sessionID can be reused unless the user's password changes. Furthermore, the master key remains unchanged even upon session recon-

Table 4- Parameters for PBKDF2 with HMAC

Parameter	Value	Note
Hash algorithm	SHA-512	-
User password	(a plain password (UTF-8	The value is converted to 'bytes' type
Salt value	a user salt obtained in the previous step	The value is converted to 'bytes' type
Iteration count	100,000	-
Key length	32	-

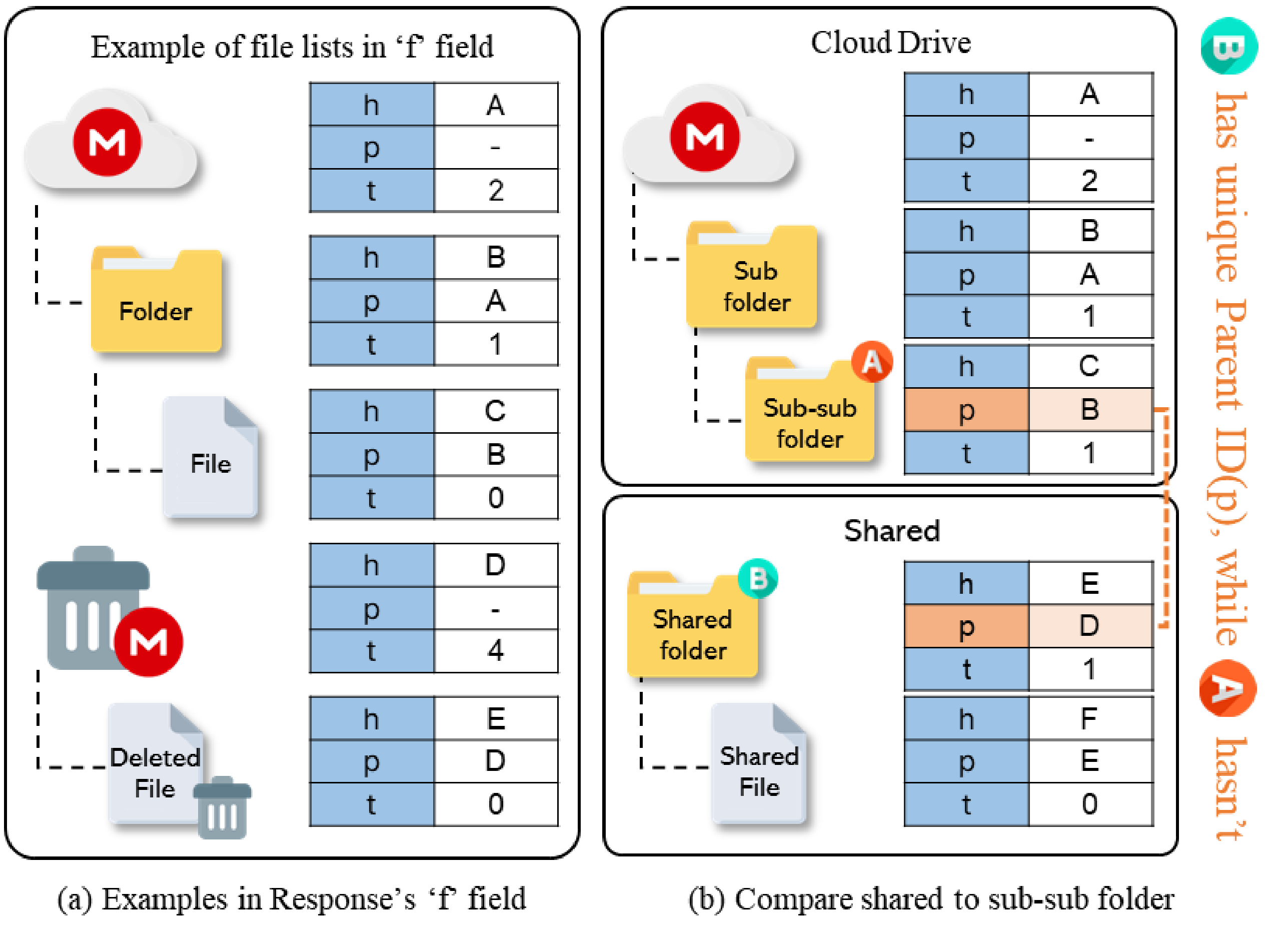


Figure 5- *Various examples of 'h' key, 'p' key, and 't' key values.*

nection. Therefore, if these values can be found in the browser or physical memory, an attacker could attempt user authentication through a Credential Cloning Attack.

