Bypassing the Integrity Checking of Rights Objects in OMA DRM: a Case Study with the MelOn Music Service

Jusop Choi[1], William Aiken[2], Jungwoo Ryoo[2], Hyoungshick Kim[1]
[1]Computer Science and Engineering, Sungkyunkwan University, Republic of Korea
[2]Information Science and Technology, Pennsylvania State University, USA
[1]{cjs1992, hyoung}@skku.edu
[2]{wva5029, jryoo}@psu.edu

ABSTRACT

Commercial digital music is typically distributed in the music source market via Digital Rights Management systems (DRM). DRM systems help remotely control the music contents. The Open Mobile Alliance (OMA) DRM became the de facto standard after major market adoption because of its support for a wide variety of different business and usage models. In OMA DRM, a popular business model is a (monthly) subscription enforced by controlling the period of playback time; once the given period of time expires, the music cannot be played. In this paper, we demonstrate how to bypass the integrity checking of the rights object in the OMA DRM system through a case study of MelOn (a well-known music distribution service in South Korea) by reverse engineering its media player equipped with a DRM agent.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information**]: Security and Protection—*Digital Rights Management*; D.2.7 [**Software Engineering**]: Distribution and debugging—*Restructuring, reverse engineering, and reengineering*

Keywords

OMA DRM, MelOn, bypassing, reverse-engineering

1. INTRODUCTION

Digital Rights Management (DRM) is a fundamental technology that makes today's music distribution market possible [14]. DRM technologies allow content providers and publishers to control the whole distribution chain and apply flexible usage rules [8]. Content providers distribute their digital contents online, but after its distribution they can still exercise control of those contents via DRM [13]. DRM systems have been designed to prevent unauthorized access to and use of protected digital content.

For example, DRM systems permit users to play digital music within a predetermined time interval and/or for a predetermined number of times. Such permission/constraint parameters are typically included in a DRM-protected file and governed by a DRM agent that is strongly coupled with a music player to access the protected file [9]. To achieve this, DRM-protected files are securely encrypted with a cryptographic key, which can only be decrypted by authorized DRM agents under specific usage rules and conditions.

However, the effectiveness of these DRM technologies in practice is still controversial. Many security experts agree that secure implementation of DRM technology is very tricky, and many commercial DRM systems have proven insecure (e.g., against memory analysis [16]). To make matters worse, even with a secure DRM implementation, there exists an inevitable problem called the *analog hole* [13] which is the duplication of DRM-protected contents by analog means (e.g., recording the analog sound which is produced by a music player).

OMA (Open Mobile Alliance[1]) DRM is an open-standards based framework for DRM technologies that supports a wide variety of different usage models, and it has now been widely adopted as the de facto standard for many business applications in the mobile industry [8]. In OMA DRM, one of the most popular business models is a (monthly) subscription by controlling the period of playback time; once the given period of time expires, the music cannot be played. In this paper, we analyze the security of this usage model through a case study of MelOn (a well-known music distribution service similar to iTunes), which has controlled about 39.1% of the online music Korean market share since 2009. According to a 2006 survey [11], the Korean digital music market was worth 191 million USD at that time [11], and it has continuously grown since then. In MelOn, digital music content is securely distributed in the encoded DRM Content Format (DCF), a file format defined by OMA DRM.

This paper examines how to forcibly modify the permissions and constraints (e.g., the time interval constraint) in an OMA DCF file by disabling the integrity checking of the MelOn media player. The integrity check can be bypassed by removing a conditional jump instruction at runtime. By doing this, DRM-protected contents can be used without any usage constraint (e.g., the time interval constraint can be set to any value). Our implementation illustrates how difficult it is to securely implement a DRM agent in the real-world.

This paper is structured as follows. In Section 2, we provide additional information about DRM and OMA DRM used by MelOn. In Section 3, we show how to bypass the OMA DRM protection with a modified DCF file. We present a discussion on countermeasures to mitigate such attacks in Section 4. In Section 5, we discuss the ethical and legal implications of this work. And lastly, in Section 6, we provide our overall conclusion.

IMCOM '16, January 04-06, 2016, Danang, Viet Nam

DOI: http://dx.doi.org/10.1145/2857546.2857609

[1]http://www.openmobilealliance.org

2. BACKGROUND

DRM technology aims to achieve the reliable and efficient distribution of digital content over the Internet by providing a trustworthy system to securely enforce the rights of the copyright holders, distribution dealers, and content creators [16].

