Certificate Injection-based Encrypted Traffic Forensics in AI Speaker Ecosystem

By:
Yeonghun Shin (Ajou University), Hyungchan Kim (Ajou University), Sungbum Kim (Ajou University), Dongkyun Yoo (Ajou University), Wooyeon Jo (Ajou University), and Taeshik Shon (Ajou University)

From the proceedings of
The Digital Forensic Research Conference
DFRWS USA 2020
July 20 - 24, 2020

Forensic Science International: Digital Investigation 33 (2020) 301010

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

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

DFRWS 2020 USA — Proceedings of the Twentieth Annual DFRWS USA

Certificate Injection-Based Encrypted Traffic Forensics in AI Speaker Ecosystem

Yeonghun Shin [a], Hyungchan Kim [a], Sungbum Kim [a], Dongkyun Yoo [a], Wooyeon Jo [a], Taeshik Shon [a, b, *]

[a] Department of Computer Engineering, Ajou University, World cup-ro 206, Suwon, 16499, Republic of Korea
[b] Department of Cyber Security, Ajou University, World cup-ro 206, Suwon, 16499, Republic of Korea

ARTICLE INFO

Article history:

Keywords:
AI Speaker
Certificate injectiion
MitM
Cloud
Amazon alexa
KT GiGA genie
SKT NUGU

ABSTRACT

AI Speakers are typical cloud-based internet of things (IoT) devices that store a variety of information regarding users on the cloud. Although analyzing encrypted traffic between these devices and the cloud, as well as the artifacts stored there, is an important research topic from the perspective of cloud-based IoT forensics, studies on directly analyzing encrypted traffic between AI Speakers and the cloud remain insufficient. In this study, we propose a forensic model that can collect and analyze encrypted traffic between an AI Speaker and the cloud based on a certificate injection. The proposed model consists of porting AI Speaker image on Android device, porting AI Speaker image using QEMU (Quick EMUlator), running exploit using the AI Speaker app vulnerability, rewriting Flash memory using H/W interface, and reworking and updating Flash memory. These five forensic methods are used to inject the certificate into AI Speakers. The proposed model shows that we can analyze encrypted traffic against various AI Speakers such as an Amazon Echo Dot, Naver Clova, SKT NUGU Candle, SKT NUGU, and KT GiGA Genie, and obtain artifacts stored on the cloud. In addition, we develop a verification tool that collects artifacts stored on KT GiGA Genie cloud.

1. Introduction

AI Speakers are rapidly increasing in usage and penetration every year. Indeed, AI Speakers are built into hotels and cars, and as Voicebot claimed, “In 2019, the number of adults using AI Speakers in the United States is one-quarter of the total adult population” (Voicebot, 2019). As the usage and penetration rate of AI Speakers that store various kinds of information regarding users increases, the importance of digital forensics increases; various studies are being conducted on the collection of artifacts from such devices. A study on the collection of artifacts stored in Amazon Echo was conducted (Chung et al., 2017), and in another study, encrypted traffic between a smartphone connected with an AI Speaker and the cloud were collected and analyzed (Jo et al., 2019). As a result of this study, it was found that the AI Speaker's cloud stores meaningful artifacts from a forensic perspective. However, there are no prior studies that analyzed encrypted traffic between AI Speakers and cloud. Therefore, a study on analyzing encrypted traffic with the AI Speaker is necessary.

AI Speakers primarily communicate with the cloud, and the communication is encrypted using Transport Layer Security (TLS) by default. As known from the previous study on encrypted traffic analysis between smartphone and the cloud of AI Speaker, the information stored in the cloud may include the voice command history, connected device information, and location information obtained using AI Speakers, which can be used as direct evidence in an forensic investigation. This is extremely important from a forensics perspective. If a forensic investigator does not know this information, important evidence could be ignored. Therefore, digital forensic investigators must be able to conduct an analysis on encrypted traffic between AI Speaker and cloud.

A Man-in-the-Middle (MitM) is thus required to analyze encrypted traffic between AI Speaker and cloud. MitM can be conducted using published HTTP proxy tools. To do so, the certificate of the HTTP proxy tool must be installed on the AI Speaker. However, unlike Android smartphones, the AI Speaker does not provide an Android UI, and thus it is difficult to install the certificate. However, even with this difficulty, we propose five methods

* Corresponding author. Department of Computer Engineering, Ajou University, World cup-ro 206, Suwon, 16499, Republic of Korea.
E-mail address: tsshon@ajou.ac.kr (T. Shon).

https://doi.org/10.1016/j.fsidi.2020.301010

for injecting the certificate into AI Speakers. Furthermore, based on this, we propose a digital forensic model for analyzing the encrypted traffic between an AI Speaker and cloud.

The contribution of this paper is to propose a TLS traffic analysis forensic model with five detailed certificate injection scenarios. Based on the digital forensic model proposed herein, we analyze the encrypted traffic between an AI Speaker and the cloud to determine what information is transmitted between them. Furthermore, it is possible to derive user information stored in the cloud by analyzing the encrypted traffic that the cloud sends to the AI Speaker.

The remainder of this paper is organized as follows. Section 2 discusses digital forensic studies conducted on AI Speakers and studies related to techniques for an encrypted traffic analysis. Section 3 introduces proposed encrypted traffic forensic model. Section 4 describes the five methods for injecting the certificate into AI Speakers of the proposed forensic model through a case study. Section 5 discusses the main artifact analysis results and validation tool that communicates directly with the cloud to collect artifacts stored there. Section 6 discusses the limitations and implications of this study. Finally, Section 7 discusses the analysis results and future work.

2. Related research

As the penetration rate of AI Speakers and IoT devices has increased, numerous forensic studies have been conducted. 'Chung et al.' analyzed the cache structure stored in Alexa devices to derive significant artifacts (Chung et al., 2017). 'Jo et al.' analyzed the TLS traffic between the cloud and the smartphone connected to the AI Speaker to derive artifacts stored in the cloud (Jo et al., 2019). 'Li et al.' proposed a digital forensic investigation model for AI Speakers and IoT devices (Li et al., 2019). The existing AI Speaker studies didn't focus on analyzing TLS traffic between AI Speaker and cloud.

As mentioned earlier, AI Speakers are cloud-based. Because the communication between the cloud and AI Speaker basically uses TLS encryption, it is difficult to apply forensics using a conventional technique in a cloud environment. Lin et al. solved this problem by using the MitM technique in conducting cloud-based network forensics (Lin et al., 2015). To apply MitM, it is common to intercept the certificate, as demonstrated by Soghoian et al. (Soghoian and Stamm, 2012). In addition, as shown by Callegati et al. and Burkholder, a vulnerability can be exploited for encrypted packets using the HTTPS protocol (Callegati et al., 2009; Burkholder, 2002). However, the latest Android OS supports SSL pinning by default and can be supported at the app level, making it difficult to execute MitM. To address this issue, studies on bypassing SSL pinning have been conducted. Ramírez-López et al. investigated the applicability of MitM to Android apps using five known SSL pinning bypass technologies and two newly proposed technologies (Ramírez-López et al., 2019).

Both mobile and IoT devices use NAND Flash as a storage device. Because AI Speakers use NAND Flash like other mobile devices, it is important to acquire data from the NAND Flash without damaging the data. To this end, various studies have been conducted on a Chip-off technique for physically separating NAND Flash from the PCB of a mobile device or an IoT device to acquire data. Aya et al. studied the effects of NAND Flash and retention aging under the short-term exposure of NAND Flash to high temperatures during a Chip-off. Thus, they proposed a data recovery method for NAND Flash damaged by heat (Fukami et al., 2017). In a later study, Aya et al. studied the chemical properties of submerged mobile devices and proposed a forensic method applicable to such devices (Fukami and Nishimura, 2019). Heckmann et al. showed that re-balling and re-soldering are possible in a low-temperature environment and can improve the reliability of NAND Flash obtained through a Chip-off (Heckmann et al., 2016).