4.3. Exploration of files and their metadata

If user authentication is successful using a valid user account, one can explore a list of files stored on cloud storage and detailed metadata. The second step in Figure-3 summarizes the process of exploring storage based on metadata from files uploaded to the MEGA. For reference, calls to all APIs refer to Table-1 described later require session ID obtained in the user authentication stage. The response values from the MEGA server directly related to data

Table 5- List of important keys and sub-keys contained in a JSON-formatted message body that is received from MEGA server

Key	Meaning	Sub-key	Meaning
ok	Relevant user information	h	User (or shared user's) ID
f (f2)	Directory/File-related information (File versioning information)	h	File/Directory ID
		p	Parent ID
		u	Owner ID
		t	T y p e File, 1: Directory (can be root if shared), 2: Cloud :0) ((Drive (root), 3: Inbox (root), 4: Trash Bin (root
		a	(.File attribute (name, label, favorite, etc
		k	Decryption key
		r	if incoming share 1
		su	Shared user's ID if incoming share
		s	File size
		fa	(File attribute (file's thumbnail
		ts	Uploaded timestamp
s	Outgoing share information	h	Shared folder ID
		u	User ID who shared this folder
		ts	Shared timestamp
us	Contacts information	u	User ID
		ts	Registered timestamp
		c	Me: 1, Friend: 2
		m	User e-mail
ps	Outgoing or Pending share information	h	File ID
		p	Shared ID
		ts	Shared timestamp
opc	Outgoing or Pending contacts information	p	Shared ID
		m	User ID who has access to this folder
		e	User ID who shared this folder
		msg	Request message
		ts	Request message sent timestamp
ph	Shared link information	h	File ID
		ph	Shared ID

exploration and collection can be referenced from Table-5.

4.3.1 Getting a list of files

The list of all items (files, directories, etc.) uploaded to the cloud storage can be obtained through the 'f' key in the response message of the API requests mentioned in (e) of Table-2 and Table-3, which includes the file ID, owner ID, upload time, and more. This key contains a sub-key 't' indicating the 'type', allowing differentiation of specific types of resources as follows: file ('t': 0), directory ('t': 1), Cloud Drive (space where users can access files and folders uploaded to MEGA, 't': 2), and Trash Box ('t': 4).

Additionally, the sub-key 'h' represents the unique ID of each item, and by referencing the parent ID, sub-key 'p', the relationships between items can be understood. Furthermore, details such as the owner ID of the file (sub-key 'u'), file size (sub-key 's'), uploaded time (sub-key 'ts'), etc., can be identified. An example of key values for folders and files in the Cloud Drive and deleted files in the Trash Box is provided in Figure-5a.

If a folder with shared relationships such as Incoming Share or Outgoing Share exists, a JSON element of folder type ('t'=1) with a unique Parent ID ('p') is generated in the sub-key 'f' within the API response. It is important to verify whether the Parent ID ('p') does not belong to the Cloud Drive Root or Trash Root, as shared folders can be misinterpreted as a subfolder of folders belonging to the Cloud Drive. The Figure-5b shows examples of the main keys of the shared folder and sub-file and the unique Parent ID of the shared folder.

By setting the value 'uq' in the 'a' key during API requests, as illustrated in Table-2 (b), it is possible to collect information about the user's cloud usage, including the Cloud Drive, Inbox, and Trash Box roots, as well as shared folder root information generated for each shared folder Table-3 (b). Through this method, it is possible to obtain information about shared folders and accurately interpret the file lists.

4.3.2. Decrypting metadata

The key related to metadata, such as file name, in the collected file list value is the sub-key 'a', which means the file attribute within the 'f' key summarized in Table-5. However, the sub-key 'a' is encrypted, necessitating a decryption step for appropriate file metadata identification. During the preprocessing step for decrypting the file attributes in AES-CBC mode, new keys such as the file unique vector value ('iv') are added to the JSON element, referring to the sub-key 'k' representing the file decryption key and 'u' representing the owner ID. If the resulting string after all the steps begins with the string 'MEGA{', then the decryption is successful, and the filename can be decrypted by parsing the JSON structure.

