Research on Note-Taking Apps with Security Features

Myungseo Park[1], Soram Kim[1], and Jongsung Kim[1,2*]
[1]Dept. of Financial Information Security, Kookmin University, 77 Jeongneung-Ro
Seongbuk-Gu, Seoul, 02707, Korea
{pms91, kimsr2040, jskim}@kookmin.ac.kr
[2]Dept. of Information Security, Cryptology and Mathematics, Kookmin University, 77
Jeongneung-Ro Seongbuk-Gu, Seoul, 02707, Korea
jskim.kookmin.ac.kr

Received: October 7, 2020; Accepted: December 21, 2020; Published: December 31, 2020

Abstract

Smartphone applications (apps) provide users with features to maximize the usefulness of smartphones in various categories, such as finance, education, health, life, and entertainment. For these features, apps store within themselves user data, which are closely related to their user. Such data can be thus used as key digital forensics clues. However, some apps use their own security features to protect data against external threats. Security features, which can effectively protect sensitive data, impose considerable digital forensics challenges that require data decryption to be used as evidence. Therefore, it is essential to conduct a preliminary study of apps with security features so that forensic investigators can perform their work efficiently. In this paper, we propose a forensic analysis of the note-taking apps ClevNote and Samsung Notes. Note-taking apps are valuable as evidence in forensic investigations because notes written by users are stored as app data, but forensic analysis is difficult as several security features protect app data. We conducted a study on a method to collect the protected app data in a form usable as evidence. To achieve this purpose, we identified the security features for target apps and obtained app data by revealing the operation process of security functions using reverse engineering.

Keywords: Smartphone application, Note-taking application, Access control, Data encryption/decryption

1 Introduction

1.1 Background

Smartphone applications (apps) store various user-related data to provide services. Apps often apply security features to protect the data, which are their primary target, against external threats. Apps use cryptographic algorithms as a crucial factor to play the role of security features. Security features are a powerful means of data protection, but they function as anti-forensics in digital investigations. For investigators to use app data as evidence, research on the security functions of each app should be conducted. An analysis for each app is required due to the characteristics of the apps that provide security functions using a unique scheme. In this paper, we focus on and analyze note-taking apps among the ones that provide security features. The main reason for this is that notes written by the user are stored as data thereby making note-taking app valuable as evidence. As first, we categorize the security features that

---

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), 11(4):63-76, Dec. 2020
DOI:*10.22667/JOWUA.2020.12.31.063*

[*] Corresponding author: Department of Information Security, Cryptology and Mathematics and Dept. of Financial Information Security, Kookmin University, 77 Jeongneung-Ro, Seongbuk-Gu, Seoul, 02707, Korea, Tel: +82-2910-4750

the note-taking apps can support and generalize the operation process for each security feature. Afterwards, a case study is performed for the forensic analysis by applying the security functions, which is defined by us, to the note-taking apps ClevNote and Samsung Notes.

1.2 Related Work

Research into smartphone apps with security features has been conducted for several years. Anglano (2014) analyzed WhatsApp, an instant messenger in the Android operating system [5]. This study analyzed the artifacts of WhatsApp and experimented with recovering deleted data. Awan (2015) researched Facebook, Twitter, and LinkedIn on Apple, Android, and Windows operating systems [8]. The author focused on artifacts for logical data acquired from each app. Anglano et al. (2016) analyzed ChatSecure, a secure instant messenger, and discovered the data encryption process based on user-entered password [6]. Frosch et al. (2016) details the cryptographic algorithms used in the authentication process and other protocols of the secure instant messenger TextSecure [13]. Azhar and Barton (2017) analyzed the security features for Wickr and Telegram, which are security messengers running on the Android operating system [9]. Rathi et al. (2018) conducted a research on the analysis of artifacts and decryption of encrypted data for the encrypted instant messaging applications Telegram, WhatsApp, Viber, and WeChat [19]. Sudozai et al.classified various events of call and chat related activities by detecting the traffic flow of the instant messenger IMO on the Android and iOS platforms [22]. Choi et al. (2019) researched the decryption of encrypted database files for KakaoTalk, NateOn, and QQ messenger in Windows operating system [11]. On the other hand, research on Telegram, an instant messenger, has been actively conducted over the past few years. Satrya et al. (2016) performed a log and packet analysis of normal and secret chat and artifact analysis, such as users, contacts, and chat content in Telegram on Android [21] [20]. Gregorio et al. (2017) aimed at a forensic analysis of Telegram for the Windows phone, which analyzed the data structure of Telegram and its major artifacts [14]. Anglano et al. (2017) revealed the unique data structure of Telegram and the possibility of recovering deleted chat data [7]. Giyoon et al. (2020) analyzed Telegram X and BBM Enterprise in mobile and desktop environments [16], which uncovered the decryption method for each app, and confirmed the possibility of decryption key recovery through memory analysis. In addition, studies on various apps, including Kik, Line, TikTok, and BiP, were conducted [17] [10] [15] [4].

1.3 Our Contribution

In this paper, we provide the forensic analysis results for ClevNote and Samsung Notes, which are note-taking apps with security features applied. Our contributions are summarized below.

i The security features of the note-taking apps are supported in various forms. Its features may be provided independently or in combination. We identified the supportable security features of the note-taking app and generalized the results of our reverse engineering analysis of the operation principle and process of the security features.

ii We analyzed ClevNote and Samsung Notes based on our generalized security features. Each app provides access control and data encryption as security features, but the method of providing the features is different. To clarify these methods, we used reverse engineering to reveal the operation process for this security feature.

iii Although the security feature analysis is completed, the artifact analysis is essential to use as evidence. We categorized the significant artifacts of ClevNote and Samsung Notes so that the analysis results can be used efficiently.