3. Proposed certificate injection-based encrypted traffic forensic model

In general, to analyze TLS-based encrypted traffic using MitM, the HTTP proxy tools are used. Since the HTTP proxy tool operates based on the certificate, decryption of encrypted traffic has the condition that a certificate must be installed on the target device. Therefore, to analyze the encrypted traffic between AI Speakers and the cloud, the HTTP proxy tool's certificate must be installed on the AI Speakers. After installing the certificate, HTTP proxy tools can be used to analyze the encrypted traffic between AI Speaker and cloud. In this study, we use the commonly used HTTP proxy tools Charles (Charles Web Debugging Pro) and Fiddler (Fiddler - Free Web Debugg) to perform MitM on encrypted traffic.

The general method of installing the certificate and setting up a proxy on Android smartphone is well known and can be easily applied through the Android UI. However, AI Speakers do not provide Android UI to users. Owing to the nature of such AI Speakers, it is difficult to install a certificate and proxy configuration on the AI Speakers. Therefore, hardware interfaces, backdoors and vulnerabilities, and a Chip-off should be used to inject the certificate and set up a proxy in the AI Speaker. In other words, to analyze encrypted traffic between the AI Speaker and cloud, various aspects of the target AI Speaker must be analyzed.

The methods of injecting the certificate into AI Speakers proposed in this paper are not always applicable to all AI Speakers. The environment of the AI Speakers varies for different manufacturers, and thus we must always analyze AI Speakers from different perspectives. Therefore, we introduce five methods that can be considered when injecting the certificate into an AI Speaker. Each method involves a software and hardware level analysis, and some method involves Chip-off and BGA Rework. Based on these methods, we propose a forensic model for analyzing encrypted traffic between AI Speakers and the cloud.

This section first discusses how to inject the certificate and set up a proxy for AI Speakers, which must be conducted to analyze encrypted traffic using MitM. We then introduce a forensic model for injecting the certificate into AI Speakers and analyzing the encrypted traffic.

3.1. Certificate injection and proxy setting for MitM

AI Speakers use the Android OS, and thus the data storage structure is similar to that of typical Android smartphone. In other words, AI Speakers and Android smartphones have the same location where certificates and proxy configuration information are stored. With this in mind, we inject the certificate and proxy configuration on the AI Speaker. First, to inject the certificate, where certificates are stored in the Android OS must be known. In Android OS, the location where user-installed certificates are stored is in the userdata partition, although the detailed location varies depending on the version of Android used. In the versions below Android 5.0, they are stored in the path "/data/misc/keychain/cacerts-a dded/," and in versions of Android 5.0 and above it is stored in the path "/data/misc/user/0/cacerts-added/". After injecting the certificate of the HTTP proxy tool into the correct path for each version of Android, the permission and ownership are set to "rw-r–r-(644)" and "Everybody (9997):Everybody (9997)."

The file for the proxy configuration is stored in the path "/data/misc/wifi/" in the userdata partition. There are two files that need to be modified: one that store the Wi-Fi settings and the other that

store the proxy configuration. The file related to the Wi-Fi configuration is "wpa_supplicant.conf". This file stores the Wi-Fi SSID and PW, and the system uses it to automatically connect to Wi-Fi. By modifying this, AI Speakers can be automatically connected to the Wi-Fi SSID where the HTTP proxy tool is running. Fig. 1 shows the Wi-Fi SSID and PW of the modified "wpa_supplicant.conf".

The file related to the proxy configuration is "ipconfig.txt," which stores information such as the proxy host and port. The proxy can be set by modifying it. Fig. 2 shows "ipconfig.txt" modified the proxy host and port. For this file, it is safe to use the file obtained from the NAND Flash of the AI Speaker because the values of the various fields vary depending on the Android version. If the file cannot be acquired from NAND Flash, it is preferable to obtain it from an Android device of the same version as the AI Speaker.

3.2. Proposed forensic model

IoT devices such as healthcare device Fitbit and Mobile Robot, as well as AI Speakers, encrypt most of their traffic with TLS. However, studies conducted to date have had difficulty acquiring significant artifacts stored in AI Speakers' cloud. To solve this problem, we propose a forensic model for the AI Speakers.

Fig. 3 shows a forensic model for analyzing encrypted traffic between AI Speaker and cloud. Five methods for injecting a certificate into an AI Speaker and setting up a proxy, and network diagrams are shown. The AI Speaker is injected with a certificate using the proposed forensic model and connected to Wi-Fi supplied by the laptop's mobile hotspot function. The laptop has the HTTP proxy tools are installed. Because certificate is injected into the AI Speaker, the HTTP proxy tool can collect and analyze encrypted traffic between the AI Speaker and the cloud. The proposed five methods for injecting the certificate and setting up proxy are as follows:

- Case 1: Porting AI Speaker Image on Android Device
- Case 2: Porting AI Speaker Image using QEMU
- Case 3: Running Exploit using AI Speaker App Vulnerability
- Case 4: Rewriting NAND Flash using H/W Interface
- Case 5: Rework and Update NAND Flash

Case 1 is a method for porting data from NAND Flash of the AI Speaker to the Android device for a certificate injection and proxy configuration. If the porting is successful, the device created in this way can achieve the same function as the AI Speaker and at the same time use the functions available on the Android smartphone. This allows the certificate to be installed and a proxy to be set up using a system setup app that is unavailable on traditional AI Speakers, as well as to obtain root privileges to directly inject the certificate and modify the proxy configuration file.

Case 2 is a method for implementing the physical device conditions of the AI Speaker in software to facilitate debugging. Porting can be carried out as needed, and the Kernel and Ramdisk images can be modified to run on a virtual Raspberry Pi device, preserving the main partitions in the NAND Flash of the AI Speakers. With this configuration, the UART, JTAG, and other ports are virtually implemented, and thus logical debugging is possible without a physical implementation. This is more economical than the hardware porting method and can also be implemented by applying a snapshot function to restore the image to a desired point in time through a backup operation. After creating a virtual Raspberry Pi device that acts as an AI Speaker, we can use it to inject the certificate and set up proxy.

Case 3 is a method that takes advantage of backdoors and vulnerabilities that exist in AI Speakers. To do so, an image of the AI Speaker is analyzed to search for backdoors and vulnerabilities in the device or to use known vulnerabilities. This method is not applicable to all AI Speakers, but it is possible that the backdoor developed on some AI Speakers for firmware update testing and debugging is not removed. Subsequently, certificate injection and proxy configuration can be done on the AI Speaker using backdoors and vulnerabilities existing in the AI Speaker.

Case 4 is a method utilizing hardware interfaces such as UART, USB Boot, and JTAG found on the board of the AI Speakers. For example, if an UART port exists, the U-Boot command can be used through the U-Boot Shell. U-Boot's MMC command can be used to extract data from an AI Speaker in the form of a raw image and then modify the extracted image to inject the certificate and set up a proxy. The fastboot features can then be used to overwrite the data of AI Speaker with certificate injected image.