In terms of DRM, content is defined as valuable intellectual property that should be protected from access by unauthorized users. A user is a subject who can access DRM contents under granted "rights" and "conditions". The "rights" are the permissions granted for DRM content, and the "conditions" include requirements for (and limitations of) how the permissions can actually be executed. By granting specific rights and conditions, content providers can control distributed digital contents as they want. Surely, these specified rights and conditions should not be able to be directly modified by unauthorized entities (e.g., the users). In general, protection from such modifications can be achieved with the implementation of cryptographic algorithms. The use of DRM contents can also be traced and monitored [14].

The main components of DRM typically include the following: the DRM agent, the Content Issuer (CI), the Rights Issuer (RI), and the user [3]. In particular, the roles of these components are very important since the overall security of a DRM system depends on the secure implementation of a DRM agent and its relationship to the other components. The DRM agent is responsible for enforcing permissions and constraints associated with DRM content as well as controlling access to that content; the CI manages the secure delivery of DRM content; the RI defines permissions and constraints associated with DRM contents and creates a rights object for expressing them; and the user is the consumer of the DRM content [3].

DRM solutions are generally classified into non-cryptographic and cryptographic techniques. Non-cryptographic DRM techniques usually operate based on identifying the user by physical aspects, usually relying on manuals, disks, or other hardware for authentication. On the other hand, cryptographic DRM techniques operate based on confirming that the user has certain access rights, using a cipher when a user tries to access any DRM content. This approach is especially useful when content distribution is done digitally, and as a result, most recent DRM services have been launched based on cryptographic techniques [16].

The digital music market has been one of the biggest adopters of cryptographic DRM techniques. Since digital files can be easily redistributed (e.g., through peer-to-peer services), many music marketplaces have no choice but to use DRM systems to enforce their end-user license agreements. However, even with DRM, some possibilities of unauthorized distribution still exist [13, 16]. Moreover, DRM is unpopular with end-users because DRM makes digital contents very uncomfortable to use and causes compatibility issues with various devices, media players, and media file formats [6, 10]. Nonetheless, music source companies still want to maintain their DRM systems because the use of DRM systems are currently regarded as the best efforts to limit unauthorized use of their digital content (e.g., unauthorized distribution to others).

2.1 OMA DRM

OMA DRM was invented by the Open Mobile Alliance with several stakeholders, such as content providers, licence providers, and device manufacturers to support a wide variety of different usage models.

The OMA DRM 1.0 specification was first released in 2002 as the world's first open DRM standard for mobile devices [2]. As a follow-up, OMA DRM 2.0 was developed in 2006, with a focus on a wide variety of different distribution mechanisms to enhance end-user experiences. The DRM architecture document particularly specified a file format named the DRM Content Format (DCF) to represent a file format for securely downloading and controlling DRM contents, which are based on the ISO base media file format (ISO14496-12) [2].

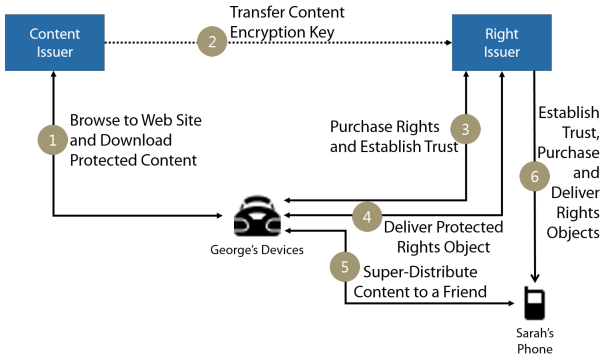


Figure 1: The OMA DRM 2.0 functional architecture (adapted from [2]).

The functional architecture of OMA DRM 2.0 is illustrated in Fig. 1. As shown in this figure, when a user downloads DRM-controlled content from the CI, the associated rights object, which includes a cryptographic key for accessing the DRM content, can be delivered from the RI in a separate manner. Also, the DRM content can be re-distributed in a local network through a distribution mechanism called *super-distribution*. Those delivery mechanisms were designed to enhance flexibility and convenience for sharing DRM-protected contents through peer-to-peer networks [2].

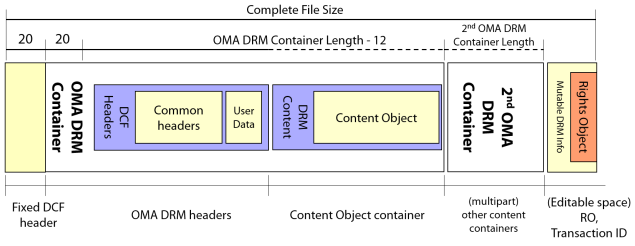


Figure 2: Structure of DCF (adapted from [1]).