We analyzed each app in 5 categories: security features, data extration, password verification, data decryption, and artifact identification. Table 1 summarizes our analysis results for both apps.

Table 1: Summary of results for ClevNote and Samsung Notes

Category	ClevNote	Samsung Notes	Remark
Security feature	Service lock, data encryption	Note lock, data encryption	Lock password does affect data encryption
Data extraction	Android backup	Samsung Smart Switch backup	Android backup: Extract entire app data Smart Switch: Extract only the sdoc.db file
Password verification	Self-password verification	Authenticator verification	
Data decryption	Decryption base on fixed keys	Decryption based on a combination of three DeviceIDs in /shared_prefs/NotesDeviceInfo.xml	ClevNote: Use hard-coded key Samsung Notes: Use a private key stored in the Android KeyStore
Artifacts identification	Classify artifacts in esmemo.db	Classify artifacts in sdoc.db	

2 Security Features of a Note-Taking Apps

The note-taking app may be provided with security features to protect memos. These functions can be provided in various forms but are typically performed based on encryption. Identifying security features and analyzing the data encryption method are essential to obtain protected memos. In this section, we describe access control and data encryption, which are the types of security features supported by note-taking apps. Figure 1 illustrates the security features applied to note-taking apps.

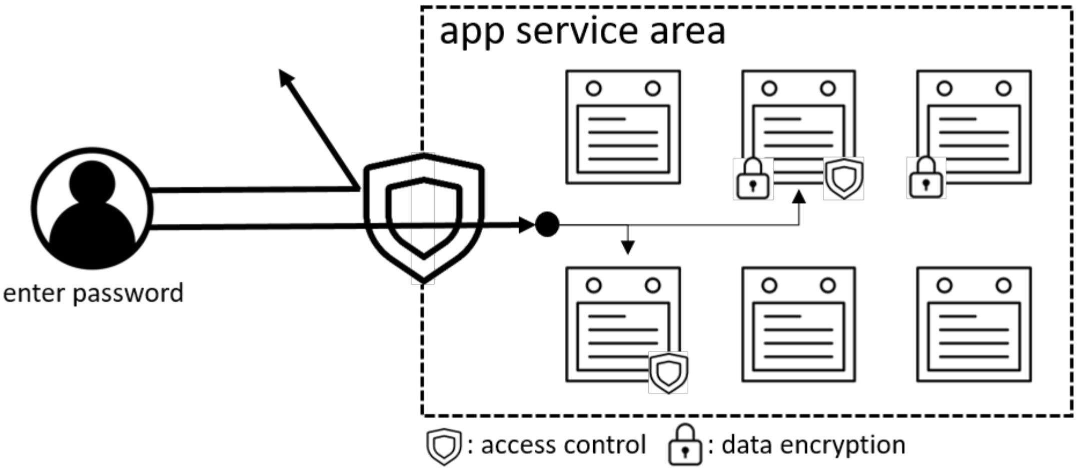


Figure 1: Security features applied to note-taking apps

Both security features can be applied alone or in duplicate. For example, if only access control is set, the protected notes can be accessed by password verification. However, the raw data of the notes remain in plaintext. Conversely, if only data encryption is set, the raw data of the protected notes remain in ciphertext, but the notes can be accessed without additional verification of the service. Notes can apply control access and store raw data as ciphertext by overlapping the two security features. In the last case, a key for encrypting the raw data is generated based on a password for access control. That is, the password for access control is used to encrypt raw data.

Access control. Access control, which allows only authorized users access to the service, usually determines whether to provide the service by authenticating the user-entered password. Access control of the note-taking apps can be provided in the form of a service lock and note lock. A service lock is activated at the perimeter of the app service to determine whether to provide the service based on the user-entered password. Similarly, the note lock locks specific notes based on a user-entered password.

Password-related data exist as app data to verify external the user-entered password and are stored differently depending on the password verification method. Figure 2 represents the verification method for a user-entered password.

