

# State of the Art of Anti-Screen Capture Protection Techniques

Young Lee<sup>1</sup>, and SangGeun Hahn<sup>1\*</sup>

<sup>1</sup> Department of Mathematical Sciences, KAIST  
291 Daehak-ro Yuseong-gu Daejeon 34141 South Korea  
[e-mail: cryptoyoung@kaist.ac.kr, sghahn@kaist.ac.kr]

\*Corresponding author: SangGeun Hahn

*Received February 20, 2021; revised April 10, 2021; accepted May 9, 2021;  
published May 31, 2021*

---

## Abstract

The transition toward a contactless society has been rapidly progressing owing to the recent COVID-19 pandemic. As a result, the IT environment of organizations and enterprises is changing rapidly; in particular, data security is expanding to the private sector. To adapt to these changes, organizations and companies have started to securely transfer confidential data to residential PCs and personally owned devices of employees working from home or from other locations. Therefore, organizations and companies are introducing streaming data services, such as the virtual desktop infrastructure (VDI) or cloud services, to securely connect internal and external networks. These methods have the advantage of providing data without the need to download to a third terminal; however, while the data are being streamed, attacks such as screen shooting or capturing are performed. Therefore, there is an increasing interest in prevention techniques against screen capture threats that may occur in a contactless environment. In this study, we analyze possible screen capture methods in a PC and a mobile phone environment and present techniques that can protect the screens against specific attack methods. The detection and defense for screen capture of PC applications on Windows OS and Mac OS could be solved with a single agent using our proposed techniques. Screen capture of mobile devices can be prevented by applying our proposed techniques on Android and iOS.

---

**Keywords:** Agent, API Level 17, Capture prevention, Detours hooking, Fairplay DRM, Profile

## 1. Introduction

**B**ecause the outbreak of COVID-19, which first started in Wuhan City, Hubei Province of China in December 2019, has been unexpectedly prolonged, teleworking from home or from other locations excluding the office is being implemented worldwide. For example, a large-scale confirmed case recently occurred in a Korean call center, after which the employees continued to work from home to avoid service interruption. Therefore, the call center company purchased PCs with security solutions and delivered them to the residence of each of their employees. These types of cases generate new security vulnerabilities to businesses by exposing an organization's internal data to civil areas, and such security issues are expected to rise.

Considering the office security environment prior to COVID-19, when we went to work, we checked for access control with fingerprints [1] or cards with a security chip. After booting a PC with the password issued by the organization, we used documents that were packaged by digital rights management (DRM) [2]. If we attempted to copy documents by inserting a USB, permission was required. All printed or copied documents were reported to the central server and the names of the individual who printed them remained logged. All employee PCs were periodically scanned to check for malware, and important security solutions were automatically patched. Thus, secure and sophisticated hardware [3], data encryption software, or anti-malware solutions existed at the work locations to prevent comprehensive cyberattacks. Organizations now need to provide a secure working environment for telecommuters in the contactless era.

Therefore, organizations and companies are introducing streaming data services, such as the virtual desktop infrastructure (VDI) [4] or cloud services, to securely connect internal and external networks. These services provide data without having the need to download to third terminals; however, attacks such as screen shooting or capturing can occur while the data are being streamed. Therefore, there has been an increasing interest in screen capture prevention techniques.

Previously, a screen capture technique mainly performed on PCs involved an attacker infecting a target PC with a malicious code, taking over the system, extracting the screens in the target PC, and transmitting them to a remote location. Currently, because general malicious codes are detected by an anti-virus software, attackers use a specific tool to create and use malicious codes that bypass this software.

Screen capture by insiders who aim for monetary gains is increasing rather than by external attackers. Here, because screen capturing is performed by a normal program, the anti-virus software does not detect any malicious codes. It is common for an authorized user to install the screen capture tools on his/her PC to capture confidential data, which is then possibly leaked. Obtaining a capture tool from the internet or online is simple, especially for professional programmers who can easily program and create a capture tool for special purposes.

In addition, screen watermarking [5] is usually used against internal attackers, where the user (department name, title, name, etc.) and system (IP, Mac address, etc.) information are displayed. However, screen watermarking is a passive defense technique because it detects leaks after the occurrence of one.

Therefore, effective detection and defense techniques are required to prevent telecommuters or attackers from capturing and leaking key business data. We have conducted research regarding techniques that can prevent these captures using one agent for each operating system (OS) in a PC, rather than techniques that prevent screen capture for individual

capture tools or individual target applications. However, for Windows or Mac, various OS versions have already been released; thus, it was difficult to develop a technique to prevent all the capture tools with a single agent. In addition, to apply the proposed prevention technique in mobile devices, a different method was required because the OS structure varies. Fortunately, mobile devices provide screen capture prevention functionalities in their OS. The initial version of Android did not, although the recent version provides these security functionalities; the iOS also provides security functionalities.

We propose agent-type screen capture prevention techniques in a PC and techniques using the security functions provided by the OS in mobile devices. We also propose a complete screen capture prevention technique that includes PCs and smartphones.

In the following section, we analyze and propose the screen capture and protection techniques on a PC with Windows and Mac OS. In Section 3, we present screen capture and protection techniques for smartphones with the Android and iOS OS. In Section 4, we compare the anti-screen capture techniques presented in the previous sections, which is finally followed by the conclusion and future work in Section 5.

## 2. Screen Capture Protection Techniques on PCs

### 2.1 Anti-screen capture techniques on the Windows OS

Whether a commercial capture tool or a special purpose capture tool developed by an attacker is used, the three main following methods are used to capture a screen for the Windows OS:

- Capture using a general capture tool
- Capture using a remote capture tool
- Capture using the Clipboard

To block all the existing screen capture tools or the screen capture of all applications, finding the capture tools and registering them individually to prevent their use would not be feasible. Furthermore, blocking the entire path that can leak the screen in a UI (User Interface) of each application is not a solution.

In this section, we introduce anti-screen capture techniques using the agent developed in the Windows OS.

#### 2.1.1 Agent-based anti-screen capture technique for a full screen

**Table 1** presents how a general capture tool captures the screen of a target application.

**Table 1.** Screen Capture Method by a Capture Tool

STEP	Process
1	<p>Capture tool calls GDXXX(HDC hdcDest, HDC hdcSrc, DWORD dwRop) function.</p> <ul style="list-style-type: none"> <li>❑ HDC hdcDest // handle to destination, the handle value of the capture tool</li> <li>❑ HDC hdcSrc // handle to source, the handle value of the target application</li> <li>❑ DWORD dwRop // its property has SRCCOPY(copy property), SRCCOPY means copy from hdcSrc's screen to hdcDest's screen</li> </ul> <p>A capture tool reads the target application handle value. This is possible because every program has a unique handle and PID (process ID) value.</p>
2	<p>If HDC hdcSrc has the same value as the entire monitor screen size and the handle value of hdcSrc is copied to hdcDest by DWORD dwRop, then a capture tool copies the target application's screen to its memory.</p> <p>We finish the screen capture.</p>

Considering the effective methods for preventing these types of capture tools, we introduce an effective screen capture prevention technique.