Fig. 2 shows the structure of a DCF file. As shown in this figure, the DCF file contains one or more containers that comprise the encrypted digital content (i.e., the content object) with meta-data that represent information such as author, title, media type and the URL to obtain the associated rights object to unlock the content. The rights object is typically implemented as an XML file that describes the permissions and constraints granted to a DRM agent when accessing a specific DCF file. Various permissions and constraints can be defined by Rights Expression Language (REL) [12]. The permission information specifies how a digital content can be accessed such as play, display, execute, print, and export; the constraint information specifies the number of times the content can be accessed and/or the expiration time of the content. The rights object also contains the content encryption key needed to decrypt the encrypted digital content. The rights object is integrity-protected by a Message Authentication Code (MAC) such as `HMAC-SHA1` and contains a list of content object identifiers and their respective

usage permissions. The MAC key is securely protected using a PKI mechanism. Only the authorized DRM agent with its own private key can access the MAC key and the content encryption key [15].

2.2 MelOn Service

MelOn[2], a popular online music store in South Korea, uses OMA DCF files to securely distribute its music files and control them with predefined permissions and constraints (see Section 2.1). In this service, downloaded DCF files can only be played with a paying user's MelOn music player (see Fig. 3) — the MelOn player, a program which can be downloaded from the MelOn website, is necessary to use MelOn services.

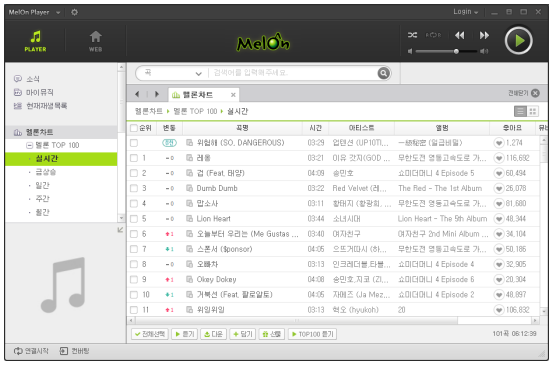


Figure 3: MelOn player interface.

MelOn launched a service called "*Free-Club*" to offer unlimited downloads and unlimited music streaming on a monthly basis. If the user's subscription is expired and is not renewed, then the user's MelOn music player will no longer play the downloaded DCF files when the expiration date of the user's subscription, explicitly given in those files, has passed.

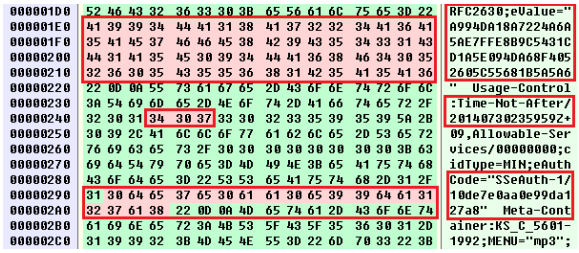
3. BREAKING THE DRM PROTECTION IN MELON

Before playing a DCF file, the DRM agent, embedded in the MelOn media player, checks whether the DCF file should be played with its current permissions and constraints. For example, once the downloaded DCF file is past its expiration date, that DCF file cannot be played anymore. In this paper, however, we show how difficult it is to deploy an access policy-based DRM implementation in this way by breaking the DRM protection in the MelOn player. In our experiments, we used a PC (with running 32-bit Windows 7).

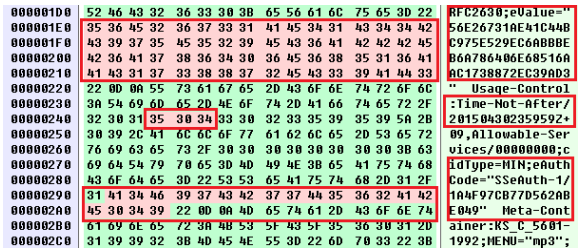
Overall, there are two ways of breaking the OMA DRM protection; we need to bypass either (1) the checks for permissions and constraints on a DCF file or (2) the check for the integrity of the rights object in a DCF file. In general, the check codes for (1) can be located in several different regions (e.g., the check code for subscription expiration time is different from the check code for count limitation). Therefore, for a more generalized breaking method, we focus on how we bypass the check code for (2).