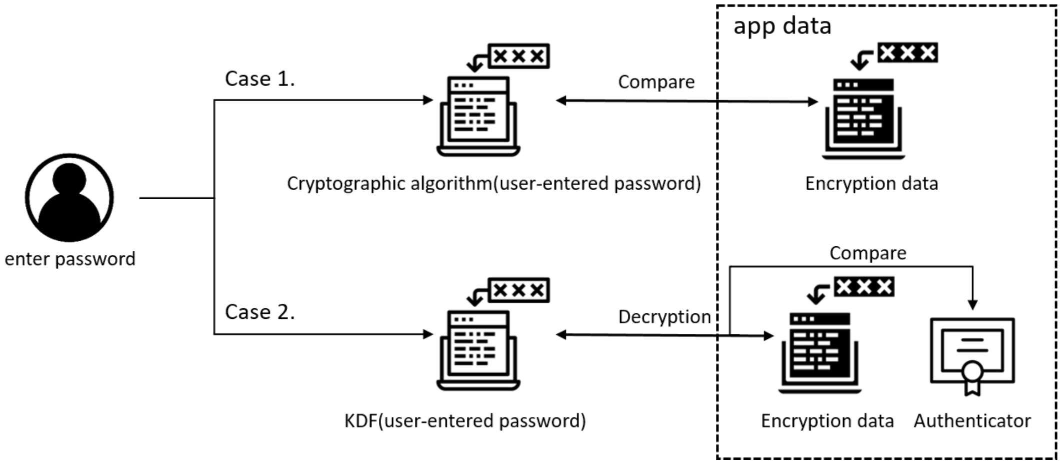


Figure 2: Verification methods for user-entered password

Password verification can be divided into two cases: self-password verification and authenticator verification. Self-password verification stores the correct password converted by a cryptographic algorithm in the app data. Afterward, this value is compared with the value converted from the user-entered password using the same cryptographic algorithm to perform verification. Unlike self-password verification, the authenticator verification does not store the password itself but stores an authenticator that can be verified based on a password. As an example of authenticator verification, the app converts the authenticator using an encryption key and cryptographic algorithm generated based on the correct password and stores it in the app data. Afterward, verification is performed by comparing whether the value obtained by decrypting the authenticator data encrypted with the decryption key generated based on the user-input password is the same as the authenticator.

Data encryption. Data encryption for data protection uses hash functions, such as the MD5 and SHA series, or block ciphers, such as DES and AES. The hash function, which is a one-way function, cannot restore the original data using the hash value. Due to the cryptographic characteristics of the hash function, the hash function can be used for password protection or verification. For example, the app stores the hash value for the correct password, and password verification can be performed by comparing the hash value of the externally entered password with the stored hash value. The block cipher encrypts or decrypts data in combination with the mode of operations, such as electronic codebook (ECB), cipher block chaining (CBC), cipher feedback (CFB), output feedback (OFB), and Counter (CTR). The block cipher combined with the mode of operation obtains encrypted data or decrypted data as output according to the purpose of encryption or decryption by entering target data, secret key, and IV (if needed, e.g., in the CBC or CFB or OFB or CTR mode of operation). The secret key can be obtained using the key derivation function (KDF) using a hardcoded fixed value or user-entered password as an input parameter. In the data cryptographic process, secret key generation and cryptographic execution are a flow. Therefore, to decrypt encrypted data, it is necessary to simultaneously identify the KDF used to generate the secret key and cryptographic algorithm used to perform the data encryption.

3 Analysis of note-taking apps

In this section, we describe the results of our analysis of the note-taking apps ClevNote and Samsung Notes in the Android operating system. As of December 2020, we analyzed the latest versions of ClevNote (version 2.21.0) and Samsung Notes (version 4.1.03.1). We identified the security features of each app and analyzed the method of extracting the app data and decrypting the encrypted data. To use the app data, it is essential to first extract the data from a smartphone. A typical method of extracting app data is to access the app data of a rooted smartphone directly. However, in terms of forensics, where integrity preservation is a significant issue, the technique of extracting data without additional measures, such as rooting, is most important. The data extraction method for our analysis target app is different, but data extraction is possible without rooting. We performed reverse engineering to reveal the decryption method for each app. We performed static and dynamic analyses using various analytical tools. We used the JEB Decompiler [3] to perform a static analysis of the app file. The app file *.apk contains the bytecode, library, and resource files. In the bytecode, the names of the Java object, functions, and variables are obfuscated by replacing them with letters, but the function names called by the library are preserved. We used IDA Pro [2] to perform a dynamic analysis for debugging to check the changes in the variables running the app.

3.1 ClevNote

ClevNote, which has more than 5 million downloads based on the Google Play store, is a note-taking app that manages notes for various purposes, such as bank account numbers, checklists, site IDs, and plaintext memos. In this section, we describe the results of our analysis of ClevNote.

Security features. ClevNote provides access control to app services and encryption of app data as security features. Access control of ClevNote is performed by verifying a four-digit password. The user setting activates this feature, and the number of attempts for password verification has no limit. ClevNote stores password-related information in the com.dencreak.esmemo_preference.xml file in the shared_prefs folder when user sets a password. Figure 3 lists the information related to the password for the service lock on ClevNote. If the value of 'ispassword' is true, the service lock is set. Conversely, if the value is false, the service lock is not set.