Case 5 is a method for separating NAND Flash from the board of the AI Speaker and then correcting the stored data or injecting new

```
ctrl_interface=/data/misc/wifi/sockets
disable_scan_offload=1
update_config=1
device_name=if_s300n
manufacturer=INFOMARK
model_name=NL-S110KR
model_number=NL-S110KR
serial_number=20013739
device_type=10-0050F204-5
config_methods=physical_display virtual_push_button
p2p_disabled=1
hs20=1
interworking=1
pmf=1
external_sim=1
tdls_external_control=1

network={
    ssid="ics_fi"
    bssid=64:e5:99:f6:93:46
    psk="11111111"
    key_mgmt=WPA-PSK
    id_str="%7B%22creatorUid%22%3A%2210019%22%2C%22configKey%22%3A%22%5C%22ics_fi%5C%22WPA_PSK%22%7D"
}
```

Fig. 1. Modified wpa_supplicant.conf

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000   00 00 00 02 00 0C 69 70 41 73 73 69 67 6E 6D 65  ......ipAssignme
00000010   6E 74 00 04 44 48 43 50 00 0D 70 72 6F 78 79 53  nt..DHCP..proxyS
00000020   65 74 74 69 6E 67 73 00 06 53 54 41 54 49 43 00  ettings..STATIC.
00000030   09 70 72 6F 78 79 48 6F 73 74 00 0D 31 39 32 2E  .proxyHost..192.
00000040   31 36 38 2E 31 33 37 2E 31 00 09 70 72 6F 78 79  168.137.1..proxy
00000050   50 6F 72 74 00 00 22 B8 00 0D 65 78 63 6C 75 73  Port.."¸..exclus
00000060   69 6F 6E 4C 69 73 74 00 00 00 02 69 64 B9 80 89  ionList....id¹€‰
00000070   05 00 03 65 6F 73                                 ...eos
```

Fig. 2. Modified ipconfig.txt.

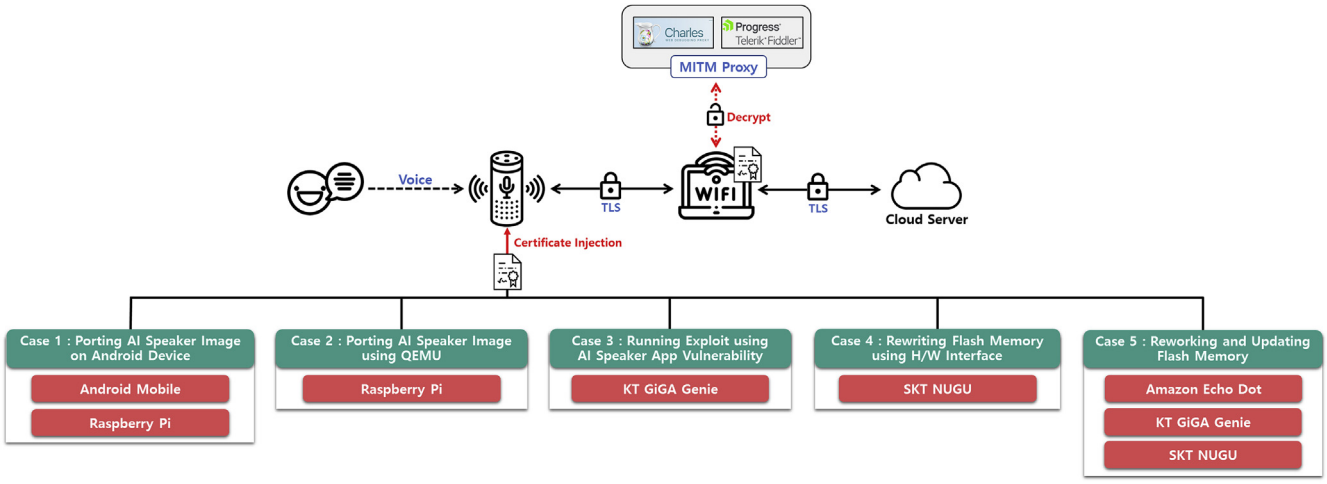

Fig. 3. Proposed encrypted traffic forensic model with five experimental methods.

data and reattaching it to the board. After separating NAND Flash from the board of the AI Speaker through a Chip-off, a certificate injection and proxy configuration are made in the NAND Flash acquired using a programmable reader. By reattaching the modified NAND Flash to the AI Speaker's board through a BGA Rework, we can analyze the TLS traffic between the AI Speaker and cloud.

4. Five forensic methods for encrypted traffic using certificate injection

This section deals with the experiments and results of applying the five methods of the forensic model to an AI Speaker to analyze the encrypted traffic between the AI Speaker and the cloud.

4.1. Porting AI Speaker Image on Android Device

This is a method for creating a new device that has the same function as the AI Speaker by porting data from NAND Flash obtained from AI Speaker to Android device. The goal is to inject the certificate and set up a proxy to run MitM using the device created. To do so, the image obtained from AI speaker via chip-off is ported to Android device. The experimental environment is presented in Table 1.

Table 1
AI speaker image porting specification.

Device	Manufacturer	OS version
Galaxy Note 3	SAMSUNG	Android 7.1.2 (AOSP)
Raspberry Pi 3B	Android Things	Android Things 1.0.3

Samsung's Galaxy Note 3 (Android 5.1.1) was used as the porting device, and Naver Clova was used as the Chip-off image. Details of the AI Speaker Chip-off image are shown in Table 2. In addition, because Android Things is being used as an AI Speaker development platform, an experiment was also conducted on a Raspberry Pi 3B.

The experiment consisted of porting AI Speaker images and installing the AI Speaker app. Porting of the AI Speaker image was conducted by modifying some data of the AI Speaker Chip-off image to suit the environment of the Galaxy Note 3. The AI Speaker app installation experiment was conducted by installing the main AI Speaker app on the Galaxy Note 3. First, to make the same version of the AI Speaker and Galaxy Note 3, we installed Android version 7.1.2 Android Open Source Project ROM on the Galaxy Note 3. Next, to avoid a driver collision, we selected and ported the data essential for the AI Speaker to operate. In addition, some data need to be modified to suit the AI Speaker environment. A typical example is code determining the model name of the speaker implemented in the AI Speaker app. Fig. 4 shows the code for determining the model of Naver Clova. For the AI Speaker app to operate properly on the Galaxy Note 3, “build.prop” in the system

Table 2
AI speaker image for porting experiment.

Category	Description
Manufacturer	Naver
Model	FRIENDS(NL-S110KR)
OS	Android 7.1.2
App version	Clova v1.9.6 (18.09′)
Size	7.27 GB