We will install an agent on a Windows PC. It will monitor the entire screen capture process and if a screen capture attempt occurs, it blocks the screen capture action; the steps are presented in **Table 2**.

**Table 2.** Agent-based anti-screen capture process against full screen capture

STEP	Process
1	To capture a specific screen, run the Capture Tool (CT.exe).
2	The Capture Prevention Module (CPM.exe), which resided for protecting the target application screen on the PC, hooks when CT.exe is executed and inserts the capture prevention DLL in CT.exe.
3	The capture tool with the DLL inserted by hooking transitions to CT.exe from CT.exe.
4	After CT.exe captures a screen of a target application following Step 2 in <b>Table 1</b> , then it deletes the captured image in its memory or replaces the captured image by another image (for example, white, black screen, or 'This page is protected.'). Then, screen capture is prevented.

To better understand the technique shown in **Table 2**, it is necessary to comprehend the hooking technique. We will present the hooking technique introduced in step 3 of **Table 2** in more detail.

All executable files provided by Windows are composed of the PE structure shown in **Table 3**.

**Table 3.** PE structure of EXE file on Windows [6]

Dos Header
PE Header
.text Section(code)
.data Section
.idata Section(IAT)
.edata Section(EAT)
Debug Symbols

The following three hooking techniques are used for screen capture prevention:

### 1. Import Address Table Hooking (IAT hooking)

IAT hooking is a method that converts the IAT of the target application into the address of the function created by CPM.exe.

### 2. Export Address Table Hooking (EAT hooking)

EAT hooking changes the EAT of the DLL (Gdixx.dll) that contains the API into the address of the function created by CPM.exe. Therefore, when the target application DLL (MyDLL) calls DLL (Gdixx.dll), it introduces a new address of the function created by CPM.exe.

### 3. Inserting an unconditional jump (ICode, Detours) [7]

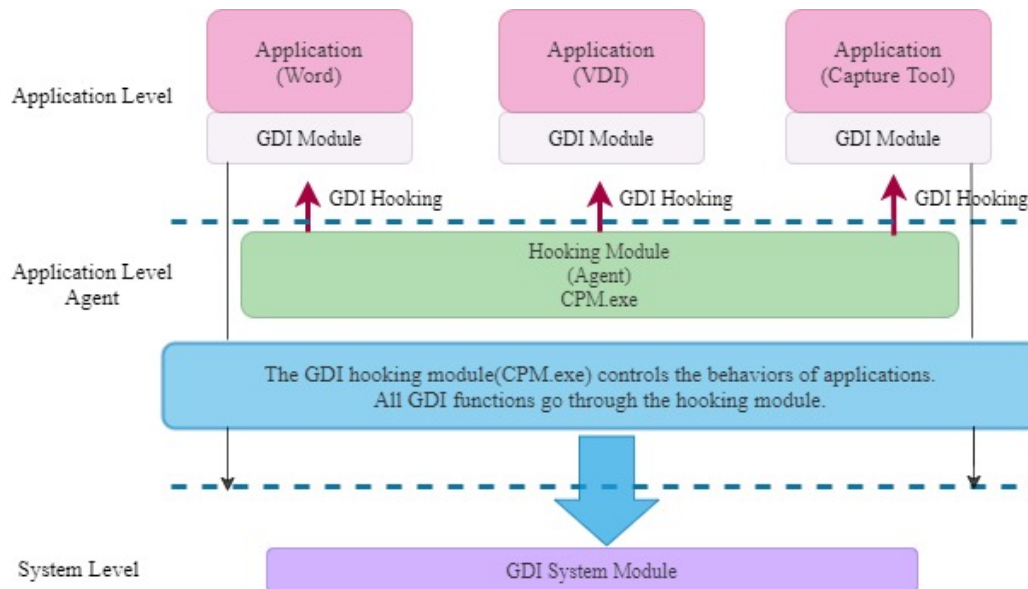
This technique inserts the jump code into the first part of the application's actual function code and then jumps into the function implemented by CPM.exe.

This technique provides stability and scalability to the system compared to the other two aforementioned technologies introduced. Therefore, we recommend this method to be mainly

used when developing an agent. This technology is provided by Microsoft and is called Detours.

**Fig. 1** presents the hooking process demonstrated in **Table 2** [8-12].

The CPM.exe file is an agent-type screen capture prevention module. It can be applied to all applications on the Windows OS. Its main role is to hook each application when launched. At this stage, CPM.exe cannot distinguish the capture tool among the applications. Therefore, CPM.exe injects its DLL into all running applications, assuming one of them may be a capture tool. We named the changed capture tool as CT'.exe (**Table 2**). The CT'.exe monitors the functions related to captures in the various system processes, unlike the original capture tool called CT.exe. In addition, when the field values of the handle are processed as indicated in **Table 1**, it is assumed that the screen is captured, and the capture is blocked.



**Fig. 1.** API hooking process for anti-screen capture.

Next, we will consider the techniques that block the capture methods that capture a partial screen rather than the entire screen.

### 2.1.2 Agent-based anti-screen capture technique for partial screen

The partial capture method is similar to the full-screen capture method, differing in that the capture tool stores the coordinate values of the area desired to be captured. After capturing the entire screen, as shown in **Table 1**, it is retrieved into the memory of the capture program and the desired area is cut according to the saved coordinate value to obtain a partial capture screen.

In this case, the capture prevention technique is the same as the full screen capture prevention technique presented in 2.1.1, with a difference that the partial screen obtained according to the coordinate value is changed with the other image, as shown in **Table 2** (STEP4), and the remaining of the screen does not change.

Therefore, the screen capture prevention process for a partial screen capture is the same as that shown in **Table 2**; however, only STEP4 is different, as shown in **Table 4**, as follows:

**Table 4.** Agent-based anti-screen capture process against partial screen capture

STEP	Process
1	To capture a specific screen, run the Capture Tool (CT.exe).
2	The Capture Prevention Module (CPM.exe), which resides for protecting the target application screen on the PC, hooks when CT.exe is executed and inserts the capture prevention DLL in CT.exe.
3	The capture tool with the DLL inserted by hooking transitions to CT'.exe from CT.exe.
4	Similar to STEP 4 of <a href="#">Table 2</a> , CT'.exe captures and saves the screen of the application in its memory. Then, CT'.exe cuts the saved screen by the coordinate value of the desired area. Finally, CT'.exe changes the cutting image to another image (for example, white, black screen, or 'This page is protected.').