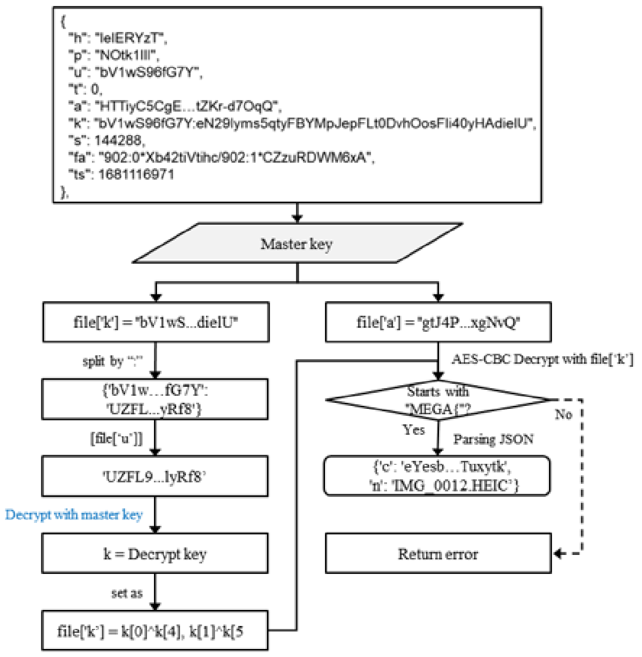
The decrypted metadata includes not only the file ID and filename but also information regarding favorites and label settings that users can set for files and folders through web service of MEGA. When comparing the JSON values of files with no settings to those with favorites or deleted labels, the sub-key configuration of the decrypted 'a' key is different. For instance, in the case of an element with favorites set, a sub-key 'fav' is created within the 'a' key, and for elements with labels set, a sub-key 'lbl' is created. The value of the sub-key 'lbl' represents each of the 7 label colors that users can set. Additionally, for deleted elements, a sub-key 'rr' is created, clearly distinguished from the JSON structure of the element that has not been deleted.

Figure-6a provides a detailed illustration of all steps to decrypt file metadata, referring to the sub-keys 'a', 'k', and 'u'.

4.3.3. Matching owner IDs to user emails

The sub-key 'u' under 'f', which can be found in the file metadata in Table-5, represents the file's owner ID. However, solely relying on the Owner ID does not easily identify the file owner. Therefore, by matching the user ID through the 'u' key in Table-5, where contact information can be obtained, the user's email can be identified based on the value of the sub-key 'm' under 'u' Table-6 (a)

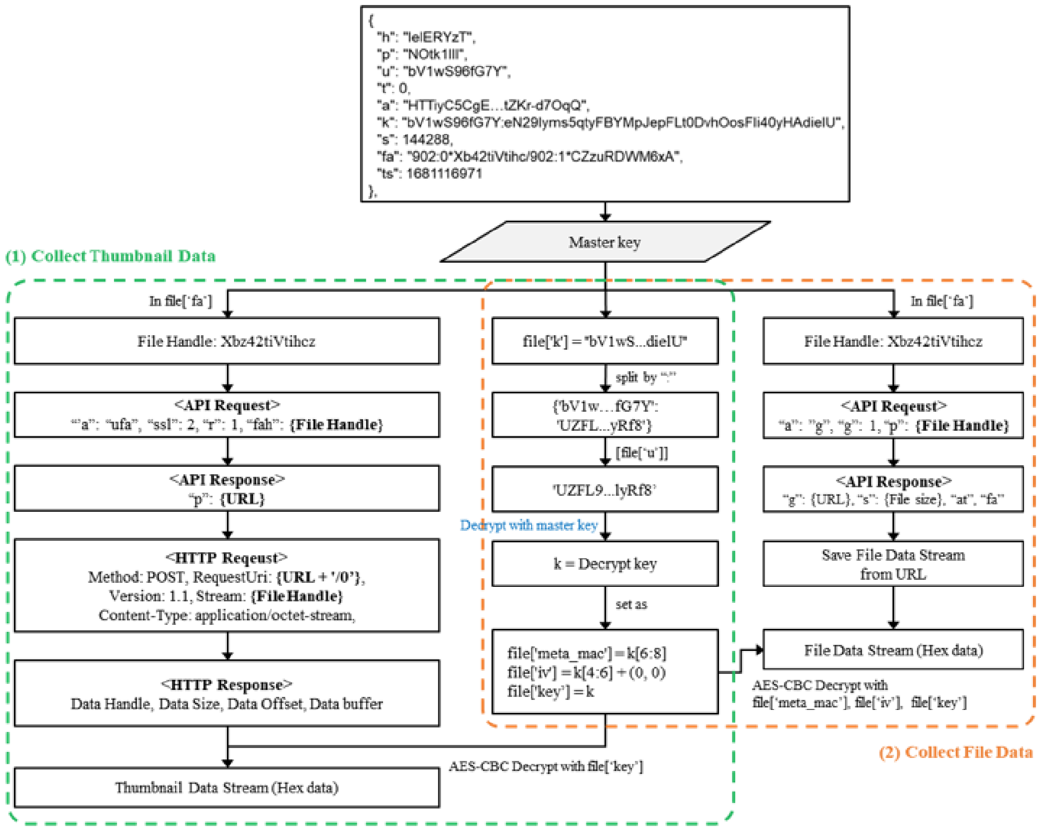
If the sub-key 'r' exists within the 'f' key, it indicates elements related to Incoming share. In this case, the simultaneously existing sub-key 'su' allows knowing the ID of the user who started the share Table-6 (b). For files and folders related to Outgoing share, the 's' key in Table-5 provides the



```
{
 "h": "lelERYzT",
 "p": "NOtk1lIl",
 "u": "bV1wS96fG7Y",
 "t": 0,
 "a": "HTTiyC5CgE…tZKr-d7OqQ",
 "k": "bV1wS96fG7Y:eN29lyms5qtyFBYMpJepFLt0DvhOosFli40yHAdielU",
 "s": 144288,
 "fa": "902:0*Xb42tiVtihc/902:1*CZzuRDWM6xA",
 "ts": 1681116971
},
```