This process can be subdivided into two phases: (A) identifying the locations of permissions and constraints in a rights object, and (B) disabling the integrity check in the MelOn media player application. Each of these phases will be discussed in the following subsections.

The first step is to identify the locations of permissions and constraints, including (but not limited to) the subscription expiration date, to be modified in a DCF file. To achieve this goal, we can compare the headers of DCF files with different permissions and constraints.



(a) `Time-Not-After` = 20140730 · · ·



(b) `Time-Not-After` = 20150430 · · ·

Figure 4: Comparison of two DCF files that can be played until July 30, 2014 and until April 30, 2015, respectively. The red boxes represent the differences (`eValue`, `Time-Not-After`, and `eAuthcode`) between those DCF files.

Fig. 4 shows the comparison results between DCF files that can be played until July 30, 2014 and April 30, 2015, respectively. We attempted to identify the differences between these two DCF files using a hex editor. The differences between the two files are `eValue`, `Time-Not-After`, and `eAuthcode` (see the red boxes in Fig. 4). With those different parts, we examined each part one-by-one and found that `Time-Not-After` (see the middle box in each figure) represents the subscription expiration date consisting of the year, month, day, hour, minute, and second. Since the digest length for `HMAC-SHA1` used in OMA DRM is 20 bytes (see Section 2.1), we surmise that `eAuthcode` is the MAC value used to ensure the integrity of the rights object in the DCF file.

When the locations of permissions and constraints are exactly identified, it becomes easy to play such DCF files without any restriction (e.g., expiration time constraint) since those permissions and constraints might be specified with any desired values. As we already described in Section 2.1, however, the integrity of the rights object in a DCF file is securely protected by its MAC. That is, the DRM agent, embedded in the MelOn music player, always checks if any of the permissions and constraints included in a DCF have been maliciously modified before playing the DCF file. Therefore, we need to disable the integrity check of the rights object in the OMA DCF files if we want to use it without any usage constraint.

3.1 Identifying the Locations of Permissions and Constraints

We performed a dynamic analysis of the MelOn media player with two popularly used tools: (1) we used OllyDBG[3] to obtain a list of all referenced strings by our target keywords such as `eValue`,

[2]http://www.melon.com/index.htm

[3]http://www.ollydbg.de/

Time-Not-After, and eAuthcode; and (2) we used IDA Pro[4] to figure out the check code for the integrity of the rights object in a DCF file and analyze its behaviors. Through this dynamic analysis, we finally identified our target codes from the MelOn media player.

3.2 Disabling Integrity Check

For checking the integrity of a rights object in a DCF file, shown in Fig. 5, the DRM agent obtains some header information, such as cid type, Time-Not-After, allowable-services and the MAC key. Next, the DRM agent calculates the MAC value using HMAC-SHA1 with the obtained strings and values from the DCF file. Finally, the DRM agent compares the calculated MAC value with eAuthCode included in the DCF file. If they are identical, the rights object must not have been modified. The encrypted content in the DCF file can then be decrypted and accessed by the DRM agent. Otherwise, the rights object is identified as modified, and the DRM agent handles the DCF file as a corrupted file.

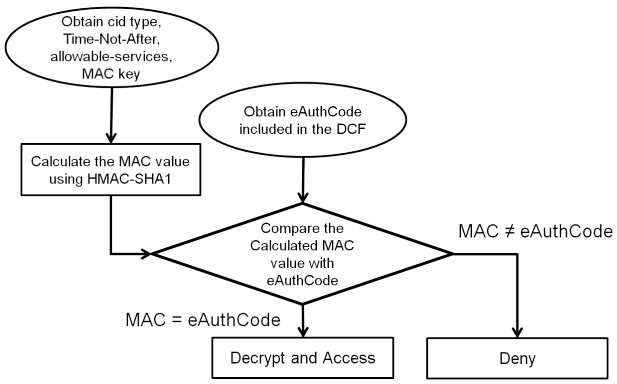


Figure 5: Procedure of Integrity Checking of Rights Objects

Fig. 6(a) shows the MAC comparison procedure at the assembly code level in the MelOn media player. We can see that the control flow can be changed at the "conditional jump" (jnz) instruction (see Line 1 in Fig. 6(a)): the difference between the calculated MAC value and eAuthCode is compared with 0; if they are not same, the control flow is changed to an error-handling routine by the jnz, short for Jump if Not Zero, instruction (see Line 2 in Fig. 6(a)); otherwise, the program executes the following instructions in sequence normally.

```
cmp     [ebp+Dest], 0
jnz     short loc_7FEBD4
mov     ax, 1
jmp     loc_7FEEE2
```

(a) Original player

```
cmp     [ebp+Dest], 0
nop
nop
mov     ax, 1
jmp     loc_7FEEE2
```

(b) Cracked player

Figure 6: MAC verification procedure in the MelOn media player is disabled by replacing the jnz instruction with nop instructions.