```
private static String getModelName(String str) {
    String str2 = null;
    if (str == null) {
        return null;
    }
    if (str.equals("NL-S100KR")) {
        str2 = "BROWN";
    } else if (str.equals("NL-S110KR")) {
        str2 = "SALLY";
    } else if (str.equals("NL-S120KR")) {
        str2 = "MINIONS";
    } else if (str.equals("NL-S130KR")) {
        str2 = "DORAEMON";
```

Fig. 4. Model identification code in AI speaker app.

partition needs to be modified.

In the image porting experiment using the Raspberry Pi 3B (Android Things) model, some partitions of the Android Things were divided into two, unlike with a general Android OS, which makes the porting difficult. The experimental results were similar to those of the Galaxy Note 3.

Table 3 summarizes the results of the Chip-off image porting experiment. The main reason for the failure is an AI Speaker system app installation error that occurs owing to the sharedUserId. The AI Speaker apps are a system app because it basically provides the ability to change the state of device. In this case, sharedUserId is used as shown in Fig. 5. AI Speaker apps from other manufacturers are also system apps, so the same problem is expected to occur.

4.2. Porting AI Speaker Image using QEMU

To solve the difficulty in porting hardware devices similar to case 1, we conducted a porting using QEMU. This method aims to make AI Speakers run on virtual Raspberry Pi. The QEMU emulator is used to run the ARM architecture on a regular PC. QEMU supports x86 and x86_64 architectures and ARM's 32- and 64-bit architectures. QEMU also enables an architectural emulation on many other virtual hardware platforms.

AI Speakers do not have a screen, thus we cannot directly check for errors during the booting process. However, as an advantage of using the QEMU emulator for porting, we can check dmesg and logcat immediately after booting. Table 4 shows the experimental environment using the QEMU emulator. And the AI Speaker Chip-off image information used is similar to the information in Case 1 presented in Table 2. The experiment was conducted by installing a QEMU emulator on macOS 10.13.6. To use the same ARM 32-bit architecture as the AI Speaker image, the Raspberry Pi 3 was used as the virtual hardware.

The AI Speaker's firmware was obtained through a Chip-off for porting the firmware. Then, using a Kernel extraction and decompression, the Ramdisk image was extracted and processed into the form required for the QEMU boot. Fig. 6 shows the process of extracting the Kernel and Ramdisk from the Naver Clova boot partition using unpackbootimg. Thereafter, we tried to emulate the QEMU using the acquired Kernel and Ramdisk, but there was a problem in that the normal boot did not operate properly. This problem occurs because it loads Adreno, Qualcomm's Framebuffer driver built into the Naver Clova Kernel. However, this is ported to the Raspberry Pi 3, the Mali GPU's Framebuffer driver must be loaded at booting time.

If a system is booted with QEMU using kernel and Ramdisk, the kernel source and device driver (kernel object library) will not match completely. Therefore, a full porting of all functions is impossible, but major systems such as microphones, speakers, and Ethernet are expected to operate. However, during this study, an open-source kernel of Naver Clova or APQ8009 was needed. However, it had not yet been released and no further research could be carried out. In the future, if we use the kernel source of APQ8009, the porting is expected to be successful. However, there are many aspects to consider, including the possibility that services and other functions may not function normally even after a successful boot.

Table 3
AI speaker image porting experiment results.

Device	Result	Cause
Galaxy Note 3	Failed	App Signature Mismatch
Raspberry Pi 3B	Failed	(1) Partition structure differences (2) App Signature Mismatch

4.3. Running exploit using AI Speaker App Vulnerability

There are numerous difficulties in installing a certificate and setting up a proxy on AI Speakers in the usual way. This method takes advantage of the backdoors and vulnerabilities that exist in AI Speakers to done certificate injection and proxy configuration. To do so, static analysis can be performed to the AI Speaker through chip-off, or known vulnerabilities and backdoors can be used. The AI Speaker KT GiGA Genie and the HTTP proxy tool Fiddler were used for the experiment.

Fig. 7 shows the experimental environment using KT GiGA Genie, and Table 5 summarizes the detailed information of the devices used in the experiment. Because KT GiGA Genie supports an external display, it is connected to a TV using an HDMI port. We also used Windows 10's mobile hotspot feature to create a Wi-Fi network that AI Speakers can connect to. The HTTP proxy tool Telerik Fiddler was installed on the laptop and used to collect and analyze encrypted traffic between KT GiGA Genie and the cloud.

In the case of KT GiGA Genie, a study on a backdoor and vulnerability analysis was previously conducted (RSR). Thus, we experimented with known backdoors and vulnerabilities from this previous study. According to the research conducted, KT GiGA Genie has a backdoor for debugging was implemented, and a vulnerability was existed. KT GiGA Genie's backdoor is implemented in StartupService.apk, which runs when the device starts. Fig. 8 shows the source code of the backdoor. Furthermore, in the previous study, it was shown that a backdoor can run the Android system setting app by taking advantage of the fact that the backdoor installs and executes a specific APK stored on a USB. It was also shown that the root privilege can be gained by exploiting the vulnerability of the running process. This vulnerability opens a socket and executes all commands entering the socket with root privileges.

We applied the proxy configuration using the Wi-Fi menu of the Android system settings app through the backdoor of KT GiGA Genie, as shown in Fig. 9. Next, we added a function to install the certificate of the HTTP proxy tool to the APK developed in a previous study. This installs the certificate using Android's Certificate Installer, which requires setting a PIN or password. However, KT GiGA Genie was prohibited from doing so owing to the AI Speaker's nature. Therefore, the certificate of the HTTP proxy tool had to be injected directly into the certificate storage path of the userdata partition. As mentioned earlier, the path for storing certificates in an Android OS differs for each version, and thus the Android version of KT GiGA Genie was determined by analyzing the “build.prop” of the system partition. Next, the certificate was

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:sharedUserId="android.uid.system"
    <uses-sdk android:minSdkVersion="24" android:targetSdkVersion="26"/>
    <permission android:name="ai.clova.cdk.permission.AUDIO_API" android:protectionLevel="normal"/>
    <permission android:name="ai.clova.cdk.permission.MEDIA_API" android:protectionLevel="normal"/>
    <permission android:name="ai.clova.cdk.permission.DEVICE_API" android:protectionLevel="normal"/>
    <permission android:name="ai.clova.cdk.permission.SOUNDTRIGGER_API" android:protectionLevel="normal"/>
    <permission android:name="ai.clova.cdk.permission.SPEECH_API" android:protectionLevel="normal"/>
    <permission android:name="ai.clova.cdk.permission.TASK_API" android:protectionLevel="normal"/>
    <permission android:name="ai.clova.cdk.permission.FOTA_API" android:protectionLevel="signatureOrSystem"/>
    <permission android:name="ai.clova.cdk.permission.COTA_API" android:protectionLevel="signatureOrSystem"/>
```

Fig. 5. SharedUserId implemented in AndroidManifest.xml.

Table 4
AI speaker QEMU emulation specification.

Device	Version
Host	10.13.6 High Sierra (17G8030)
emulator	QEMU 4.1.50
Raspberry Pi	Raspberry Pi 3 Model B
AI Speaker	Android 7.1.2

directly injected using the vulnerability analyzed in the previous study. As Fig. 10 indicates, the certificate was injected correctly. After certificate injection and proxy setup, the HTTP proxy tool was used to analyze the encrypted traffic between GiGA Genie and the cloud.

4.4. Rewriting flash memory using H/W interface

Most IoT devices implement some debugging ports such as UARTs in Pogopin format and place them at the bottom or invisible of the product. In particular, IoT devices in a headless form, in which the display is removed, may use this to confirm normal booting during the manufacturing process. AI Speakers, likewise, have an interface for accessing internal systems. This interface is used to inject certificate and set up proxy. The UART port on a SKT NUGU Candle was used for the experiment. The experiment consists of connecting the UART port and host through a USB device to the TTY and extracting, modifying, and rewriting the data of NAND Flash through a U-Boot command. The detailed experiment environment is presented in Table 6. For the USB to TTY device, FT232 FTDI and CP2102 were used. We used PuTTY in Windows because a terminal application is required to read and write data from the serial device.

Although a contact connection can be achieved using Pogopin, this study did not use the Pogopin method, which can be disconnected when extracting data for a long time at a high speed. We connected the contacts directly through soldering, as shown in Fig. 11. The contacts that need to be connected are UART_TX, UART_RX, and GND. Labeled PCBs are easy to find, but if they are not labeled, a boundary scan is required to find the TX/RX port. After connecting the contacts, the terminal applications PuTTY and USB are connected to the TTY with the device to see the boot message. When booting during a normal boot sequence, stdio, stdout, and stderr are redirected to/dev/ram using Android bootcmd, and thus the serial output to UART does not occur after the bootm command of U-Boot. Therefore, before the bootm command is executed, a specific key is typed, as shown in Fig. 12, to enter U-Boot Shell.

Then, to access NAND Flash from the U-Boot Shell and extract the data, the MMC command is used to format the specific block data of NAND Flash in hexadecimal format. Because SKT NUGU Candle is arranged at 512 bytes per block, they are read by 512 bytes using an MMC dump. To this end, as shown in Fig. 13, a tool was developed to automate the MMC dump command at the end of the NAND Flash.

After NAND Flash extraction, certificate injection and proxy configuration are conducted. The procedure used for modifying data in the userdata partition for a certificate injection and proxy configuration is described in 3.1. When the data are fully modified, they are rewritten into the userdata partition of the SKT NUGU Candle. To do so, the fastboot flash command is applied, as shown in Fig. 14. In the case of SKT NUGU Candle, the size of an image that can be uploaded at a single time is less than 1 GB. Therefore, to upload userdata partitions and system partitions of larger than 1 GB, the sparse image option provided by fastboot is used. The upper part of the figure shows the fastboot command entered in the host OS, and the lower part is the log output when the fastboot flash command is used.

```
sokdak@Sokdakino-ICSMac /Volumes/DataSpace/Works/2019 AI Forensics/AI Speaker Chip-off Dump/KL-S110KR_SD_2019011
0_FULL/b unpackbootimg -i ../KL-S110KR_SD_20190110_boot.mdf 2 ↵ 505 02:27:39
Android magic found at: 0
BOARD_KERNEL_CMDLINE console=ttyHSL0,115200,n8 androidboot.console=ttyHSL0 androidboot.hardware=qcom msm_rtb.filter
=0x237 ehci-hcd.park=3 androidboot.bootdevice=7824900.sdhci lpm_levels.sleep_disabled=1 earlyprintk buildvariant=us
er
BOARD_KERNEL_BASE 00008000
BOARD_RAMDISK_OFFSET 01000000
BOARD_SECOND_OFFSET 00f00000
BOARD_TAGS_OFFSET 00000100
BOARD_PAGE_SIZE 2048
BOARD_SECOND_SIZE 0
BOARD_DT_SIZE 0
```

Fig. 6. Kernel Extraction using unpackbootimg

Fig. 7. Kt GiGA genie experiment.