Master key
file['k'] = "bV1wS...dielU"
split by ":"
{'bV1w…fG7Y': 'UZFL...yRf8'}
[file['u']]
'UZFL9...lyRf8'
Decrypt with master key
k = Decrypt key
set as
file['k'] = k[0]^k[4], k[1]^k[5
file['a'] = "gtJ4P...xgNvQ"
AES-CBC Decrypt with file['k']
Starts with "MEGA{"?
Yes
No
Parsing JSON
{'c': 'eYesb…Tuxytk', 'n': 'IMG_0012.HEIC'}
Return error

(a) The entire process of the file metadata decryption

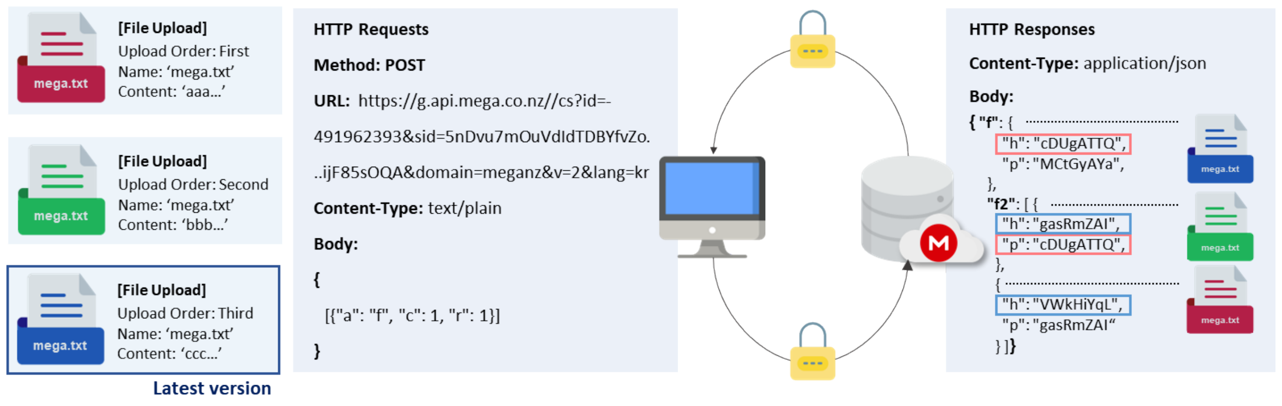


```
{
 "h": "lelERYzT",
 "p": "NOtk1lIl",
 "u": "bV1wS96fG7Y",
 "t": 0,
 "a": "HTTiyC5CgE…tZKr-d7OqQ",
 "k": "bV1wS96fG7Y:eN29lyms5qtyFBYMpJepFLt0DvhOosFli40yHAdielU",
 "s": 144288,
 "fa": "902:0*Xb42tiVtihc/902:1*CZzuRDWM6xA",
 "ts": 1681116971
},
```

Master key
(1) Collect Thumbnail Data
In file['fa']
File Handle: Xbz42tiVtihcz
<API Request>
"a": "ufa", "ssl": 2, "r": 1, "fah": {File Handle}
<API Response>
"p": {URL}
<HTTP Reqeust>
Method: POST, RequestUri: {URL + '/0'},
Version: 1.1, Stream: {File Handle}
Content-Type: application/octet-stream,
<HTTP Response>
Data Handle, Data Size, Data Offset, Data buffer
Thumbnail Data Stream (Hex data)
AES-CBC Decrypt with file['key']
file['k'] = "bV1wS...dielU"
split by ":"
{'bV1w…fG7Y': 'UZFL...yRf8'}
[file['u']]
'UZFL9...lyRf8'
Decrypt with master key
k = Decrypt key
set as
file['meta_mac'] = k[6:8]
file['iv'] = k[4:6] + (0, 0)
file['key'] = k
In file['fa']
File Handle: Xbz42tiVtihcz
<API Reqeust>
"a": "g", "g": 1, "p": {File Handle}
<API Response>
"g": {URL}, "s": {File size}, "at", "fa"
Save File Data Stream from URL
File Data Stream (Hex data)
AES-CBC Decrypt with file['meta_mac'], file['iv'], file['key']
(2) Collect File Data