### 2.1.3 Agent-based anti-screen capture technique against a remote attack

Sections 2.1.1. and 2.1.2 presented the techniques for capturing and blocking the entire or partial screen, respectively. Here, we present the blocking technique for capturing a screen by remotely connecting to a specific PC.

The following two methods are used for remote capturing: a capture method through **virtual network computing (VNC)** and a remote capture method through **Terminal Service** by Microsoft. The technical details regarding the capture method through the Terminal Service are not disclosed.

The capture method using the VNC is as follows. Remote access by the VNC indicates that the screen of the target PC is captured and transmitted in real-time. It also indicates that after capturing the entire screen on the client PC, where the target application is installed, the resulting screen is compressed using image compression technology and then sent to the VNC server. We assume that the Terminal Service follows a similar process.

To prevent remote screen capturing, it is necessary to first determine whether it is in a remote connection state. In general, to obtain that information, the **EnumDisplayDevices** function needs to be called, which obtains the **DISPLAY\_DEVICE** information from that function. The following functions:

**DISPLAY\_DEVICE\_ATTACHED\_TO\_DESKTOP** or  
**DISPLAY\_DEVICE\_MIRRORING\_DRIVER** [8]

are presented to determine whether there is a remote connection in a remote or virtual OS environment.

If it is determined that it is remotely connected by the VNC method, the CPM.exe module does hooking the VNC module installed on the target PC in the same manner as shown in [Fig. 1](#). This indicates that the VNC program, which has completed hooking, captures the screen and immediately replaces it with a different screen before transmission and then transfers it to the VNC server. Thus, on the VNC server side, either only a black or white screen is displayed, or 'This page is protected' is displayed. The detailed process is presented in [Table 5](#).

**Table 5.** Agent-based anti-screen capture process against remote screen capture

STEP	Process
1	The CPM.exe module is installed on the target PC to monitor all processes.
2	A module to communicate with the VNC server is installed on the target PC by an attacker.
3	CPM.exe has been monitoring whether a remote connection occurred by the following functions:

	DISPLAY_DEVICE_ATTACHED_TO_DESKTOP or DISPLAY_DEVICE_MIRRORING_DRIVER
4	CPM.exe hooks the VNC client module when the remote connection is confirmed.
5	The hooked VNC module transfers the captured screen by replacing it with another screen just before sending it to the VNC server. The screen replacement method is the same as shown in step 4 of <a href="#">Table 2</a> .

The technical details are not disclosed for Microsoft's Terminal Service, thus it is impossible to control hacking using the hooking technology shown in [Table 5](#).

Therefore, the following functions:

### **DISPLAY\_DEVICE\_ATTACHED\_TO\_DESKTOP or DISPLAY\_DEVICE\_MIRRORING\_DRIVER**

identify whether remote access by Microsoft's Terminal Service is established. If it is identified as the Microsoft Terminal Service, then CPM.exe minimizes the screen size of the target application. Therefore, the screen is hidden, thereby neutralizing the capture attempt.

#### **2.1.4 Agent-based anti-screen capture technique for the Clipboard**

To block an image captured by the Clipboard, the specific attributes of a content corresponding to the image on the Clipboard need to be determined and deleted; it is a relatively easy capture blocking method. From the contents on the Clipboard, only the following two attributes indicate pictures: CF\_BITMAP and CF\_DIB [\[13\]](#). Therefore, after verifying the existence of these attributes, screen captures can be blocked using the STEP4 method shown in [Table 2](#).

[Table 6](#) presents how to block screen capturing by the Clipboard.

**Table 6.** Example of a screen capture prevention method using the Clipboard

```

OpenClipboard(NULL);
glb = ::GetClipboardData(uFormat);
if(uFormat == CF_BITMAP)// If it is a picture
{
    // Clear Clipboard. Can be changed to another picture
    EmptyClipboard();
}
else if(uFormat == CF_DIB)
{
    // Clear Clipboard
    EmptyClipboard();
}

```

In Section 2, we presented four methods by which an attacker can obtain user data by hacking the PC screen on the Windows OS and demonstrated the techniques used to prevent these attacks. As an optimal solution, it would be encouraging to be able to protect data from all these attacks using one agent. In the following section, we will explain which defense techniques are available on the Mac OS against the same attack.



## 2.2 Anti-screen capture technique on Mac OS

The following four main methods are used to capture a screen on the Mac OS:

- Full screen or partial screen capture by using a capture tool
- Screen capture using a remote capture tool
- Screen capture using the Clipboard and Desktop.

### 2.2.1 Agent-based anti-screen capture techniques for a full and a partial screen

As previously indicated, we developed an agent using a hooking technique and prevented capture attempts in Windows. However, for Mac OS, hooking technology is not allowed by the OS. In addition, Mac OS saves all the captured images to the Clipboard, except for Desktop capturing, whether using a full or a partial screen capture tool. On the Windows OS, the capture tool could save the captured image to its memory to allow editing if necessary; however, this is not allowed in the Mac OS. Therefore, in Mac OS, the capture can be blocked by simply blocking the Clipboard. Therefore, in this section, we introduce techniques for blocking screen capture in the Clipboard. The Desktop method will be presented in Section 2.2.3.

Screen capture blocking by the Clipboard in the Mac OS is as follows. In the Windows OS, when capturing the screen with the Clipboard, it is possible to prevent screen capturing by determining the existence of two attributes, **CF\_BITMAP** and **CF\_DIB** in the copied contents, and deleting them. Unlike the Windows OS, certain number values are generated in Mac OS. After checking these values, screen capture can be blocked. **Table 7** demonstrates the technique used to block screen capturing by using the default function values provided in the Mac OS.

**Table 7.** Example of screen capture prevention method using Clipboard on Mac OS

<p><b>CGSSetSymbolicHotKeyEnable (key, nProtect);</b></p> <p>This function determines whether a certain key is blocked. If the key value is 0, it is allowed; if it is 1, it is blocked. Thus, apply the following to block or allow screen capturing:</p> <p><b>Command+ Shift+Control+3:</b> Save a full screen on the Clipboard. <b>CGSSetSymbolicHotKeyEnable (29, 1); Block screen capture</b> <b>CGSSetSymbolicHotKeyEnable (29, 0); Allow screen capture</b></p> <p><b>Command +Shift+ Control+4:</b> Save a partial screen on the Clipboard. <b>CGSSetSymbolicHotKeyEnable (31, 1); Block partial screen capture</b> <b>CGSSetSymbolicHotKeyEnable (31, 0); Allow partial screen capture</b></p>
--

Therefore, when the agent (CPM.exe) saves the list of applications to be protected and the application is started, screen capture can be prevented by using the function **CGSSetSymbolicHotKeyEnable (29 or 31, 1)** (shown in **Table 7**) in the source code against capture tools. Therefore, if there is an attempt to save an image of a required protected application to the Clipboard, it is regarded as being captured and can be easily blocked using the technique presented in **Table 7**.