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
- <map>
    <string name="IAZPPTAPS_TPBR">MjAyMCwwNCwwNCx3aHlyZWQsMDUsMTg= </string>
    <boolean name="ispassword" value="true"/>
    <string name="fbconfig_and_init_check_continuously">false</string>
    <string name="EncryptMainPass">3CYR5qTf+sIkoRUYfVpO0A== </string>
    <string name="IARPATZPS_ITHD">ZmFsc2U= </string>
    <string name="IGML_and_inter_tmedit_gapmin">1585880348574</string>
    <string name="fbconfig_and_rewarded_durationmin">1000</string>
    <string name="IAZLASPTS_TPBR">dHJ1ZQ== </string>
    <string name="App_Execute_First">1585880329135</string>
    <string name="fbconfig_and_menu_forceopenday">7 / 60</string>
    <string name="chl_ctadd">2</string>
```

/shared_prefs/com.dencreak.esmemo_preference.xml

Service lock of ClevNote

Figure 3: Information related to the password for service lock on ClevNote

App data encryption, another security feature of ClevNote, targets the service lock password and note content. The encrypted service lock password is stored as a value of the attribute "EncryptMainPass," as depicted in Figure 3. The note content is stored in the esmemo.db file, and the specific column data for each table are encrypted. Table 2 summarizes the columns in which encrypted notes for ClevNote are stored.

Table 2: Classification of encrypted column app data on ClevNote

Package name	File path	Table name	Column name
com.dencreak.esmemo	databases\esmemo.db	accountdatum	a_name, a_bank, a_number, a_holder, a_memo, a_protect
		birthdaydatum	b_name, b_memo, b_protect
		cartdatum	c_name, c_memo, c_protect
		folderdatum	f_name, f_protect
		siteiddatum	s_name, s_address, s_siteid, s_memo, s_protect
		textmemodatum	t_subject, t_body, t_protect

Data extraction. Regarding data extraction, the app data for ClevNote can be extracted using Android backup [1]. We confirmed that the value of android:allowBackup, which is a backup-related setting, is true in the AndroidManifest.xml file that contains various configuration information for ClevNote. The android:allowBackup value is true or false, and the data extraction using the Android backup is possible only if the value is true. We extracted the app data for ClevNote by performing an Android backup through the following adb command.

- adb command: `adb backup com.dencreak.esmemo -f backup.esmemo.ab`

The Android backup file is packed in its own way, so unpacking is essential to obtain the data. We unpacked this file using the Android backup extractor developed by Nikolay Elenkov [12]. We use Android backup to extract all the data in the app package, including databases, settings, and resource files.

Password verification. Concerning password verification, the service lock provided by ClevNote uses self-verification performed using an encrypted service lock password stored in the app data. If the user-input password is the same as the decrypted value of the encrypted service lock password, verification is completed. The encrypted service lock password is stored as the value of the attribute "EncryptMainPass" in com.dencreak.esmemo_preference.xml. The decryption method of the encrypted service lock password is shown in Eq. (1):

$$P = \text{AES256-CBC-DECRYPT}(C, MK, IV). \tag{1}$$

In Eq. (1), the parameters C, MK, and IV of AES256-CBC-DECRYPT use an encrypted service lock password, a 32-byte fixed value FK_1, and a 16-byte fixed value fIV_1, respectively. In addition, FK_1 and fIV_1 are hardcoded values in the ClevNote source code, which we revealed through the reverse engineering analysis. As a result of this equation, we can obtain the service lock password. The decryption of the service lock password that is not involved in other data encryption on ClevNote may not be the main consideration. However, this password, which has the advantage of being relatively easy to obtain, can be very useful in social engineering aspects in other application environments. For example, this password could be used by other apps or could be a clue to password recovery.

Data decryption. In this subsection, we describe how to decrypt note content, which is the encryption target of ClevNote. We classified the encrypted notes as presented in Table 2 in the previous section. The encrypted data are stored in column units, and the column name uses the first letter of the table as a separator. For example, the column names of the account table are defined as a_name, a_bank, and a_memo with 'a_' as a separator. The encrypted note content is included in the columns, except for *_protect, among the columns we classified in Table 2. The character * is a separator. The *_protect column data are used to obtain a decryption key DK to decrypt the encrypted column data in the same row. We explain the decryption of the encrypted column data using *_protect. The decryption process consists of obtaining the decryption key DK and decrypting the encrypted data. Decrypt *_protect is first necessary to obtain the decryption key DK. Its decryption is possible using Eq. (1). The parameters C, MK, and IV of AES256-CBC-DECRYPT use a value of the *_protect column, a 32-byte fixed value FK_2, and a 16 bytes fixed value fIV_1, respectively. The plaintext P is output as a string in the form $d_0|d_1|...|d_{11}$ ($0 \leq i \leq 11, 0 \leq d_i \leq 99$). We separate the integers from this string sequentially and assign them to the array *arr* of size 12. Then, we input the array *arr* in the decryption key acquisition function getKey to obtain the decryption key DK. Algorithm 1 represents the detailed process of the getKey function.

Algorithm 1: ClevNote decryption key acquisition algorithm