(b) Full process for (1) thumbnail data and (2) file data collection

Figure 6- *Data collection process of metadata and file.*



HTTP Requests

Method: POST

URL: https://g.api.mega.co.nz//cs?id=-491962393&sid=5nDvu7mOuVdIdTDBYfvZo...ijF85sOQA&domain=meganz&v=2&lang=kr

Content-Type: text/plain

Body:

```
{
  [{"a": "f", "c": 1, "r": 1}]
}
```

HTTP Responses

Content-Type: application/json

Body:

```
{ "f": {
    "h": "cDUgATTQ",
    "p": "MCtGyAYa",
  },
  "f2": [ {
    "h": "gasRmZAI",
    "p": "cDUgATTQ",
    },
    {
    "h": "VWkHiYqL",
    "p": "gasRmZAI"
    } ]}
```

Figure 7- *Interpretation of JSON structures for collecting file version information.*

ID of the shared folder (sub-key 'h') and the recipient's ID (sub-key 'u'). In such cases, by finding the File/Directory ID in the 'f' key that matches the value of the sub-key 'h' under the 's' key and finding the User ID in the 'u' key that matches the value of the sub-key 'u' under the 's' key, the user's email can be specified by referring to a total of three keys Table-6 (c).

4.3.4. Collecting file history

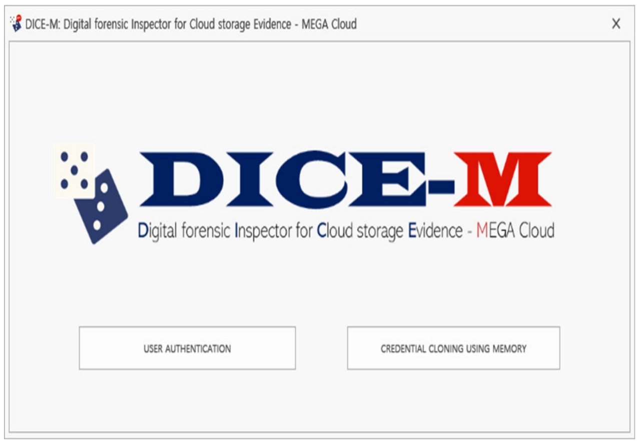
File versioning is a feature that tracks the changes made to a file and allows for the restoration of previous versions. When attempting to upload a file with the same name as one already presents in MEGA, by choosing 'Upload and update', users can view information about each version, such as the upload timestamp and the modifying user, and restore to a specific version.

When collecting version information of files using the Internal API, the request key as in Table-2 (e) can be used, and in response, the version relationship can be identified by using the 'f' key and the 'f2' key of Table-3 (e). Let's assume three file versions were created by uploading different files with the same name twice in addition to the initial upload. The metadata for the most recently uploaded version can be collected from the 'f' key, while the metadata for the previous two versions can be collected from the 'f2' key. Each time a new version of the file is uploaded, the parent ID of the previous version file changes to the ID of the new version file. Therefore, by referring to the sub-keys 'h (File ID)' and 'p (Parent ID)' under the 'f2' key, as shown in Figure-7, it is possible to determine the presence of file versions and the upload sequence between versions.

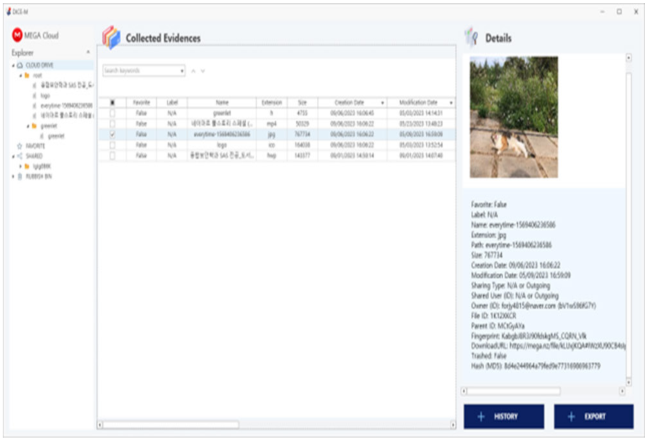
4.3.5. Collecting thumbnails of files

The sub-key 'fa' under 'f', which can be found in Table-5, signifies file attributes related to thumbnails for image and document files. This key is also encrypted, so a series of decryption steps is required to collect thumbnail information. First, it is possible to determine the existence of thumbnail and preview data by checking the presence of the '/' symbol in the sub-key 'fa'. If they exist, the string preceding the '/' symbol can be divided into a file handle and file ID using the ':' and '*' symbols as further separators.