Table 5
AI speaker vulnerability experiment specification.

Device	Manufacturer	Version
GiGA Genie	KT	Android 5.1.1
Laptop (Gram)	LG	Windows 10 1903
Fiddler	Telerik	5.0.20192
TV (Display)	Samsung	

Next, fastboot mode is exited to check whether the AI Speaker is working properly, and the system is rebooted by entering the reset command in the U-Boot Shell. If the AI Speaker is booting normally, the HTTP proxy tool can collect and analyze the encrypted traffic. However, several considerations, such as SSL pinning implemented directly in Android and other applications, can make it difficult to analyze encrypted traffic. At this time, a modification of the system

```
private void launchExternalFactory(File file) {
    try {
        getPackageManager().installPackage(Uri.fromFile(file), new Stub() {
            public void packageInstalled(String packageName, int returnCode) throws RemoteException {
                if (returnCode == C0002StartupService.CONDITION_A_FACTORY_MODE) {
                    C0002StartupService.this.StartActivity(new ComponentName(
                        C0002StartupService.PACKAGE_AFACTORYMODE,
                        "net.quber.afactorymode.AFactoryModeMainActivity"));
                }
            }
        }, CONDITION_DEFAULT_APP_INSTALL, null);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Fig. 8. Backdoor implemented in StartupService.apk (RSR).

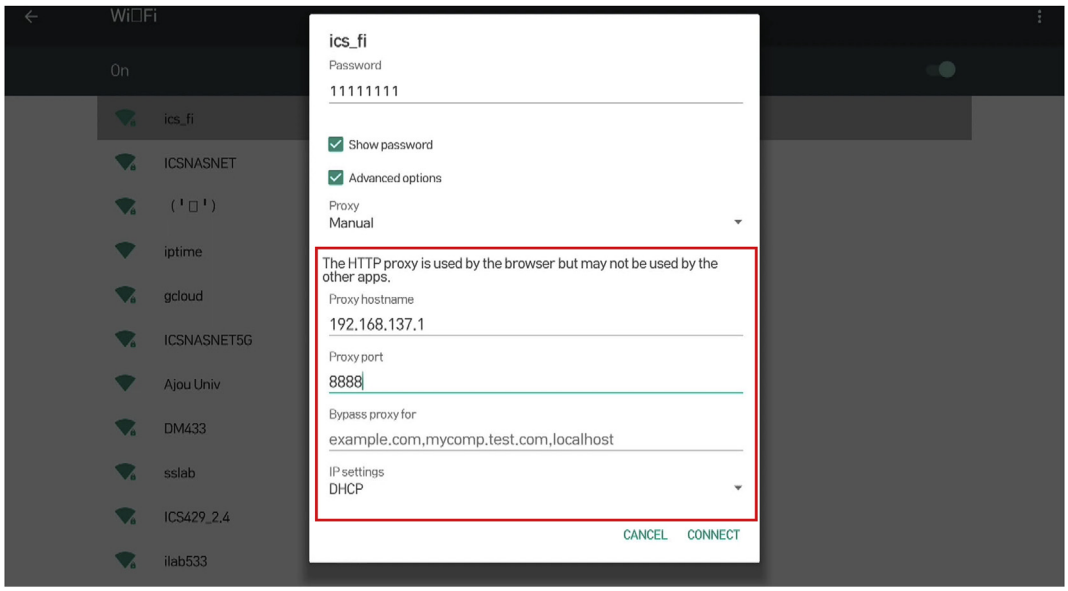

Fig. 9. Proxy Configuration using the System Settings App.

partition is required, and a method for bypassing a device using firmware tampering protection technology is necessary.

4.5. Reworking and updating flash memory

This is the last option that can be used when a certificate insertion and proxy configuration through other techniques are not possible. Chip-off is applied by default to directly modify the data stored in the NAND Flash obtained from the AI Speaker's board. The experimental environment is presented in Table 7. Since Chip-off is performed, various types of equipment are required, as shown in Fig. 15. In addition, a technical understanding of a Chip-off and BGA Rework is required.

The experiment consists of NAND Flash acquired from the AI Speaker, modifying the data, and then conducting a BGA Rework. First, Chip-off is performed to acquire the NAND Flash. Next, the detached NAND Flash is mounted on a PC using a programmable NAND Flash reader. Then the certificate and proxy configuration are injected. The methods used for a certificate injection and proxy configuration are described in 3.1. After completing a certificate injection and proxy configuration, a re-balling is performed to apply a BGA Rework. In this experiment, a 0.3 mm ball was used for re-balling and heated to a temperature of 275 °C. Next, a BGA Rework is performed according to the guidelines of NAND Flash. If the BGA rework is done successful, the HTTP proxy tool can be used to check the packets that the AI Speaker is communicating with the cloud server.

We applied this method to Amazon Echo Dot, SKT NUGU, and KT GiGA Genie, and verified that each device works properly. If the certificate injection and proxy configuration are properly applied, a connection confirmation message is displayed on the HTTP proxy tool, and the device name on the list of connected devices can be found in the mobile hotspot menu, as shown in Fig. 16.

Table 6
AI speaker UART experiment specification.

Item	Device	Version
OS (Host)	LG Gram	Windows 10 Pro
OS (Guest)	VMware	Ubuntu 18.04
AI Speaker	SKT NUGU Candle	Android 7.0
Serial Device	FT232 (FTDI), CP2102	

5. Experiment results and evaluation tool

This section deals with the main artifacts obtained through experiments applying the proposed forensic model to AI Speakers. We then introduce a digital forensics tool developed to collect artifacts stored in KT GiGA Genie cloud.

5.1. Significant artifacts from TLS traffic analysis

In the experiments performed with proposed five methods, we injected the certificate into Amazon Echo Dot, SKT NUGU Candle, SKT NUGU, and KT GiGA Genie and tried to analyze the encrypted traffic. For Amazon Echo Dot, SKT NUGU Candle, SKT NUGU, the