```
Function getKey (arr);
Input  : Array arr of integers of size 12
Output: Decryption key DK
if arr[3] mod 2 !=0 then
    if arr[11] mod 2 == 0 then
        R = arr[6];
    else
        R = arr[5];
    end
else if arr[11] mod 2 == 0 then
    R=arr[9];
else
    R=arr[7];
end
EncryptedDK = Base64Decoding(KeyArr[R]);
DK = AES256-CBC-Decrypt(C(=EncryptedDK), MK(=FK_2), IV(=fIV_1))
Return(DK);
```

In Algorithm 1, R is determined using a conditional expression with the array *arr* ($0 \leq R \leq 99$). Moreover, R indicates the index of a value to be used as a decryption key in the array *KeyArr* in which 100 encrypted key candidates are stored. In addition, $KeyArr[i]$ is a base64-encoded value of encrypted decryption key candidates ($0 \leq i \leq 99$). We obtain $EncryptedDK$, which is the value of base64 decoded $KeyArr[R]$, and then decrypt it using AES256-CBC-DECRYPT. Except for changing ciphertext C to $EncryptedDK$, the parameters of AES256-CBC-DECRYPT are the same as those of the decryption of *_protect. Finally, we decrypted the encrypted column data of the same row using the obtained decryption key DK. Table 3 summarizes the decryption methods of the encrypted data on ClevNote.

Table 3: Summary of decryption targets and decryption methods on ClevNote

Decryption target	Decryption Algorithm	Input parameters	Output parameter
The value of EncryptMainPass	V=Base64Decoding(V) P=AES256-CBC-DECRYPT(C, MK, IV)	C: Decryption target MK: FK_1 IV : fIV_1	P= service lock password
*_protect column data		V: Decryption target MK: FK_2 IV : fIV_1	P= string to find the index R of *KeyArr*
$KeyArr[R]$		V: Decryption target MK: FK_2 IV: fIV_1	P= the decryption key DK
Encrypted column data in the same row as the corresponding *_protect column data		V: Decryption target MK: DK IV: fIV_1	P= decrypted column data

Artifact Identification. ClevNote manages memo data in the esmemo.db file, and stores the data according to the supported memo types such as account, birthday, checklist, site ID, and text memo in the table units. Encrypted note content is essential to be decrypted for use as digital evidence. Table 4 categorizes the major artifacts based on the data in the esmemo.db file that we decrypted.

Table 4: Artifacts in esmemo.db of ClevNote

Table	Column	Content
accountdatum	a_name	Account name
	a_bank	Bank name
	a_number	Account number
	a_holder	Account holder
	a_memo	Account related notes
birthdaydatum	b_name	Birthday person's name
	b_memo	Birthday related notes
cartdatum	c_name	Checklist item
	c_status	Checklist status (true or false)
	c_memo	Checklist related notes
siteiddatum	s_name	Site name
	s_address	Site address
	s_siteid	Site ID
	s_memo	Site ID related notes
textmemodatum	t_subject	Title for the note
	t_body	Note content
	t_edttime	Note-taking time

3.2 Samsung Notes

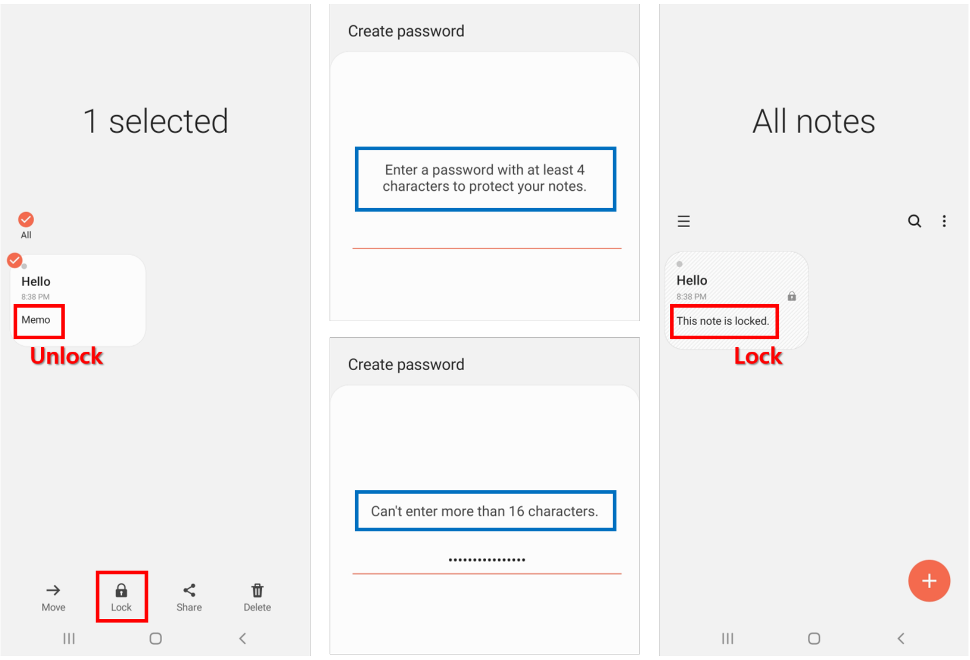
Samsung Notes can save available text, pictures, drawings, voice records, and files. It manages the notes with each directory made by the user. It can also synchronize with other devices and share through SNS applications. The latest version is 4.1.03.1, and the number of downloads is over 500 million based on the Google Play store.

Security features. Samsung Notes provides a note lock as shown in Figure 4, and the data are encrypted. The lock is available only if logging in using a Samsung account. A password for the lock can be 4 to 16 digits long and include numbers, letters, and special characters. The password is only used to verify a user and is not related to data encryption.

We identified whether the memo is locked or not through the flag as shown in Figure 5. The flag is saved in the isLock and the ContentSecureVersion column of the sdoc table in sdoc.db. If the flag is set to 1, the memo is locked. When the flag is set to 0, the memo is unlocked.

Table 5 summarizes the columns in which encrypted notes for Samsung Notes are stored. Samsung Notes encrypts data in the content, strippedContent, and displayContent columns of the sdoc table in the sdoc.db file.

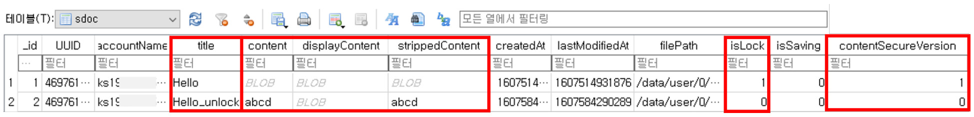
Data extraction. Unlike ClevNote, Samsung Notes cannot extract app data using Android backup because the android:allowBackup value is false. If data extraction using Android backup is not possible, the only way to directly access app data located in the data partition is to obtain administrator permission through rooting. Compared to rooting, which cannot avoid data tampering, extracting data from unrooted



Enter a password with at least 4 characters to protect your notes.

Can't enter more than 16 characters.

Figure 4: Information related to the password for service lock on Samsung Notes



	_id	UUID	accountName	title	content	displayContent	strippedContent	createdAt	lastModifiedAt	filePath	isLock	isSaving	contentSecureVersion
	…	필터	필터	필터	필터	필터	필터	필터	필터	필터	필터	필터	필터
1	1	469761…	ks1[redacted]…	Hello	BLOB	BLOB	BLOB	1607514…	1607514931876	/data/user/0/…	1	0	1
2	2	469761…	ks1[redacted]…	Hello_unlock	abcd	BLOB	abcd	1607584…	1607584290289	/data/user/0/…	0	0	0

Figure 5: Lock flag depending on whether lock or not

Table 5: Classification of encrypted column app data on Samsung Notes

Package name	File path	Table name	Column name
com.samsung.android.app.note	databases\sdoc.db	sdoc	content, strippedContent, displayContent

Android devices is more forensically effective. For this reason, we used Samsung Smart Switch as an app data extraction method. Samsung Smart Switch is a dedicated program for data backup of Samsung smartphone. Samsung Smart Switch back up contacts, text messages, call logs, multimedia data, settings, and app data as well as Samsung Notes as separate item. Backup items excluding multimedia data are encrypted, but we were able to decrypt them using the tool developed in our previous study [18]. This method cannot extract all data in the app package like Android backup, but it is possible to extract the sdoc.db file containing the notes.

Password Verification. The password is required when the user locks the notes, and it is used to determine the access authentication. It is unique and applies to all notes. When creating a password, the encrypted password and salt are stored in com.samsung.android.app.notes_preferences.xml and UserAuthInfo.xml in the shared_prefs as shown in Figure 6. The encrypted password is saved as element contents with the element name NotesPasswordHash_enc, and the encrypted salt is saved as element contents with the element name NotesPasswordSalt_enc.

The iteration count for PBKDF2withHMACSHA1 is 4,000, and the key length is 256 bytes. The Android KeyStore manages the key for RSA/ECB/OAEPWithSHA-256AndMGF1Padding. Each value

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
- <map>
    <int name="UNLOCK_TRY_COUNT" value="0"/>
    <string
      name="PrefPasswordEncryptedHashBackup">06355d61391393dfb4abce9dc578b26
    <boolean name="AvailableIris" value="true"/>
    <string name="local_password_owner">rcgqq4iqdg</string>
    <string name="PrefPasswordOwnerBackup">rcgqq4iqdg</string>
    <string name="PrefPasswordSaltBackup"/>
    <string name="DEVICE_BUILD_MANUFACTURER">Samsung</string>
    <long name="BlockEndTime" value="0"/>
    <string
      name="PrefPasswordEncryptedSaltBackup">4fed86965690fe4c7cc4baeb502f73229
    <long name="OldBiometricMethodBlockEndTime" value="0"/>
    <boolean name="AvailableFingerprint_SEC" value="true"/>
    <string name="PrefPasswordHadhBackup"/>
</map>
```