The file handle, as mentioned in Table-2 (f), is utilized as the request key to collect URL information for downloading thumbnail images. Subsequently, upon receiving a URL in response to this request Table-3 (f), the HTTP request proceeds through the process of Base64-URL decoding of the file handle. During this process, the hexadecimal data of the


DICE-M: Digital forensic Inspector for Cloud storage Evidence - MEGA Cloud
DICE-M
Digital forensic Inspector for Cloud storage Evidence - MEGA Cloud
USER AUTHENTICATION
CREDENTIAL CLONING USING MEMORY

(a) Initial interface of the automated tool



	Favorite	Label	Name	Extension	Size	Creation Date	Modification Date
☐	False	N/A	greenlet	h	4735	09/06/2023 16:06:45	05/03/2023 14:14:31
☐	False	N/A	[illegible]	mp4	50329	09/06/2023 16:06:22	05/23/2023 13:48:23
✓	False	N/A	everytime-1569406236586	jpg	767734	09/06/2023 16:06:22	05/09/2023 16:59:08
☐	False	N/A	logo	ico	164038	09/06/2023 16:06:22	05/03/2023 13:52:54
☐	False	N/A	[illegible] SAS [illegible]	hwp	143377	09/01/2023 14:50:14	09/01/2023 14:07:40

Favorite: False
Label: N/A
Name: everytime-1569406236586
Extension: jpg
Path: everytime-1569406236586
Size: 767734
Creation Date: 09/06/2023 16:06:22
Modification Date: 05/09/2023 16:59:08
Sharing Type: N/A or Outgoing
Shared User (ID): N/A or Outgoing
Owner (ID): forjy4815@naver.com (bV1w59tKG7Y)
File ID: 1K12XXCR
Parent ID: MCtGyAYa
Fingerprint: KabgbJ8R3/90fdskgMS_CQRN_Vlk
DownloadURL: https://mega.nz/file/kLUxjKQA#Wi2XJ90CB4dq
Trashed: False
Hash (MD5): 8d4e244964a79fed9e77316908963779

(b) Main interface of the automated tool

Figure 8- *The interfaces of the automated tool.*

thumbnail image can be obtained through the returned data buffer. The complete steps for thumbnail collection are elaborated in detail in Figure-6b (1).

4. 4. Filtering and Collection of files' content

The filtering and file content collection stage involves selectively collecting files based on the metadata such as file names and thumbnails identified in the previous exploration stage. In the third step of Figure-3, the whole process is summarized. Analysts investigating the MEGA can examine aspects such as whether the file data is related to criminal activity and is in the analyst's interest to determine whether the file data is collected.

4. 4. 1. Filtering files

In the prior research, the CATCH framework, which was proposed for selective data collection in the cloud, utilized a method of requesting the server to filter files based on collected metadata by including queries with specific keywords or times [17]. However, in the case of MEGA, it does not provide a separate Internal API for searching. Therefore, to enable easy data filtering and searching, the functionality of DevExpress, utilized in the implementation of the WPF tool aimed at independently selecting the results obtained in the exploration phase, was utilized at the local level.

4.4.2 Decrypting content of individual files

The (g) of Table-2 and Table-3 describe the message body that should be included in the API request and response packets used to collect file contents. In this context, the encrypted data stream can be obtained from the URL included in the API response, and the data stream can be decrypted using the decrypt key, initialization vector, and Meta Mac code for integrity verification. Detailed steps for collecting file contents are explained in Figure-6b (2).

4.5. Implementation

Based on the research described in Chapter 4 and the published tools [17, 23, 24], a tool that can be used for digital forensic analysis of MEGA was developed. This tool, shown as Figure-8, provides storage navigation using user account information and selection and collection of thumbnail and file content. In addition, even if user account information is not secured through a memory dump with a login history, the key required for login may be searched and utilized (Credential Cloning Attack: described at the end of Chapter 4.2.3). As a result, the proposed tool can

Table 6- Keys referenced when matching owner ID and user email

Key	Sub-key	Meaning	Matching Owner ID and User email		
			a) Uploaded User)	**b) Incoming) Share User**	**c) Outgoing) Share**
f	h	File/Directory ID			V
	u	Owner ID	V		V
	r	(Only exists when is Incoming Share) 1		V	
	su	Shared User ID (Only exists when is Incoming Share)		V	
s	h	Shared Folder ID			V
	u	Sharing Started User ID			V
u	u	User ID	V	V	V
	m, m2	User Email	V	V	V

collect various 17 metadata types crucial for digital forensic investigations from MEGA Cloud, demonstrating its effectiveness compared to tools limited to collecting only 8 [23] or 9 [24] metadata types.