### 2.2.2 Agent-based anti-screen capture techniques against a remote attack

Next, considering the blocking of a remote capture by the remote tool, unlike Windows, Mac OS does not provide remote functionality as a standard feature. Thus, unlike Windows, remote tools are not versatile. As a result, it is relatively less vulnerable to threats of connecting remote locations and capturing screens.

Remote programs officially licensed under the MAC OS include **TeamViewer**, **LogMeIn**, **Devolutions Remote Desktop Manager**, and **Jump Desktop**. The Mac OS blocks remote programs by detecting the executable file names of the aforementioned remote programs using the agent and then minimizes the screen to prevent screen capture, similar to the blocking method used in the **Windows Terminal Service** presented in 2.1.3.

The process of this remote program can be obtained using the **GetNextProcess** and **GetProcessPID** functions.

### 2.2.3 Agent-based anti-screen capture techniques for the Clipboard and the Desktop

We introduced a method that blocks screen captures using the clipboard in Section 2.2.1. Here, we introduce a screen capture method by the Desktop, which is provided only by Mac. Screen capturing in the Mac OS occurs by simultaneously pressing three or four keys on the keyboard. It has the same function as the **PrtSc** key or **Alt+PrtSc** key of Windows. That is, if the three keys indicated in **Fig. 2** are simultaneously pressed, the capture screen is saved on the Desktop.



**Command+Shift+3**

**Fig. 2.** Capture method in the Desktop on Mac.

For a partial screen capture, “**Command+Shift+4**” is keyed and then the area to be captured is chosen. Therefore, the partially captured screen is saved on the Desktop.

Next, screen capture blocking by the Desktop in the Mac OS is presented as follows.

As previously indicated, when capturing the screen with the Clipboard in the Windows OS, it is possible to prevent screen capturing by determining the existence of the two picture files, **CF\_BITMAP** and **CF\_DIB**, and deleting them. Unlike the Windows OS, certain number values are generated in Mac OS. After verifying these values, screen capturing can be blocked. **Table 8** presents the technique used to block screen capturing using the default function values provided in Mac OS.

**Table 8.** Example of screen capture prevention method using Desktop on Mac OS

<p><b>CGSSetSymbolicHotKeyEnable (key, nProtect);</b></p> <p>This function determines whether a certain key is blocked. If the key value is 0, it is allowed; if it is 1, it is blocked. Thus, apply the following to block or allow screen capturing:</p> <p><b>Command+Shift+3:</b> Save a full screen on Desktop. <b>CGSSetSymbolicHotKeyEnable (28, 1); Block screen capture</b> <b>CGSSetSymbolicHotKeyEnable (28, 0); Allow screen capture</b></p> <p><b>Command+Shift+4:</b> Save a partial screen on Desktop. <b>CGSSetSymbolicHotKeyEnable (30, 1); Block screen capture</b> <b>CGSSetSymbolicHotKeyEnable (30, 0); Allow screen capture</b></p>
---

Therefore, CPM.exe of the agent-type developed in Mac OS retains the list of applications that need to be screen-protected, and CPM.exe uses the **CGSSetSymbolicHotKeyEnable(key, nProtect)** function to protect the screens of the applications according to the security policy. Therefore, if there is an attempt to save an image of a required protected application to the Desktop, it is regarded as being captured and can be easily blocked using the technique presented in **Table 8**.

3. Screen Capture Protection Techniques on Smartphones

Considering the advent of smartphones, screen capture prevention technology is being challenged. In particular, data leakage through screen capture by insiders is a new threat. With the expansion of mobile offices, personal smartphones or tablet PCs are used daily to browse the data of organizations or companies. In particular, senior executives or salespeople who have several external activities often search for e-mails through mobile devices and frequently view important attached files. In this case, they can capture all the displayed data with a simple click using the capture button provided by the mobile device. Screen capture by internal individuals is no longer an area that requires special technical knowledge to use or make a capture tool.

A few methods to prevent these include the installation of mobile device management (MDM) solution [14] or mobile application management (MAM) solution [15] to disable the camera capture function when a specific application is running. These methods remain to be the most widely used. Regarding the MDM, it has a function that disables the capture button within a specific area. For devices with MDM installed, the screen capture function provided by the manufacturer is stopped while the specific application is being operated, and remote screen capture is blocked; however, all the data on the phone can be reviewed by the MDM administrator. This type of control is difficult to apply for large corporations owing to conflicts with unions due to privacy issues. In particular, MDM has to run until it leaves a certain area, thereby inconveniently slowing down the smartphone and rapidly draining battery power.

MAM is relatively better; the “A” in MAM indicates an application, and the capture function stops only when a specific application is running. This method has been used more recently.

In the future, the use of mobile devices in work environments will increase to be similar to the PC usage rate. Therefore, the screen capture prevention techniques considered for the PC should also be considered at the same level for mobile devices. In particular, the attack of capturing the screen of a third PC or a third mobile device using a high-resolution smartphone should be prevented in the future.

In this section, we present defense techniques against attacks in which an insider leaks confidential business data displayed on a smartphone by screen capturing without permission or attacks by hackers injecting malicious codes into smartphones and then capturing the screens without permission and sending them externally.

The routes to capture and leak data on a mobile screen are similar to those of Windows or Mac. Those are as follows:

- Capture using the capture shortcut provided by a device
- Capture by using a capture tool
- Capture by using a remote capture tool
- Capture by using the Clipboard

The prevention of these capture techniques in Android and iOS is presented here.

### 3.1 Anti-screen capture techniques on Android

Considering the screen capture prevention techniques against capture tools in Mac OS, the captured image is ultimately saved via the Clipboard, thus data leakage by screen capture can be prevented by controlling only the Clipboard.

Similar to Mac, all Android smartphones released after 2016 provide a function to control captured events. Namely, all four capturable methods previously indicated can be controlled with one function.

For example, the simplest way to capture a screen on Android is to press the capture button provided by the device. To prevent this, screen capturing is blocked by applying a specific function (`getWindows().addFlags`) to the target screen aimed to be protected when it runs on the device. A practical example of using these functions is presented in [Table 9](#).

**Table 9.** Screen capture prevention method on Android [16]

**addFlags:** Screen capture prevention function **clearFlags:** Screen capture release function.

When blocking screen capture in a specific area, create API as follows: **onCreate.**  
**getWindows().addFlags(WindowsManager.LayoutParams.FLAG\_SECURE);**

When canceling screen capture in a specific area, create API as follows: **onCreate.**  
**getWindows().clearFlags(WindowsManager.LayoutParams.FLAG\_SECURE);**

**[onCreate()]**

Called when the activity is first created. This is where all the normal static should be set up: create views, bind data to lists, etc. This method also provides a bundle containing the activity's previously frozen state, if there was one.