Therefore, we can bypass this MAC verification procedure by replacing the jnz instruction with nop, short for No Operation, instruction in a simple manner (see Fig. 6(b)).

The jnz instruction is two bytes long. That is, we should replace this instruction with other instructions that are also two bytes long in order to keep the other instructions at the same positions in the original code. In this paper, we use two nop instructions (0x90) to achieve this goal without changing any semantic of the original code[5] With this technique, we cracked the MelOn media player in order to verify the feasibility of the proposed attack and found that the cracked MelOn media player runs normally with any DCF file even when its rights object is modified (e.g., the value of Time-Not-After is intentionally extended).

4. COUNTERMEASURES

By a reverse engineering analysis of the MelOn media player, we managed to reveal the structure of its machine codes and found a method to modify the assembly code to bypass security checks enforced by the DRM agent. Thus, a straightforward recommendation would be to use code obfuscation [4] and/or anti-debugging [5] techniques in order to hide critical codes and control flows. For example, a packing algorithm called "Alternate EXE Packer"[6] provides a reasonable level of protection against debugging (see Fig. 7) — our disassembling program (i.e., IDA Pro) cannot produce decompiled codes with any meaningful string that provide clues to the functionality and various system calls made by codes. This would make the DRM agent program harder to analyze.

```
dd 0B6F60D08h, 30296064h, 1100805h, 21E408D8h, 55102020h
dd 2B62B6B5h, 9508E287h, 1FD48660h, 38142420h, 0C2081B2Ch
dd 4CA4278h, 76280C16h, 0CEB0EC9Bh, 14242818h, 3F3E16A3h
dd 6051750Bh, 1C307524h, 0EF51828h, 38B7185Bh, 70F07520h
dd 1B161C03h, 0F8051786h, 22556430h, 0D49C30A4h, 17333B78h
dd 0A48065h, 5D1C2C0Ch, 405BBB94h, 304018A8h, 0E7911D16h
dd 688C3B65h, 11242Ch, 6C05FB28h, 3F285C7Bh, 2C303407h
```

Figure 7: Decompiled MelOn program after packing the program with "Alternate EXE Packer".

However, software-based DRM solutions have inherent limitations since program codes themselves can always be reviewed and then compromised in practice, before or during runtime. Without strong protection for DRM agents, DRM techniques only offer a marginal security improvement. Therefore, we need to consider a mechanism for verifying the trustworthiness of DRM agents.

In general, the basic mechanism is as follows: Before running a media player, a service provider remotely verifies the media player integrity and grants access to that running only when it is not compromised. Such an integrity checking system can be used together with some hardware mechanism such as a Trusted Platform Module (TPM) [7] or ARM TrustZone [17]. Those mechanisms provide a means to reliably report the integrity of software and/or platform configurations with protected key storage to build a strong platform integrity verification mechanism by securely protecting the hash of the normal program binary image through a hardware chip embedded in the motherboard.

5. ETHICAL AND LEGAL ISSUES

The main motivation of our experiments is to analyze potential risks of OMA DRM services and to suggest reasonable countermeasures to mitigate such risks. Therefore, we evaluated whether DRM protection can be disabled through a case study of MelOn. The discovered vulnerability has been completely reported to Loen

[4]https://www.hex-rays.com/products/ida/support/download.shtml

[5]There are several different techniques to achieve the same goal. For example, instead of using two nop instructions with 0x90 and 0x90, a sequence of "0x0F and 0x0D" or "0x50 and 0x58" instructions can also be used.

[6]http://www.alternate-tools.com/pages/c_exepacker.php?lang=ENG

Entertainment Inc., the organization that operates the MelOn service. We also reported this vulnerability to the Korea Internet and Security Agency (KISA) running a bug bounty program for applications used in South Korea.