```
root@Hi3798CV200:/data/misc/user/0/cacerts-added # ll
-rwxrwx---  system  system      950 2019-11-01 14:58 807e3b02.0
-rwxrwx---  system  system      950 2019-11-01 14:58 807e3b03.0
-rwxrwx---  root    root        950 2019-11-01 14:58 807e3b04.0
-rwxrwx---  root    root        950 2019-11-01 14:58 807e3b05.0
-rwxrwx---  system  system      950 2019-11-01 14:58 807e3b06.0
root@Hi3798CV200:/data/misc/user/0/cacerts-added #
```

Fig. 10. Certificate injected into the certificates install path.

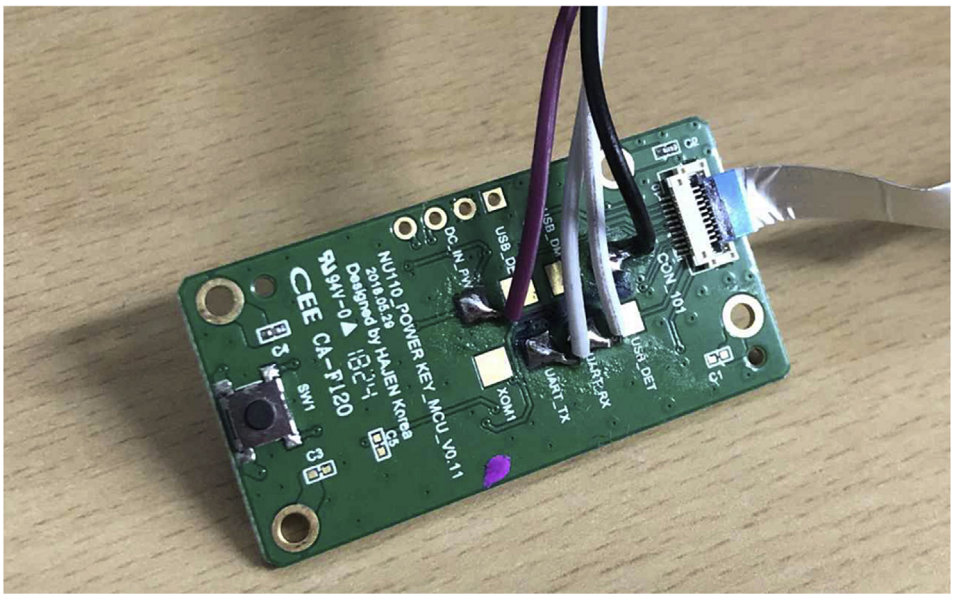

Fig. 11. SKT NUGU Candle connected with soldering.

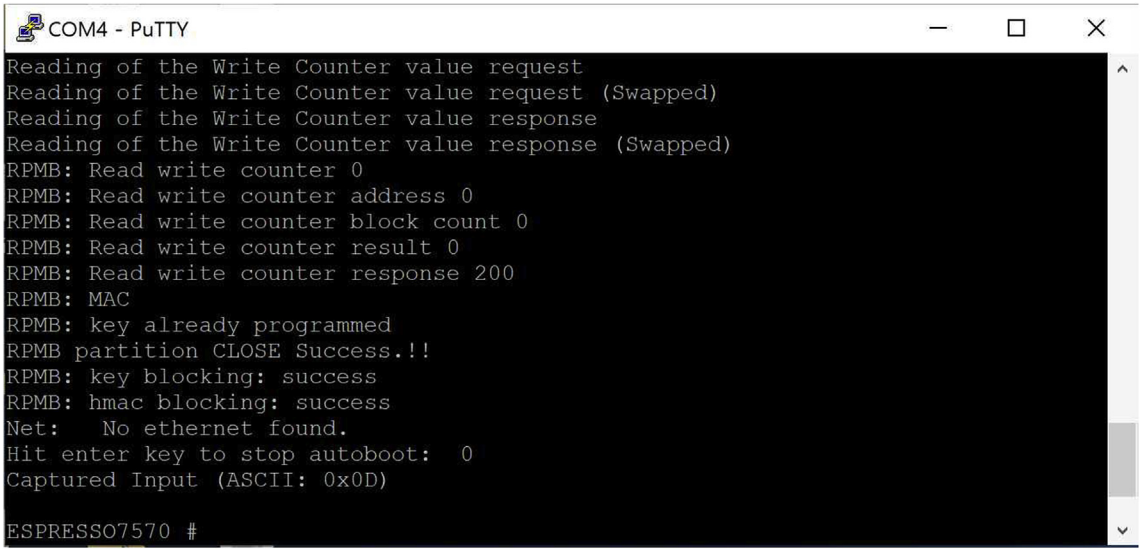

Fig. 12. Boot Sequence Escape in U-Boot (Shell dropping).

analysis was nearly impossible due to SSL pinning, especially worse on Amazon Echo Dot and SKT NUGU Candle. For the Amazon Echo Dot, we also used the Alexa Pi platform provided for the Raspberry Pi for further experiments, but the results were the same. Experiments with Amazon Echo Dot and Alexa Pi found only the addresses of cloud servers that appear to be receiving voice commands. Whether or not to store voice commands is unclear due to SSL pinning. In case of SKT NUGU, all traffic could not be analyzed due to SSL pinning, but artifacts such as authentication token, device information, location information, and usage time were acquired. But for KT GiGA Genie, we were able to analyze most of the encrypted traffic. As a result of analyzing the encrypted traffic of KT GiGA Genie, we found traffic for a voice query, memo registration, alarm registration, schedule registration, device information, and user information. In the case of voice command information, we confirmed the sending of a request to store the history in the cloud. In addition, it was estimated that the method for storing and processing voice command records was developed as a result of the app source code analysis. Also interesting is the session tokens and credentials that KT GiGA Genie uses to communicate with the cloud, stored in plain text in the KT GiGA Genie app directory. As shown in Fig. 17, the session token used for actual communication is stored in plain text. As shown in Fig. 18, the credentials of the device information used for most types of communication are also stored in plain text.

The cloud artifacts derived by applying the proposed five methods to various AI Speakers are shown in Appendix A. For Amazon Echo Dot, SKT NUGU Candle, and Naver Clova, no artifacts were obtained because encrypted traffic could not be analyzed due to SSL pinning or H/W porting issues. Traffic analysis allowed us to identify only the address of the cloud server. For SKT NUGU and KT GiGA Genie, we were able to derive 10 different artifacts stored in the cloud.

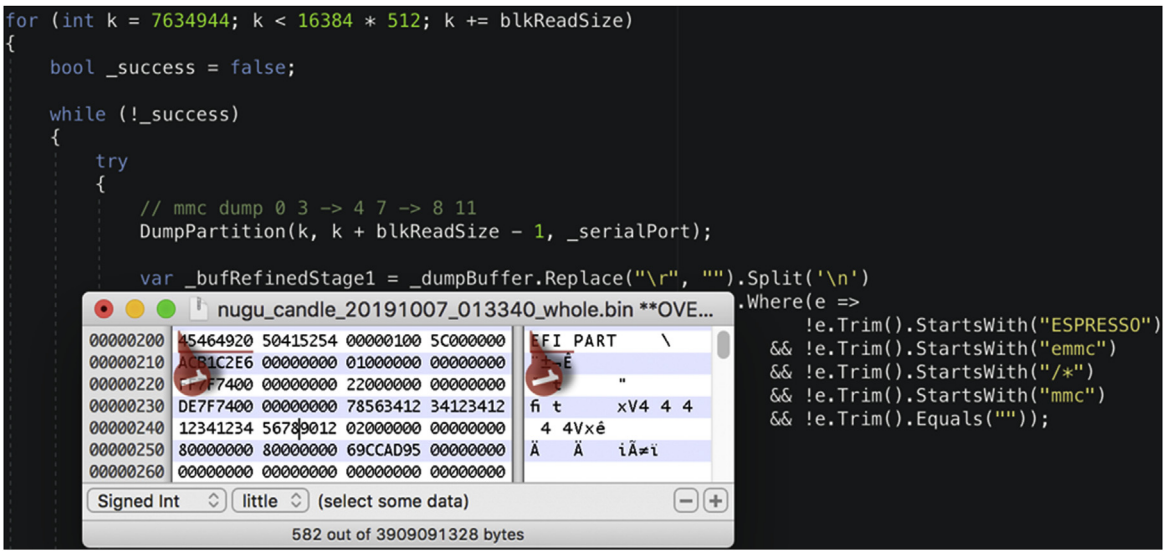

Fig. 13. Firmware automatic extraction tool.

5.2. Evaluation tool