The **API Level** is an integer value that uniquely identifies the framework of **API** revision offered by a version of the Android platform. The Android platform provides a framework **API** that applications can use to interact with the underlying Android system. Then, the function **AddFlags** was added from **API Level 17**, which indicates it was recommended for

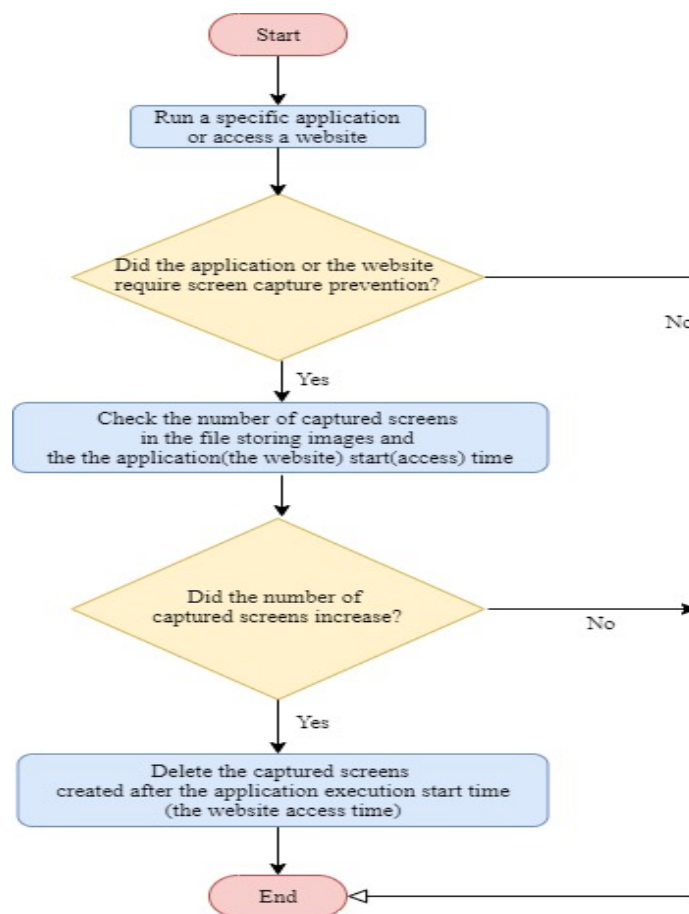
use in 2013. As a result, **API Level 17** contains an anti-screen capture function. However, users started applying this function near 2016. It is presumed that this is because the devices with that function were applied from 2016 by the manufacturers.

Depending on the smartphone manufacturers, when the screen capture prevention function is activated, a toast message saying “The screen cannot be captured according to the security policy” is displayed, or no action occurs.

In conclusion, if the screen capture prevention function provided by the Android OS is inserted at the source level in the application to be protected, the function will automatically make the screen appear black or print the toast message provided by the manufacturer when hackers or insiders try to capture by a capture tool, clipboard, or remote access.

In [Fig. 3](#), we will briefly introduce another method used to prevent screen capturing when the device capture button was pressed prior to 2016.

This method was possible because the folders that were saved after capturing were fixed in devices released before 2016. However, since 2016, the location of stored files has been diversified; moreover, as manufacturers supply functions to detect and control captured events by the **AddFlags** function, devices can be easily prevented against capture events. However, this also eliminated the technological level difference between companies that provide mobile screen security technology.



**Fig. 3.** Capture prevention technique against a capture shortcut [17].

### 3.2 Anti-screen capture techniques on iOS

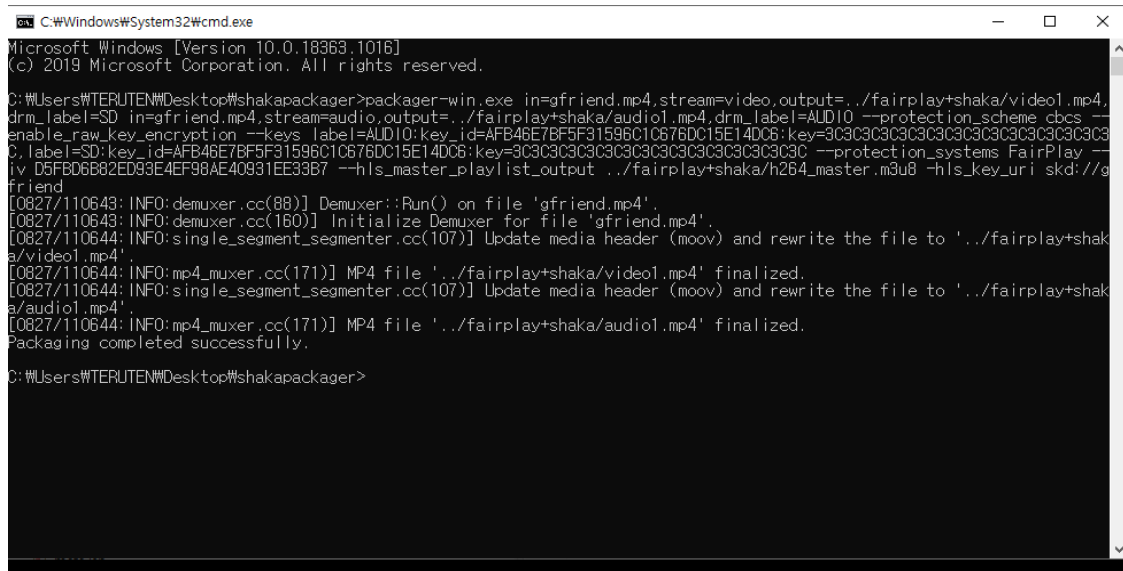
As previously indicated, all Android phones released after 2016 can control capture events for each application using the (`getWindows().addFlags`) function. A profile file performs the same function in iOS, which does not support multiple processes. Therefore, the capture tool cannot capture the screen. In conclusion, we can prevent screen capture by simply blocking the capture button. As shown in [Table 10](#), to prevent screen capturing using the capture button provided by the iPhone, the **profile file** can be used, which is a file that manages the iOS security policy and is an environment file for device control provided by Apple.

**Table 10.** Screen capture prevention method on iOS [18]

**Step1 :** find `allowScreenShot` in Profile provided by Apple.  
**Step2 :** Put True/False as follows:  
`allowScreenShot = TRUE (Allow capture) FALSE (Block capture)`

However, there is one problem with using this method. After applying the capture blocking policy in Profile, all the captured functions will not be available in all applications. This indicates that all the applications on the iPhone can either be captured or not captured. As a result, this may be inconvenient for users.