/shared_prefs/com.samsung.android.app.notes_preferences.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<map>
    <string
      name="NotesPasswordHash_enc">06355d61391393dfb4abce9dc578b26
    <string
      name="NotesPasswordSalt_enc">4fed86965690fe4c7cc4baeb502f73229
</map>
```

/shared_prefs/UserAuthInfo.xml

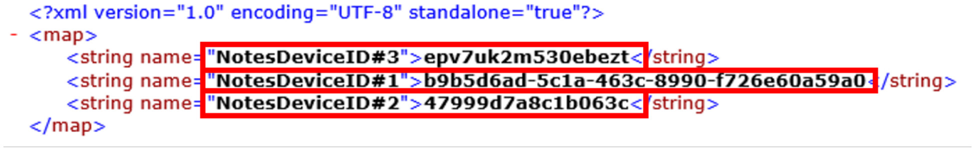
Figure 6: The factors for password verification

saved in the XML file is derived as shown in Eq.(2):

$$
\begin{aligned}
PBEkey &= \text{PBKDF2withHMACSHA1}(password, salt, iteration, keylength),\\
NotesPasswordHash\_enc &= \text{RSA/ECB/OAEPWithSHA-256AndMGF1Padding}(key, PBEkey),\\
NotesPasswordSalt\_enc &= \text{RSA/ECB/OAEPWithSHA-256AndMGF1Padding}(key, salt).
\end{aligned}
\tag{2}
$$

The app uses the information above to verify the password when the user accesses the note. The first step is decrypting NotesPasswordHash_enc and NotesPasswordSalt_enc to obtain NotesPasswordHash and NotesPasswordSalt. The next step is deriving the PBEkey with the user-enter password and NotesPasswordSalt. Finally, if NotesPasswordSalt and PBEkey are equal, verification is successful, and it is available to access the note.

Data decryption. The decryption method for encrypted note data is the same as Eq. (1) used in ClevNote. The *IV* is 16 bytes and is fixed at 0. The 32-byte *MK* for data decryption can be derived the three items of information in NotesDeviceInfo.xml in /shared_prefs/ folder. Each piece of information is described in Figure 7.