- The proposed tool collects: file name, trashed, created time, modified by me time, owner name, owner id, file size, file extension, file hash, file version, incoming/outgoing share, shared with me time, sharing user id, sharing user email, file downloadable URL, file path, thumbnail
- In contrast, one existing tool [23] only collects: file name, trashed, created time, modified by me time, owner id, file size, file extension, file downloadable URL
- Additionally, another tool [24], compared to the former, can additionally collect thumbnails

5. Conclusions and future work

This paper introduces the E2EE technology used by MEGA, detailing the process of collecting file metadata, thumbnails, and contents from cloud resources. It focuses on user authentication and permission acquisition within MEGA, elucidating the crucial keys for subsequent storage exploration and data collection. The responses to the initial research questions are outlined below:

- **RQ1:** The Internal API used for communication between the server and client in MEGA includes specific key-value pairs in JSON format in the body field of each request and response packet. Chapter 4.1 describes the functioning mechanism of the Internal API, and in Figure-4, you can see an example of JSON-formatted key-value pairs in the body field of Internal API requests and response packets.
- **RQ2:** The web APIs used for MEGA's user authentication, data exploration, and selective collection have different request URL and packet body structures according to their respective purposes. Table-1 presents the Internal API and request URLs of MEGA, while Table-2 and Table-3 describe the JSON-formatted key-value pairs included in the requests and response packets of the Internal API.
- **RQ3:** MEGA's E2EE technique encrypts and decrypts data on the client device, making it difficult for cloud service providers to access

decrypted data. Consequently, in digital forensics activities targeting cloud servers, direct access to data is challenging, and decryption of data requires the client device with user account ID, password, or connection records.

- **RQ4:** Most of the existing research on selective cloud data collection does not focus on cloud services with E2EE, such as MEGA. Therefore, in this study, we propose a data collection approach that can be utilized in digital forensics activities for MEGA by dividing it into three stages: authentication, exploration, filtering & collection Figure-3. Furthermore, we implemented automated tool for efficient analyst.
- **RQ5:** The sub-key 'a' under the 'f' key in Table-5 is end-to-end encrypted, requiring a separate decryption process. Figure-6 illustrates the process of decrypting file metadata and collecting thumbnail and file data. Additionally, as described in Table-6, by referencing other sub-keys, we can match the file owner's ID and email. Further details regarding this are outlined in Chapter 4.3.

The authentication, exploration, filtering, and collection phases for digital forensic investigation of MEGA presented in this study support efficient digital forensic activities. Existing forensic research on cloud storage focused on services without E2EE between clients and servers. Thus, analysis of cloud services with features like MEGA, where the intermediate server cannot access the contents of files uploaded by clients, becomes even more important.

The method proposed in this paper is based on encryption and decryption algorithms used in the complete implementation of the End-to-End Encryption function targeted by MEGA Cloud. This allows various information on resources uploaded to the MEGA Cloud to be obtained. However, if security policies or encryption mechanism of cloud service providers change, the applicability of the methods proposed in this study may be limited.

Nevertheless, even assuming that MEGA's fundamental algorithm has changed, the methodology proposed in this paper for the purpose of effective digital forensics investigation remains unchanged. However, it will be necessary to analyze the new decryption process according to the changed encryption algorithm. This does not take much time for technical updates as long as MEGA maintains an open-source policy. If the policy of disclosing services provided by MEGA as open-source software changes, the effectiveness of research and description on MEGA Cloud, including this paper, will decrease. However, considering the direction of pursuing complete encryption, it can be said that it is very likely to disclose updates of algorithms, additions of new functions, and minor modifications to their official repositories [25] as they have so far.

Furthermore, cybercrime and hacking attacks continue to increase and evolve. Consequently, new security tools and technologies to defend against and respond to them will continue to emerge. Therefore, ongoing reviews incorporating updated encryption technologies are necessary. Henceforth, continual professional development is imperative for security researchers in educational institutions and security personnel in companies and organizations. This will enable them to effectively combat potential attacks and illegal content dissemination crimes, enhance awareness of cloud storage service security, and utilize necessary digital forensic investigations efficiently and proactively. Additionally, future research should propose methods supporting digital forensic investigations targeting other cloud storage services alongside E2EE. Through such analysis, it is anticipated that more effective data collection and analysis of cloud services in the digital forensic field can be achieved. Constant learning and skill enhancement are crucial for leading the future of cybersecurity and responding actively to it. Ultimately, a deeper understanding of E2EE will lead to the introduction of new methodologies and tools for cloud forensics in the future..

Conflict of interest

The authors declare no conflicts of interest.

Source of funding

This work was supported by Police-Lab 2.0 Program(www.kipot.or.kr) funded by the Ministry of Science and ICT(MSIT, Korea) & Korean National Police Agency(KNPA, Korea). [Project Name: Research on Data Acquisition and Analysis for Counter Anti-Forensics/Project Number: 210121M07].