In addition, iOS provides a video capture blocking technique. The technique is presented in [Fig. 4](#) and [Fig. 5](#). This technology is provided by Apple and blocks only video capture. It is called **FairPlay** [19].



```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.18363.1016]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\TERUTEN\Desktop\shakapackager>packager-win.exe in=gfriend.mp4,stream=video,output=../fairplay+shaka/video1.mp4,
drm_label=SD,in=gfriend.mp4,stream=audio,output=../fairplay+shaka/audio1.mp4,drm_label=AUD10,--protection_scheme cbcs --
enable_raw_key_encryption --keys_label=AUD10:key_id=AFB46E7BF5F31596C1C676DC15E14DC6:key=3C3C3C3C3C3C3C3C3C3C3C3C3C3C3C3C
C,label=SD:key_id=AFB46E7BF5F31596C1C676DC15E14DC6:key=3C3C3C3C3C3C3C3C3C3C3C3C3C3C3C3C --protection_systems FairPlay --
iv D5FB06B82ED93E4EF98AE40931EE33B7 --hls_master_playlist_output ../fairplay+shaka/h264_master.m3u8 --hls_key_uri skd://g
friend
[0827/110643: INFO: demuxer.cc(88)] Demuxer::Run() on file 'gfriend.mp4'.
[0827/110643: INFO: demuxer.cc(160)] Initialize Demuxer for file 'gfriend.mp4'.
[0827/110644: INFO: single_segment_segmenter.cc(107)] Update media header (moov) and rewrite the file to '../fairplay+shak
a/video1.mp4'.
[0827/110644: INFO: mp4_muxer.cc(171)] MP4 file '../fairplay+shaka/video1.mp4' finalized.
[0827/110644: INFO: single_segment_segmenter.cc(107)] Update media header (moov) and rewrite the file to '../fairplay+shak
a/audio1.mp4'.
[0827/110644: INFO: mp4_muxer.cc(171)] MP4 file '../fairplay+shaka/audio1.mp4' finalized.
Packaging completed successfully.

C:\Users\TERUTEN\Desktop\shakapackager>

```

**Fig. 4.** VOD content encryption packaging tool of the **FairPlay**.

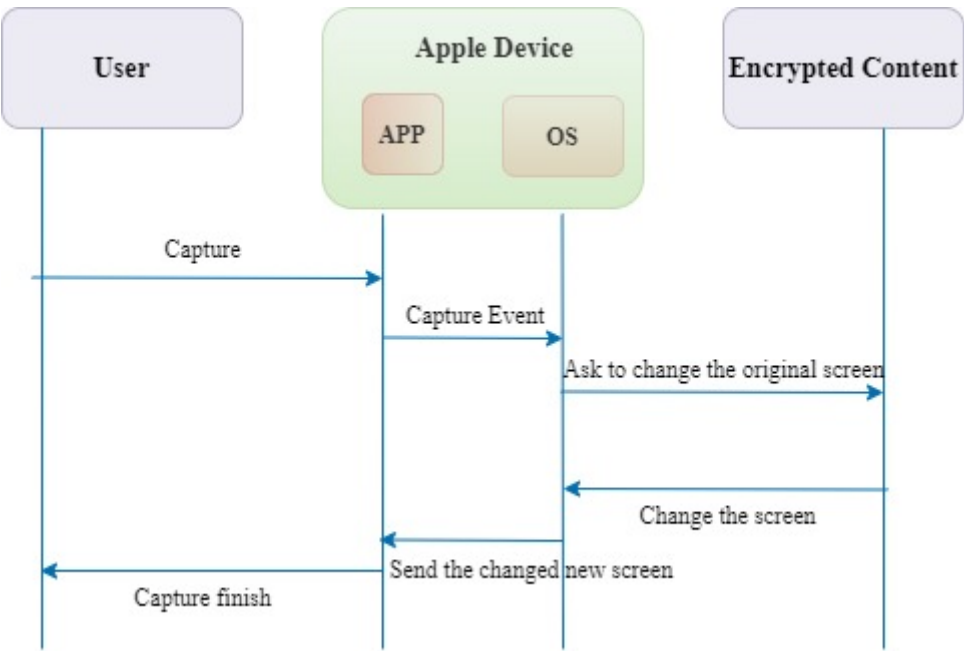
To use the **FairPlay** technology, the original video content must achieve packing using an encryption tool provided by Apple, which consists of the Dos commands presented in [Fig. 4](#). The main variable values in [Fig. 4](#) are as follows:

**Table 11.** Implications of the main variable values of the **FairPlay** encryption packaging tool

(packager-win.exe in)	=original video content
(stream=video):	implies video content
(output=C:\DRMFolder\Video DRM.AVI):	content saving location
(drm_label=Apple):	content owner
(stream=video, output):	saved location and name of encrypted VOD
(stream= AUDIO):	implies audio content
(output= C:\DRMFolder\SoundDRM.mp3):	content saving location
(stream= AUDIO, output):	saved location and name of encrypted AOD
(hls_master_playlist_output):	saved location and name of the master file (VOD+AOD, to be serviced file)

Videos encrypted by **FairPlay**'s encryption packaging tool (Fig. 4) are serviced as follows after distribution:

The technique presented in Fig. 5 is the same as the general screen capture prevention technique indicated in the previous sections. Namely, the captured screen was replaced with another screen. To achieve this, it is necessary for the **FairPlay** agent to obtain the location where the captured screen is finally saved and to change the image to another image. Perhaps iOS reads the extension of the content encrypted with the **FairPlay** packager and saves the captured image to a specific folder when a capture event occurs while the content is playing.



**Fig. 5.** Capture prevention process of content packed by **FairPlay**.

The technique presented in Fig. 5 is provided only for videos. For images, one must proceed through the aforementioned Profile setting. However, the capture prevention of video content can be accomplished without affecting other applications owing to not setting the Profile function.

#### 4. Comparison of Anti-Screen Capture Techniques

We previously reviewed the screen capture methods that can occur on a PC and the corresponding anti-screen capture techniques in Section 2. The agent-type capture prevention module was developed based on the detours hooking technique and installed on the PC to defend against various screen capture attacks. In Section 3, we examined the types of screen capture and the corresponding anti-screen capture techniques in mobiles. Unlike the PC, in mobile devices, the OS detects and controls all of the capture events of the device. Therefore, it was relatively easy to develop a capture prevention module only by setting specific functions provided by the OS. This chapter presents a relative comparison table for the techniques presented in Sections 2 and 3. **Table 12** summarizes the capture prevention techniques for each device and OS.