```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
- <map>
    <string name="NotesDeviceID#3">epv7uk2m530ebezt</string>
    <string name="NotesDeviceID#1">b9b5d6ad-5c1a-463c-8990-f726e60a59a0</string>
    <string name="NotesDeviceID#2">47999d7a8c1b063c</string>
</map>
```

/shared_prefs/NotesDeviceInfo.xml

Figure 7: Information for generating encryption key

This information is set up when the user creates a note for the first time. NotesDeviceID#1 is a randomly chosen UUID. NotesDeviceID#2 is an Android ID. NotesDeviceID#3 is a randomly chosen

16-byte string. This decryption method can decrypt the encrypted column data that we identified in Table 5. Table 6 summarizes the information in which the encrypted notes and password for Samsung Notes are stored.

Table 6: Summary of decryption targets and decryption methods on Samsung Notes

Decryption target	Decryption Algorithm	Input parameters	Output parameter
content, strippedContent	P=AES256-CBC-DECRYPT(C, MK, IV)	C: Decryption target MK: *key* IV : 0	P= Note data (String, excluded new line characters)
displayContent			P= Note data (BLOB, included new line characters)
NotesPasswordHash_enc,	P=RSA/ECB/OAEPwithSHA-256andMGF1(C,MK)	C=Decryption target MK=KeyStore privatekey	P = NotesPassworHash
NotessPasswordSalt_enc	P=RSA/ECB/OAEPwithSHA-256andMGF1(C,MK)	C=Decryption target MK=KeyStore privatekey	P = NotesPasswordSalt

Artifact Identification. Table 7 lists the artifacts in sdoc.db of Samsung Notes. Most of the data are contained in sdoc and the category_tree table. When a note is locked, only the content is encrypted, not the title. The content is saved in the content, displayContent, and stippedContent columns. The content and strippedContent columns the save same data and consist of a string excluding a new line character. The displayContent is configured as Binary large object (BLOB) including a new line character. The contentUUID is filled if the memo contains image, voice recordings, or audio files. The notes can be classified by creating categories. The UUID per category is saved in the categoryUUID column in the sdoc table and the UUID column in the category_tree table. The category name is saved in UUID column in the category_tree table. Therefore, we can determine the category name in the displayName column in the category_tree by matching the UUID and categoryUUID columns. The isDeleted column contains the flag concerning whether the note is deleted. Even if the note is deleted, only the flag changes, and the data appear in the database. The flag in the sdoc table is set to 1 when the note is deleted, but the flag in the category_tree is set to 2 when the category is deleted. The create time and last modified time of the note and category are saved in the creatAt and lastModifiedAt columns, respectively.

4 Discussion and Research Challenges

In this paper, we generalized the security features of note-taking apps and performed digital forensic analysis of ClevNote and Samsung Notes as a case study. Note-taking apps implement security features such as access control and data encryption into their own scheme. Since security features interfere with the use of app data in digital forensic, detailed analysis of security schemes for each app is required. Even if the app analysis is sufficient, it may not be practically utilized in digital forensics unless data collection is preceded. Therefore, we considered extraction and analysis for each app. ClevNote can extract all app data using Android backup. This means that our app analysis results can be sufficiently utilized. For example, among the extracted files, com.dencreak.esmemo_preference.xml can be used for service lock password recovery. On the other hand, since Samsung Notes cannot extract data using Android backup, there are restrictions on using the results of our app analysis. Samsung Smart Switch extracts only the sdoc.db file containing crucial artifacts from the Samsung Notes app data. This means that the NotesDeviceInfo.xml file, which is essential for decryption of encrypted locked notes, cannot be obtained. Although the analysis results of Samsung Notes may not be able to be used for normal smartphones, there is a possibility of using them in digital forensic investigations that can face various situations. For example, a rooted smartphone that can access app data can extract data necessary for analysis from the app data of Samsung Note. However, these samples are only a few. Therefore, research

Table 7: Artifacts in sdoc.db of Samsung Notes

Table	Column	Content
sdoc	accountName	Samsung account ID
	UUID	UUID per note
	categoryUUID	UUID per Category
	title	Note title(String)
	displayTitle	Note title(BLOB data)
	content	Note content(String)
	strippedContent	Note content(String)
	displayContent	Note content(BLOB data)
	size	Size of note
	createAt	Note created time
	lastModifiedAt:	Last Note modified time
	isLock	Flag (Lock: 1, Unlock: 0)
	ContentSecureVersion	Flag (Lock: 1, Unlock: 0)
	isDeleted	Flag (Deleted: 2, Not deleted: 0)
	recycle_time_bin_moved	Note deleted time
category_tree	UUID	UUID per directory
	displayName	Directory name
	createAt	Category created time
	lastModifiedAt	Category last modified time
	isDeleted	Flag (Deleted: 1, Not deleted: 0)

on data extraction from normal smartphones is indispensable in order to more effectively use digital forensics for apps that cannot use Android backup, such as Samsung Note.

5 Conclusion

Smartphone apps are important in digital forensic investigations because they can be used to obtain critical evidence. However, some apps apply security features to protect their data. Therefore, it is necessary to analyze various apps to which security features are applied for digital forensic investigation. In this paper, we generalized the security features for note-taking apps and analyzed the latest versions of ClevNote and Samsung Notes as a case study. We believe that our work enables efficient analysis of smartphone backup data and will significantly affect future forensic investigations.

Acknowledgments

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2017-0-00520, Development of SCR-Friendly Symmetric Key Cryptosystem and Its Application Modes).

References

[1] Android backup manager. `https://developer.android.com/reference/android/app/backup/BackupManager`,[Online; Accessed on December 1, 2020].