References

1. TechNavio. Cloud Storage Services Market. TechNavio; 2019.
2. United States Department of Justice. KC Man Sentenced to 16 Years for Distributing Child Pornography. https://www.justice.gov/usao-wdmo/pr/kc-man-sentenced-16-years-distributing-child-pornography. Updated 28 Nov 2023.
3. United States Department of Justice. Citrus Heights Man Pleads Guilty to Possession of Child Pornography. https://www.justice.gov/usao-edca/pr/citrus-heights-man-pleads-guilty-possession-child-pornography. Updated 26 Sep 2023.
4. United States Department of Justice. Nicholas Biase: Former West Point Staff Sergeant Sentenced To 42 Months In Prison For Possession Of Child Pornography. https://www.justice.gov/usao-sdny/pr/former-west-point-staff-sergeant-sentenced-42-months-prison-possession-child. Updated 19 Apr 2023.
5. United States Department of Justice. Daniel Ball: Van Wert County Man Sentenced to 10 Years in Prison for Possession of Child Pornography. https://www.justice.gov/usao-ndoh/pr/van-wert-county-man-sentenced-10-years-prison-possession-child-pornography. Updated 19 Apr 2023.
6. United States Department of Justice. Noblesville Man Sentenced to over 10 Years in Federal Prison for Distributing Child Sexual Abuse Material. https://www.justice.gov/usao-sdin/pr/noblesville-man-sentenced-over-10-years-federal-prison-distributing-child-sexual-abuse. Updated 21 Dec 2022.
7. United States Department of Justice. Indianapolis Man Sentenced to 151 Months in Federal Prison for Transporting Child Sexual Abuse Material. https://www.justice.gov/usao-sdin/pr/indianapolis-man-sentenced-151-months-federal-prison-transporting-child-sexual-abuse. Updated 8 Nov 2022.
8. New Jersey Attorney General's Office. Suspended State Trooper Indicted on Child Pornography Charges. https://www.nj.gov/oag/newsreleases19/pr20191219c.html. Published 19 Dec 2019.
9. Seo JY. Discussions on the Searching and Seizing Digital Evidence from Cloud Computing Environments. Jeonbuk Law Review. 2020;64:323-351.
10. Mega Limited. MEGA cloud SDK. sdk. https://github.com/meganz/sdk.git.
11. Mega Limited. Protecting your data and respecting your privacy. https://mega.io/security.
12. Mega Help Centre. What does "zero-knowledge" mean? How does MEGA's zero-knowledge encryption work. https://help.mega.io/security/data-protection/zero-knowledge-encryption.
13. Chung H, Park J, Lee S, Kang C. Digital forensic investigation of cloud storage services. Digital Investigation. 2012;9(2):81-95. doi:10.1016/j.diin.2012.05.015.
14. Martini B, Choo KKR. Cloud storage forensics: ownCloud as a case study. Digital Investigation. 2013;10(4):287-299.

15. Han J, Lee S, Oh J, Kim J, Jeong H. Implementation of Selective Acquisition for Cloud Storage Services based on Metadata. Journal of Digital Forensics. 2020;14(3):305-315. doi:10.22798/kdfs.2020.14.3.305.

16. Kim D, Kim J, Lee S. An Analysis of Google Cloud Data from a Digital Forensic Perspective. Journal of the Korea Institute of Information and Communication Engineering. 2020;24(12):1662-1669. doi:10.6109/jkiice.2020.24.12.1662.

17. Yang J, Kim J, Bang J, Lee S, Park J. CATCH: Cloud Data Acquisition through Comprehensive and Hybrid Approaches. Forensic Science International: Digital Investigation. 2022;43:301442. doi:10.1016/j.fsidi.2022.301442.

18. Thamburasa S, Easwaramoorthy S, Aravind K, Bhushan SB, Moorthy U. Digital forensic analysis of cloud storage data in IDrive and Mega cloud drive. In: 2016 International Conference on Inventive Computation Technologies (ICICT). IEEE; 2016. doi:10.1109/INVENTIVE.2016.7830159.

19. Daryabar F, Dehghantanha A, Choo KKR. Cloud storage forensics: MEGA as a case study. Australian Journal of Forensic Sciences. 2017;49(3):344-357. doi:10.1080/00450618.2016.1153714.

20. Ji Q, Rao Z, Ni L, Zhao W, Fu J. Vulnerability Analysis of MEGA Encryption Mechanism. CMC-COMPUTERS MATERIALS & CONTINUA. 2022;73(1):817-829. doi:10.32604/cmc.2022.026949.

21. MEGA Limited. MEGA Security White Paper Third Edition. https://mega.nz/SecurityWhitepaper.pdf.

22. Mega Limited. MEGA cloud SDK, SDK Release Version v4.28.0, https://github.com/meganz/sdk/releases/tag/v4.28.0

23. odwyersoftware. mega.py. https://github.com/odwyersoftware/mega.py.

24. Pailler G. MegaApiClient. https://github.com/gpailler/MegaApiClient.

25. Mega Limited. MEGA cloud SDK. https://github.com/meganz