**Table 12.** Summary and comparison of screen capture prevention techniques

Device	PC		Smartphone	
OS	Windows	Mac	Android	iOS
<b>Full Screen</b>	Blocking by examining the handle value using the hooking technique of the detours method	By using <b>CGSSetSymbolicHotKeyEnable(29, 1)</b> , and blocking	By using <b>getWindows().addFlags: Block</b>  <b>getWindows().clearFlags: Allow</b>	None
<b>Partial Screen</b>	Use the technique used in the full screen (TFS) in the area to be partially captured	By using <b>CGSSetSymbolicHotKeyEnable(31, 1)</b> , and blocking		
<b>Remote</b>	By the <b>EnumDisplayDevices</b> function, check the existence of <b>DISPLAY_DEVICE _ATTACHED_TO_DESKTOP</b> or <b>DISPLAY_DEVICE _MIRRORING_DRIVER</b> . If yes, VNC connection is blocked by TFS. In the case of Terminal Service of MS, screens are minimized and blocked.	By the <b>GetNextProcess</b> and <b>GetProcessPID</b> functions, check the existence of <b>TeamViewer, LogMeIn, Devolutions Remote Desktop Manager, Jump Desktop</b> . If yes, screens are minimized and blocked.		
<b>Clipboard</b>	If there is a <b>CF_BITMAP</b> or <b>CF_DIB</b> file in Clipboard, it is blocked by changing the image.	By using <b>CGSSetSymbolicHotKeyEnable(29, 1)</b> , and blocking <b>CGSSetSymbolicHotKeyEnable(31, 1)</b> , blocking		None



Capture shortcut or Capture button of a device	When capturing by pressing PrtSc or Alt+PrtSc, it is blocked by Clipboard method.	When capturing by pressing shortcut, block by using <b>CGSSetSymbolicHotKeyEnable(#, 1)</b> function.	Blocking by <b>getWindows().addFlags</b>	Blocking by <b>Profile.mobileconfig</b>
Memory Usage	5MB	Use nothing	No use, blocking capture in OS	No use, blocking capture in OS
CPU Usage	Within 1%	Use nothing	No use, blocking capture in OS	No use, blocking capture in OS
CPM Type	.exe, .dll	.pkg	.jar	By setting in Profile.mobileconfig
File Size	15MB	20MB	Within 1MB	Within 1MB

In addition, we explain how the techniques proposed and described herein differ from those used in existing products.

The data security area can be broadly divided into file encryption, input/output control, such as keyboards, and data leakage prevention by screen capture. Among them, screen capture prevention techniques were not in great demand in the market. Therefore, they began with preventing screen capture through the mouse right-click.

Subsequently, as various screen capture tools became popular, the screen capture prevention technique developed into a method of forcible termination, where the corresponding capture tool was operated by converting the capture tool's names into a database using the blacklist method. However, in this case, the blacklist method was bypassed by simply replacing the capture tool's name. Furthermore, products that combine methods of registering specific processes of capture tools in a blacklist, rather than simple programs or file names, have emerged. However, with the release of many different capture tools, these methods have had limitations in DB conversion, and cannot prevent non-commercial capture tools produced by professional hackers with advanced technologies. Therefore, herein we propose an agent-based anti-screen capture technique for Windows, which is controlled by hooking the basic functions used by capture tools to perform the capturing, whether commercial or non-commercial, as shown in [Fig. 1](#).

The proposed method can be implemented in Windows. However, in the case of Mac, hooking technology is not available and the number of officially licensed remote programs is also limited. Therefore, there may be few technical differences from those of existing products. One distinct difference from the technique used in existing products on the Mac is that it detects and blocks keystrokes used to capture. Therefore, we provide an additional blocking function when the blocking method based on the clipboard chain monitoring is broken.

In addition, in the case of mobile devices, manufacturers provide APIs that control screen capture in a simple way, unlike PCs. Therefore, the techniques we have presented herein might have no significant differences from those of existing products. However, it is meaningful that this study presents an integrated screen capture prevention technique for PC (Windows, Mac) and Mobile (Android, iOS).

Screen capture prevention techniques have made rapid progress in recent years due to the needs of customers in the commercial market. The two main reasons are the 4th Industrial Revolution and COVID-19. We also found that there are not many international capture prevention products compared to domestic ones. Moreover, most of the capture prevention

companies in Korea are investing a lot of manpower to implement the screen capture prevention functions by controlling the menus of a specific product. The screen capture prevention market in Korea is divided into the blacklist type product market and system integration (SI) market.

More detailed comparison results are described in [Tables 13 and 14](#).

**Table 13.** Technical differences on Windows

	<b>Windows</b>	
	<b>Existing product techniques</b>	<b>Suggested techniques</b>
<b>Full Screen</b>	The list of capture tools is managed as blacklist, and if the tool is executed, the process of the capture tool is forcibly terminated.	Hooks the API used when the capture program captures an image and then takes it to its own memory. Thereafter, the entire captured image is converted to an anti-capture image. The main advantage of this method is that you can control the capture functions of all capture tools running on Windows OS, and not just a specific capture program.
<b>Partial Screen</b>	Same method as full screen capture. However, from all the captured images, only the part to be protected is converted to an anti-capture image.	Same method as full screen capture. However, from all the captured images, only the part to be protected is converted to an anti-capture image.
<b>Remote</b>	The list of remote programs is managed as blacklist, and when the program is executed, the remote program's process is forcibly terminated.	In the case of commercial remote programs such as <b>TCO Stream</b> and <b>Team Viewer</b> , after confirming that the BitBlt function is executed for screen capture, the entire captured image is converted to an anti-capture image. In the case of virtual network computing (VNC) using a "Mirror Driver" for capture, it detects the driver use and minimizes the screen of the program to be protected. When using Microsoft Terminal Service, it detects the use of the <b>EnumDisplayDevices</b> function and changes the entire captured image to an anti-capture image. The main advantage of this method is that you can control the capture function of all remote tools running on Windows OS, and not just a specific remote tool.
<b>Clipboard</b>	The clipboard contents are monitored using the clipboard chain, and a blocking action is executed when there is an attempt to add new contents to the clipboard.	Hook the API that puts contents on the clipboard or gets contents from the clipboard. If the obtained attributes type of the contents is CF_BITMAP or CF_DIB, blocking action is executed.

<b>Remark</b>	The blacklist method has limitations in registering all existing commercial capture tools or remote programs. In addition, non-commercial professional programs cannot be supported. Therefore, it cannot perfectly prevent screen capture.	This technique is not affected by specific capture tools or remote programs. Therefore, it can cope with the screen capture tools that might be developed in the future.
---------------	---	--

**Table 14.** Technical differences on Windows

	<b>Mac</b>	
	<b>Existing product techniques</b>	<b>Suggested techniques</b>
<b>Clipboard</b>	The clipboard contents are monitored using the clipboard chain, and a blocking action is executed when there is an attempt to add new contents to the clipboard.	<p>The clipboard contents are monitored using the clipboard chain, and a blocking action is executed when there is an attempt to add new contents to the clipboard.</p> <p>In addition, it detects the keystroke for capture and blocks the operation of the keystroke.</p> <p>e.x.)</p> <p>CGSSetSymbolicHotKeyEnable (key, nProtect);</p> <p>This function determines whether a certain key is blocked.</p> <p>If the key value is 0, it is allowed; if it is 1, it is blocked.</p> <p>CGSSetSymbolicHotKeyEnable (29, 1);</p> <p>Block screen capture</p> <p>CGSSetSymbolicHotKeyEnable (29, 0);</p> <p>Allow screen capture</p> <p>CGSSetSymbolicHotKeyEnable (31, 1);</p> <p>Block partial screen capture</p> <p>CGSSetSymbolicHotKeyEnable (31, 0);</p> <p>Allow partial screen capture</p>

## 5. Conclusion and Future Work

In this study, we presented various methods for screen capture (screen recording is also a continuous screen capture) and proposed techniques that can defend against these attacks.