6. CONCLUSION

In this paper, we implemented a proof-of-concept cracking procedure to disable the protection of OMA DRM. We carefully examined DCF files with different constraints and identified their exact locations. With that information, we reverse-engineered the MelOn media player to bypass the integrity check of the rights object in a DCF file and successfully cracked the media player. Our study shows how difficult it is to securely protect digital items with software-based DRM solutions alone since DRM agents themselves can be compromised. Therefore, it is important to deploy appropriate defense mechanisms that would ensure the integrity of DRM agents with roots of trust at a low level (e.g., through hardware mechanisms). Although we currently limited our security analysis to the MelOn service alone, we believe this type of attack can also be applicable to other systems based on OMA DRM. For future work, we are planning to extend our security analysis to other DRM applications.

7. ACKNOWLEDGEMENTS

This work was supported by the NRF grant funded by the Korea government (No. 2014R1A1A1003707). This work was also funded in part by the ICT R&D program (2014-044-072-003 , 'Development of Cyber Quarantine System using SDN Techniques') of MSIP/IITP. This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC support program (IITP-2015-H8501-15-1008) supervised by the IITP.

8. REFERENCES

[1] Open Mobile Alliance. Candidate version 2.0–13. 2004.

[2] Willms Buhse and Jan van der Meer. The open mobile alliance digital rights management [standards in a nutshell]. *Signal Processing Magazine, IEEE*, 24(1):140–143, 2007.

[3] Feng-Cheng Chang, Chiao-Lin Wu, and Hsueh-Ming Hang. A Switchable DRM Structure for Embedded Device. In *Proceedings of the Third International Conference on International Information Hiding and Multimedia Signal Processing*, 2007.

[4] Christian S. Collberg and Clark Thomborson. Watermarking, tamper-proffing, and obfuscation: Tools for software protection. *IEEE Transactions on Software Engineering*, 28(8):735–746, 2002.

[5] Michael N Gagnon, Stephen Taylor, and Anup K Ghosh. Software protection through anti-debugging. *Security & Privacy, IEEE*, 5(3):82–84, 2007.

[6] David Geer. Digital rights technology sparks interoperability concerns. *Computer*, 37(12):20–22, 2004.

[7] Trusted Computing Group. TPM Main Specification Version 1.2 rev. 103. http://www.trustedcomputinggroup.org/resources/tpm_main_specification, 2007.

[8] F. Hartung and F. Ramme. Digital rights management and watermarking of multimedia content for m-commerce applications. *Communications Magazine, IEEE*, 38(11):78–84, 2000.

[9] James Irwin. Digital rights management: The open mobile alliance DRM specifications. *Information Security Technical Report*, 9(4):22–31, 2004.

[10] Ton Kalker. On Interoperability of DRM. In *Proceedings of the ACM Workshop on Digital Rights Management*, 2006.

[11] Kyoung-Joo Lee. The coevolution of it innovation and copyright institutions: The development of the mobile music business in japan and korea. *The Journal of Strategic Information Systems*, 21(3):245–255, 2012.

[12] Nicholas Paul Sheppard and Reihaneh Safavi-Naini. On the operational semantics of rights expression languages. In *Proceedings of the Nineth ACM Workshop on Digital Rights Management*, DRM '09, pages 17–28, New York, NY, USA, 2009. ACM.

[13] Mark Stamp. *Information security: principles and practice*. John Wiley & Sons, 2011.

[14] SR Subramanya and Byung K Yi. Digital rights management. *Potentials, IEEE*, 25(2):31–34, 2006.

[15] Daniel Thull and Roberto Sannino. Performance Considerations for an Embedded Implementation of OMA DRM 2. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 46–51, 2005.

[16] Ruoyu Wang, Yan Shoshitaishvili, Christopher Kruegel, and Giovanni Vigna. Steal This Movie - Automatically Bypassing DRM Protection in Streaming Media Services. In *Proceedings of the 22Nd USENIX Conference on Security*, 2013.

[17] J. Winter. Experimenting with ARM TrustZone – Or: How I Met Friendly Piece of Trusted Hardware. In *Proceedings of Conference on Trust, Security and Privacy in Computing and Communications*, pages 1161–1166, 2012.

APPENDIX

A. APPENDIX OF ABBREVIATIONS

ABBR.	Description
DRM	Digital Rights Management
OMA	Open Mobile Alliance
CI	Content Issuer
RI	Rights Issuer
RO	Rights Objects
REL	Rights Expression Language
DCF	DRM Content Format
MAC	Message Authentication Code
TPM	Trusted Platform Module
KISA	Korea Internet and Security Agency