We applied the proposed forensic model to various devices. As a result of case study, the most significant result was obtained in KT GiGA Genie's case. Therefore, we developed a verification tool that collects artifacts stored in KT GiGA Genie's cloud.

This section introduces a cloud artifact collection tool developed based on the credentials obtained from analyzing encrypted traffic between KT GiGA Genie and the cloud. The developed tool takes advantage of the ability to issue a session token directly using credentials stored in the device as plain text. The session token is obtained by using the credentials stored in the "launcherCommon.db" file that can be obtained from the AI Speaker and requests information from the cloud server in the same format as the AI Speaker. We can use this tool to acquire artifacts stored in the cloud. When the tool first starts, it uses the credentials acquired from KT GiGA Genie to get session token from the cloud. The tool then sends a request to the cloud in the same format that the AI Speaker uses when it communicates with the cloud. At this point, the session token just issued is used. Fig. 19 shows memo information stored in the cloud obtained through the process described above.

Table 7
NAND flash BGA rework experiment specification.

Device	Manufacturer	Remarks
Echo Dot	Amazon	AI Speaker
NUGU	SKT	AI Speaker
GiGA Genie	KT	AI Speaker
FX-951	HAKKO	Soldering iron
FR-810B	HAKKO	Heat Gun
Gordak 853	Gordak	Preheater

6. Discussion

The methods of Case 1 and Case 2 are methods to inject the certificate after porting the AI Speaker image to other devices. As already mentioned, the experiments were not good results. As a result of this experiment, both Case 1 and Case 2 had difficulties. The reasons for the failure are as follows: Case 1 failed owing to a signature conflict between the AI Speaker app and the system app installed on the ported Android device. We tried to solve this by decompiling and tampering with the app, but due to the high level of source code obfuscation, we couldn't do any further research. Case 2 failed owing to a problem at the kernel level. This was due to the difference between the Raspberry Pi and Naver Clova chipsets.

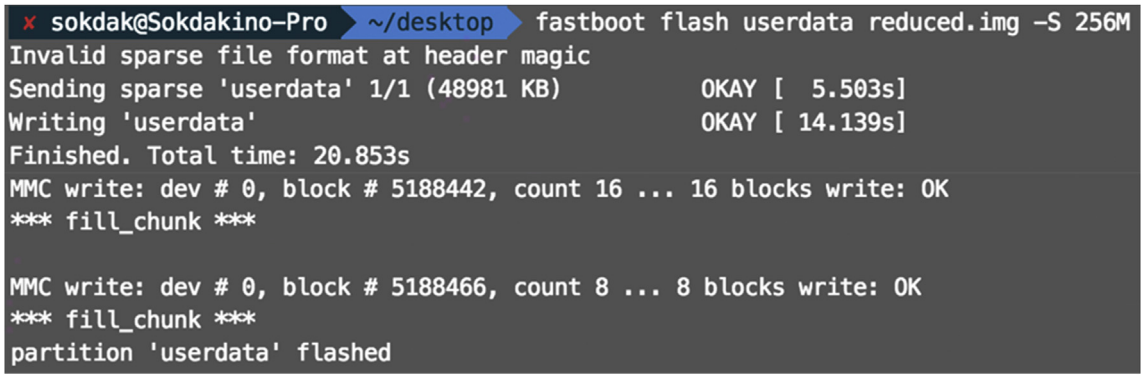

Fig. 14. Userdata Partition Flash using Fastboot

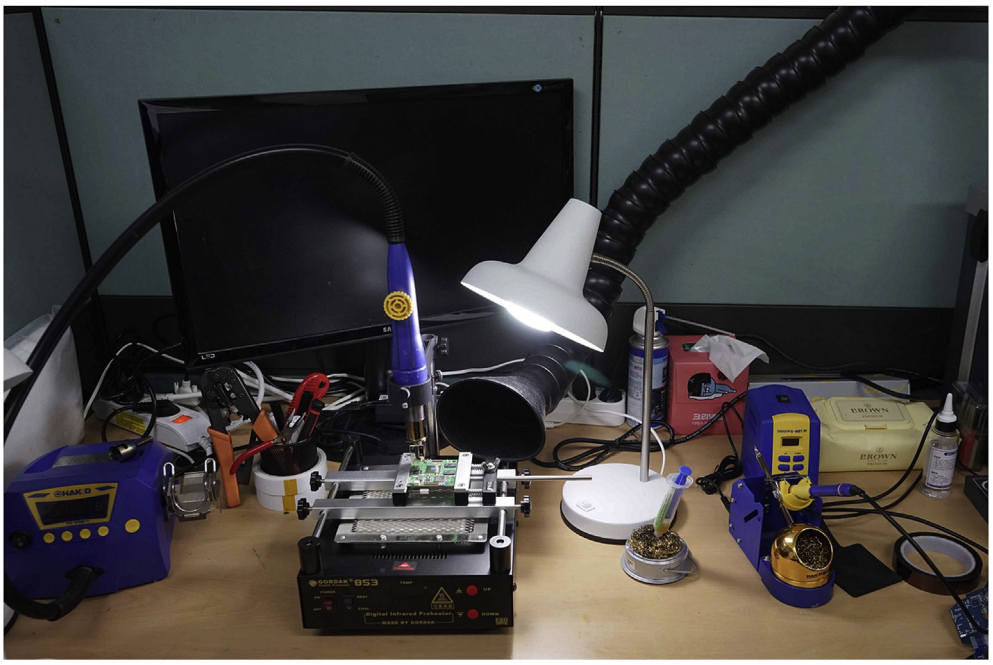

Fig. 15. Equipment for chip-off and BGA rework.

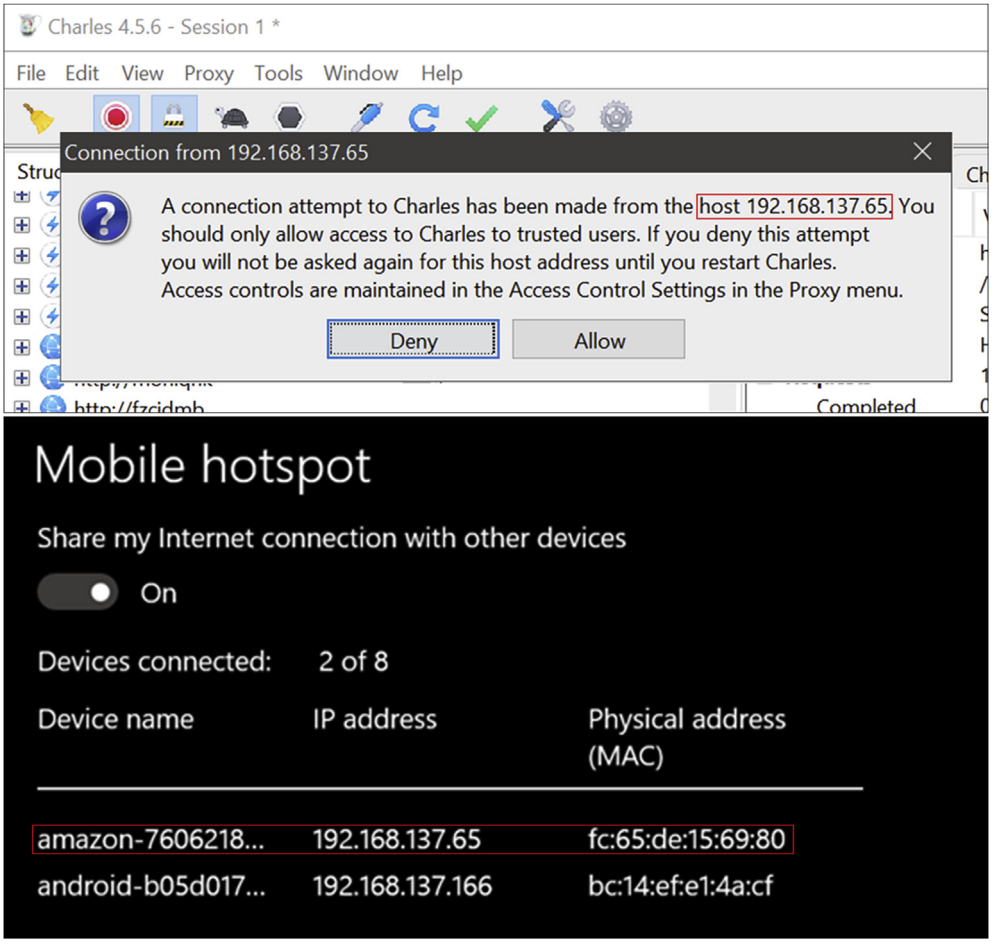

Fig. 16. Charles Proxy Connection and Connected list.

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="JS0001">1.5.0</string>
    <string name="cacheDevVersion">06.02.01</string>
    <string name="dongCodeIp">192.168.137.120</string>
    <long name="dataSyncTime" value="1569226777175" />
    <int name="prefVersion" value="1" />
    <string name="JS0002">1.0.2</string>
    <string name="sessionToken">beKJVpFVeyjvfYBJxRK9hZ87h7LVt6HIKYUvJhiY7fNHqidXFpGCyXB7kPjyOS7mTIeF/
    qhmRF4gXXljmOsTH6QNTbJpJ+CxMCfwyOPmarornU71RR0zQhW/XBcQ9bOH</string>
</map>
```