For the Window OS, as a hooking technique is allowed; a technique that prevents the hacking of all application screens is proposed as an agent-type mainly developed using the Detours method provided by Microsoft among the three hooking techniques..

For the Mac OS, because it does not support the hooking technique and allows all images captured by the capture tools to be saved in the Clipboard, except for the Desktop capture, anti-screen capture techniques were relatively simpler compared to Windows.

For Android, it is possible to prevent the screen capture relatively simply by using the function provided in the API Level 17 from 2013. It is possible by setting the profile file in the case of iOS. In particular, for iOS, video capture blocking technology is provided using FairPlay without setting Profile; however, to use it, video content must be prepacked, and this

technique is only applied for VOD.

In this study, we introduced screen capture methods by insiders and external attackers and corresponding anti-screen capture techniques for the PCs of Windows OS, Mac OS, and for Android OS and iOS devices. As a result, complete screen capture prevention techniques that included PCs and mobile devices were presented.

Furthermore, this study did not provide an anti-screen capture solution against shooting and capturing another device's screen using a smartphone. Until now, there has been no technical prevention method against shooting and capturing the screen of a PC or other devices using a smartphone. To prevent this, a single-use security sticker was attached to the camera, a mobile MDM was installed, or a monitor watermark solution containing personal information was used before. However, these are not sufficient solutions for blocking data leakage on the screen. Considering the increase in the number of smartphone users, anti-screen capture techniques against smartphones must be developed. In the near future, we will propose a technique to close the screen by detecting the shooting behavior of smartphone cameras using the AI method [20, 21]. If the AI-based module is integrated with the agent introduced in Section 2, it can block every screen capture attack with a single module in a PC.

## References

- [1] J. A. Siegel and P. J. Saukko, *Fingerprints*, in Encyclopedia of Forensic Sciences, 2nd ed, Oxford, UK: Elsevier Science & Technology, pp. 346-351, 2013.
- [2] E. Diehi, *Securing digital video techniques for DRM and content protection*, Berlin, German: Springer Berlin Heidelberg, 2012. [Article \(CrossRef Link\)](#)
- [3] F. Rahman, M. Farmani, M. Tehranipoor and Y. Jin, "Hardware-assisted cybersecurity for IoT devices," in *Proc. of 18th International Workshop on Microprocessor and Social Testing, Security and Verification*, TX, USA: MTV, pp. 51-56, 2017.
- [4] M. Rouse and J. Madden, "Desktop virtualization," Tech target.
- [5] M. Ashish and D. Vedvyas, *Watermarking Techniques for Copyright Protection of Videos*, Springer International Publishing, 2019. [Article \(CrossRef Link\)](#)
- [6] D. Y. Jang, *Windows Architecture and Principles: Principles of Programming through the OS*, Seoul, Korea: Hanbit Media, 2006, Ch. 14.
- [7] C. Petzold, *Blocking by examining the handle value using the hooking technique of the detours method*, in *Programming Windows*, 5th ed, Washington, USA: Microsoft Press, 1998, Ch. 21, pp. 1243-1266.
- [8] J. Richter, *Advanced Windows Third Edition*, Microsoft Press, 2006.
- [9] E. N. Dekker and J. M. Newcomer, *Developing Windows NT Device Drivers*, Microsoft Press, 1999.
- [10] J. Prosise, *Programming Windows with MFC*, Washington, USA: Microsoft Press, 2001.
- [11] C. Petzold, *Bitmaps and bitblts*, in *Programming Windows*, 5th ed, Washington, USA: Microsoft Press, 1998, Ch. 14, pp. 641-711.
- [12] Y. Lee, D. H. Hwang and S. W. Kwak, "Security program and computing device for providing security function per each user session and method therefore," ROK. Patent 10 2019 0025814, Mar. 7, 2019.
- [13] C. Petzold, *The clipboard*, in *Programming Windows*, 5th ed, Washington, USA: Microsoft Press, 1998, Ch. 12, pp. 567-590.
- [14] M. Pierer, *Mobile Device Management*, Wiesbaden: Springer Fachmedien Wiesbaden, 2016.
- [15] S. G. Yoon and D. H. Hwang, "Method for managing secure access of mobile application," ROK. Patent 10 1420 3830000, Oct., 16, 2012.  
[Online]. Available: <http://www.ncbi.nlm.nih.gov/pubmed/3830000>
- [16] K. Makan and S. Alexander-Bown, *Android Security Cookbook*, Birmingham, UK: Packt Publishing Limited, 2013.

- [17] J. H. Yu, “Apparatus and method for scene (screen) capture security,” ROK. Patent 10 1403 6950000, May, 28, 2014.
- [18] D. Till, J. Lee and S. Lee, *iOS Application Security*, Seoul, Korea, Acorn Publishing Co., 2017.
- [19] D. Eran, “How Fairplay Works: Apple’s iTunes DRM Dilemma,” Internet archive\_Available: (archive.org) 3, Mar., 2007.  
[Online]. Available:  
<https://web.archive.org/web/20170927030905/http://www.roughlydrafted.com/RD/RDM.Tech.Q.1.07/2A351C60-A4E5-4764-A083-FF8610E66A46.html>
- [20] Y. Lee and D. H. Hwang, “Computer program to prevent leakage of information on display device and security service using it,” ROK. Patent 10 2018 0020509, Feb., 21, 2018.
- [21] Y. Lee and Y. Y. Yu, “Computer program for preventing information spill displayed on display device and security service using the same,” ROK. Patent pending 10 2020 0053260, May, 4, 2020.



**Young Lee** received the M.S. degree from the KAIST, Republic of Korea and she is currently pursuing the Ph.D. at Dept. of Mathematical Sciences in KAIST. Her research interests are data protection, IoT security and blockchain technology.



**SangGeun Hahn** received the Ph.D degree from Ohio State University in 1987. He is currently a professor of Mathematical Sciences and adjunct professor of Graduate School of Information Security at KAIST, Republic of Korea. His research interests are cryptography, social network and post-quantum cryptography.