[2] IDA pro. `https://www.hex-rays.com/products/ida/index.shtml` [Online; Accessed on December 1, 2020].

[3] Jeb decompiler by pnf software. `https://www.pnfsoftware.com` [Online; Accessed on December 1, 2020].

[4] E. Akbal, I. Baloglu, T. Tuncer, and S. Dogan. Forensic analysis of bip messenger on android smartphones. *Australian Journal of Forensic Sciences*, 52(5):590–609, September 2020.

[5] Anglano and Cosimo. Forensic analysis of whatsapp messenger on android smartphones. *Digital Investigation*, 11(3):201–213, September 2014.

[6] C. Anglano, M. Canonico, and M. Guazzone. Forensic analysis of the chatsecure instant messaging application on android smartphones. *Digital investigation*, 19:44–59, December 2016.

[7] C. Anglano, M. Canonico, and M. Guazzone. Forensic analysis of telegram messenger on android smartphones. *Digital Investigation*, 23:31–49, December 2017.

[8] Awan and F. Ali. Forensic examination of social networking applications on smartphones. In *Proc. of the 2015 Conference on information assurance and cyber security (CIACS'15), Rawalpindi, Pakistan*, pages 36–43. IEEE, December 2015.

[9] M. B. Azhar and T. E. A. Barton. Forensic analysis of secure ephemeral messaging applications on android platforms. In *Proc. of the 11th International Conference on Global Security, Safety and Sustainability (IGCS3'17), London, UK*, volume 630 of *Communications in Computer and Information Science*, pages 27–41. Springer, Cham, January 2017.

[10] M. S. Chang and C. Y. Chang. Forensic analysis of line messenger on android. *Journal of Computers*, 29(1):11–20, February 2018.

[11] J. Choi, J. Yu, S. Hyun, and H. Kim. Digital forensic analysis of encrypted database files in instant messaging applications on windows operating systems: Case study with kakaotalk, nateon and qq messenger. *Digital Investigation*, 28:50–59, April 2019.

[12] N. Elenkov. Android backup extractor. `https://github.com/nelenkov/android-backup-extractor` [Online; Accessed on December 1, 2020].

[13] T. Frosch, C. Mainka, C. Bader, F. Bergsma, J. Schwenk, and T. Holz. How secure is textsecure? In *Proc. of the 2016 IEEE European Symposium on Security and Privacy (EuroS&P'16), Saarbrucken, Germany*, pages 457–472. IEEE, March 2016.

[14] J. Gregorio, A. Gardel, and B. Alarcos. Forensic analysis of telegram messenger for windows phone. *Digital Investigation*, 22:88–106, September 2017.

[15] N. H. Khoa, P. T. Duy, H. Do Hoang, V.-H. Pham, et al. Forensic analysis of tiktok application to seek digital artifacts on android smartphone. In *Proc. of the RIVF International Conference on Computing and Communication Technologies (RIVF'20), Ho Chi Minh, Vietnam*, pages 1–5. IEEE, July 2020.

[16] G. Kim, M. Park, S. Lee, Y. Park, I. Lee, and J. Kim. A study on the decryption methods of telegram x and bbm-enterprise databases in mobile and pc. *Forensic Science International: Digital Investigation*, 35:1–12, December 2020.

[17] K. M. Ovens and G. Morison. Forensic analysis of kik messenger on ios devices. *Digital Investigation*, 17:40–52, June 2016.

[18] M. Park, O. Yi, and J. Kim. A methodology for the decryption of encrypted smartphone backup data on android platform: A case study on the latest samsung smartphone backup system. *Forensic Science International: Digital Investigation*, 35, December 2020.

[19] K. Rathi, U. Karabiyik, T. Aderibigbe, and H. Chi. Forensic analysis of encrypted instant messaging applications on android. In *Proc. of the 6th International Symposium on Digital Forensic and Security (ISDFS'18), Antalya, Turkey*, pages 1–6. IEEE, May 2018.

[20] G. B. Satrya, P. T. Daely, and M. A. Nugroho. Digital forensic analysis of telegram messenger on android devices. In *Proc. of the 2016 International Conference on Information and Communication Technology and Systems (ICTS'16), Surabaya, Indonesia*, pages 1–7. IEEE, October 2016.

[21] G. B. Satrya, P. T. Daely, and S. Y. Shin. Android forensics analysis: Private chat on social messenger. In *Proc. of the 8th International Conference on Ubiquitous and Future Networks (ICUFN'16), Vienna, Austria*, pages 430–435. IEEE, July 2016.

[22] M. Sudozai, S. Saleem, W. J. Buchanan, N. Habib, and H. Zia. Forensic study of imo call and chat app. *Digital investigation*, 25:5–23, June 2018.

Author Biography

Myungseo Park received his Bachelor degree in Mathematics and his Master degree in Financial Information Security from Kookmin university, Korea in 2013 and 2015, respectively. He had been a Researcher of National Security Research Institute (NSR), Korea, from December 2014 till February 2017, He has been a PhD candidate of Dept. of Financial Information Security at Kookmin University, Korea, since March 2017. His research interests are symmetric cryptography and digital forensic.

Soram Kim received her Bachelor degree in Mathematics and Master degrees in Financial Information Security from Kookmin university, Korea in 2016 and 2018, respectively. She is currently studying for Doctor's degree in Dept. of Financial Information Security at Kookmin University, Korea, since March 2018. She is a researcher in the DF&C (Digital Forensic & Cryptanalysis) Laboratory. Her research interests include information security and digital forensics.

Jongsung Kim received his Bachelor and Master degrees in Mathematics from Korea university, Korea in 2000 and 2002, respectively. He received double Doctoral degrees completed in November 2006 and February 2007 at the ESAT/COSIC group of Katholieke Universiteit Leuven and at Engineering in Information Security of Korea University, respectively. He had been a Research Professor of Center for Information Security Technologies (CIST) at Korea University, Korea, from March 2007 till August 2009, and an assistant professor of department of e-business at Kyungnam University, Korea, from September 2009 till February 2013. Dr. Kim has been an associate professor of Dept. of Information Security, Cryptology, and Mathematics / Dept. of Financial Information Security at Kookmin University, Korea, from March 2013 till August 2020. He has been a full professor at the same departments since September 2020. Dr. Kim has published more than 60 research papers in international journals, conferences and books. His research interests include security issues, cryptography, and digital forensics.