Fig. 17. Session token stored in PREF_SETTINGS.xml.

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: common

	devServiceId	devMacId	userKey	devNickname	devName	devAuthKey
	Filter	Filter	Filter	Filter	Filter	Filter
1	D100107111513y4DbJAm	80:8C:97:90:7B:25	A2434755804	GiGAGenie_7b25	GiGAGenie_7b25	nQS6Srdq67

Fig. 18. Credentials stored in the launcherCommon.db.

The analyzed TLS Traffic is the result of the AI speaker's previous command using Replay Attack.

no : 14871

ownerKey	A2434755804
memoText	The crime place is AjouUniversity
memoTitle	None
memoTime	20190712135353

Fig. 19. Memo information collected by developed tool.

To solve this, we needed the open source kernel of APQ8009. However, since it was not publicly available, no further research could be carried out. In the future, if we obtain this open source kernel or use a development board that uses APQ8009, porting will be possible.

The methods of Case 4 and Case 5 are methods for injecting the certificate by modifying the data stored in the NAND Flash of the AI Speaker. Both methods have the same idea, except that Case 4 uses the H/W interface UART and Case 5 uses Chip-off and BGA Rework. As already mentioned, the experimental results were not good except for KT GiGA Genie. The experimental results for each method are as follows. In Case 4, we successfully injected the certificate into SKT NUGU Candle but failed to analyze encrypted traffic due to SSL pinning. In Case 5, we successfully injected the certificate into Amazon Echo Dot, SKT NUGU, and KT GiGA Genie. For KT GiGA Genie, we were able to analyze most of the encrypted traffic, the same as in Case 3. However, for Amazon Echo Dot and SKT NUGU, most of the encrypted traffic could not be analyzed due to SSL pinning. SKT NUGU could see some traffic, but Amazon Echo Dot was especially bad. As our experiment shows, even if we inject the certificate into the AI Speaker, it is difficult to analyze the encrypted traffic when SSL pinning is implemented in the AI Speaker app. In this regard, there have been some studies on bypassing SSL pinning for Android apps. AI Speakers also use Android OS, but they have different environments from Android smartphones. Because of this, existing studies that bypass SSL pinning cannot be applied to AI Speaker as they are. So further research is needed to solve it.

Another limitation of the methods we propose is that they can potentially be short-lived. Basically, the IT field continues to contain new technologies and becomes more complex. AI Speakers are also updated frequently, and security concerns will be quickly improved. Because the proposed methods rely on R/W and exploitation of flash memory, it may not be possible to apply it due to certain security updates. However, there will continue to be a way to approach vulnerability again within complexity. And it's also

an important point that AI Speakers are updated much more frequently than other IoT devices. This means that the methods we propose can still be applied to other IoT devices.

7. Conclusion

Analyzing the encrypted traffic between AI Speaker and cloud, as well as artifacts stored on the cloud, is one of the most important research topics from the perspective of cloud-based IoT forensics, including AI Speakers. In this study, the forensic models using an MitM-based AI Speaker certificate injection were proposed and tested. The forensic model consists of five methods for injecting the certificate. These methods utilized porting, the backdoor and vulnerability of an app, the hardware interface, and a BGA Rework. To inject the certificate, three of the methods were applied to AI Speakers including an Amazon Echo Dot, and some AI Speakers of Amazon, Naver, SKT and KT were able to analyze the encrypted traffic. In fact, a forensic investigator can apply our forensic model with five methods to AI Speakers to obtain artifacts stored on the cloud. For this purpose, an acquisition and static analysis of the AI Speaker's NAND Flash through a Chip-off must be conducted. If the proposed method cannot be applied, a BGA Rework will be the last option.

Future work includes porting research using an open source kernel and bypassing SSL pinning, the biggest problem in analyzing encrypted traffic from AI Speakers.

Acknowledgements

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2018-0-01000, Development of Digital Forensic Integration Platform).

This research was supported by Energy Cloud R&D Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT (2019M3F2A1073386).

Appendix A. Cloud Artifacts of KT GiGA Genie

Device	Methods	Cloud Artifact
Amazon Echo Dot	Case 5	Unknown*
Naver Clova	Case 1, 2	Unknown**
SKT NUGU Candle	Case 4	Unknown*
SKT NUGU	Case 5	Authentication token Device info Location info Usage Timestamp
KT GiGA Genie	Case 3, 5	Session token User info Device info Memo info Alarm info Schedule info Location info Third-party app info Voice command Usage Timestamp

*Should be considered SSL pinning.
**Difficulties from H/W porting issues.

References

Burkholder, Peter, 2002. SSL Man-In-The-Middle Attacks. The SANS Institute.
Callegati, F., et al., 2009. Man-in-the-middle attack to the HTTPS protocol. In: IEEE Secur. Priv. https://doi.org/10.1109/MSP.2009.12.
Charles Web Debugging Proxy. https://www.charlesproxy.com/. Accessed at 19/01/20.
Chung, H., et al., 2017. Digital forensic approaches for Amazon Alexa ecosystem. In: DFRWS 2017 USA - Proceedings of the 17th Annual DFRWS USA. https://doi.org/10.1016/j.diin.2017.06.010.
Fiddler - Free Web Debugging Proxy - Telerik. https://www.telerik.com/fiddler. Accessed at 19/01/20.
Fukami, A., Nishimura, K., 2019. Forensic analysis of water damaged mobile devices. Digit. Invest. https://doi.org/10.1016/j.diin.2019.04.009.
Fukami, A., et al., 2017. Improving the reliability of chip-off forensic analysis of NAND flash memory devices. In: DFRWS 2017 EU - Proceedings of the 4th Annual DFRWS Europe. https://doi.org/10.1016/j.diin.2017.01.011.
Heckmann, T., et al., 2016. Low-temperature low-cost 58 Bismuth - 42 Tin alloy forensic chip re-balling and re-soldering. Digit. Invest. https://doi.org/10.1016/j.diin.2016.10.003.
Jo, W., et al., 2019. Digital forensic practices and methodologies for AI speaker ecosystems. Digit. Invest. https://doi.org/10.1016/j.diin.2019.04.013.
Li, S., et al., 2019. IoT forensics: Amazon Echo as a use case. IEEE Internet Things J. https://doi.org/10.1109/JIOT.2019.2906946.
Lin, F.Y., et al., 2015. A cloud-based forensics tracking scheme for online social network clients. Forensic Sci. Int. https://doi.org/10.1016/j.forsciint.2015.08.011.
Ramírez-López, F.J., et al., 2019. A framework to secure the development and auditing of SSL pinning in mobile applications: the case of android devices. Entropy. https://doi.org/10.3390/e21121136.
RSR. GiGA GENIE. https://blog.lvu.kr/%EA%B8%B0%EA%B0%80%EC%A7%80%EB%8B%88/. Accessed at 11/05/19.
Soghoian, C., Stamm, S., 2012. Certified lies: detecting and defeating government interception attacks against SSL (short paper). In: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). https://doi.org/10.1007/978-3-642-27576-0_20.
Voicebot, 2019. Smart Speaker Consumer Adoption Report.