International Journal of Cyber-Security and Digital Forensics (IJCSDF) 2(1): 56-69
The Society of Digital Information and Wireless Communications, 2013 (ISSN: 2305-0012)

A DRM Scheme Using File Physical Information

Cheng Qu, Yinyan Yu*, Zhi Tang and Xiaoyu Cui
Institute of Computer Science and Technology, Peking University
Beijing 100080, China
E-mail:{qucheng, yuyinyan, tangzhi, cuixiaoyu}@pku.edu.cn

Abstract—A digital file has both the content and physical information, however the latter was not fully made use of in previous digital rights management (DRM) systems. This paper introduces the idea of making use of file physical information to improve the system security and provides a scheme based on this idea to resist the replay attack in DRM systems. In our scheme, compared to commonly used schemes, we remove the dependency on continuous online connection from the client-side to the server-side or the usage of tamper-proof hardware, such as Trusted Platform Module (TPM). The scheme is appropriate for offline digital content usage. Primary experiments demonstrate that our scheme is secure enough to be put into practice use.

Keywords: digital rights management, file physical information, system security, replay attack, tamper-proof hardware, offline digital content usage.

1 INTRODUCTION

Digital Rights Management (DRM) [1] is used to control the distribution and usage of digital contents. New research is continually carried out in its supporting technology [2], such as access control and digital signature. As to a digital file, it has both the content and physical information; however the latter was not fully made use of in previous research. In some scientific research, the location of an object always plays an important role. Liam Smit et al. [3] showed us how the location of a cellphone can be made use of in digital forensic investigations. In this paper, the physical information of a digital file mainly refers to file size and physical level file location, which is measured by the addresses of file storage cells in the stable storage device. The way we make use of file physical information includes recording a file's size, changing a file's size, recording a file's location, changing a file's location, and so on. How we combine these operations depends on what problem we want to solve. In order to give a real case using file physical information to improve the system security, we provide a scheme based on the usage of physical level file location to resist the replay attack in DRM systems.

1.1 Replay Attack

Content usage is usually controlled by licenses. The user rights are expressed in the form of permissions and constraints, which are recorded in licenses. Some licenses, known as stateful licenses, contain state information (i.e., state-based constraints). State information indicates the usable amount of some permissions (e.g., playing twice, printing three times and one hour's total reading time). Permissions constricted by state information are stateful rights.

Although DRM schemes have provided a great deal of work so far, they still suffer from the replay attack. William Shapiro et al. [4] described the replay attack problem in the client that is completely under the control of the user. For example, a user takes a snapshot of the current state information, uses some

* Yinyan Yu is the corresponding author.

stateful rights, and restores the state information from the snapshot, removing all records of content usage after the snapshot. Some tamper-detection could be provided by attaching a Message Authentication Code (MAC) to each data record based on a secret key [4]. However, this mechanism cannot detect the replay attack.

The schemes proposed in [5-6] can resist the replay attack in digital products trade between clients and servers. However, this type of replay attack is a network attack, which is different from the attack we talk in this paper.

Many content providers offer the trial version of products to consumers, who can try out the products for a limited number of times before they buy the full products. In addition, some consumers may want to transfer the possession of their products to others. Some challenging questions could be raised: how to ensure the trial version of a product is used according to the agreed license and consumers cannot replay it or use it forever? How to prevent consumers from continuing using the transferred products by replaying them? It is obvious that resisting the replay attack is vital for implementing a DRM system.

1.2 Related Work

In many Enterprise Rights Management (ERM) schemes, digital content can be used in two ways: online and offline. The former needs a continuous connection from the client-side to the server-side, and the replay attack can be prevented by means of continuous server-side verification. However, the latter is vulnerable to the replay attack, and all ERM schemes do not claim that they can prevent the replay attack [7].

In [7-8], TPM-based DRM solutions were presented to resist the replay attack. Aimin Yu et al. [9] also adopted TPM to resist the replay attack and proposed TBDRM. Furthermore, Ce Meng et al. [10] integrated the latest advances both in TPM and virtualization. A formal analysis of authentication in TPM was presented in [11]. Authors in [12-16] have proposed schemes using smart card, which is valid in resisting many kinds of attacks. Maryam Savari et al. [17] gave a brief introduction of encryption in smart card.

All the schemes discussed above can be divided into two types. The first one needs continuous online connection, and the second one requires tamper-proof hardware to record some important information. Our scheme eliminates dependency of them.

In this paper, we make full use of file physical information to resist the replay attack. The effectiveness of our scheme bases on the change of file size and physical level file location. Our scheme is appropriate for offline digital content usage. Primary experiments demonstrate that our scheme is secure enough to be put into practice use.

Our scheme bases on previous research, such as software tamper resistance and data integrity, but we do not discuss them in detail. [18-22] showed the detail of them.

Absolute perfect DRM scheme does not exist. When designing a DRM system, we should make a balance between security and convenience according to the actual needs of application scenarios.

The replay attack is so easy to carry out that most digital content consumers can master the method. Our scheme is not designed for

hackers; instead, it faces most digital content consumers. Our scheme cannot resist the attack in the form of hard disk clone; however, after judging and weighing operating risk, time cost and rewards of attack, most consumers will cancel the idea to carry out this attack.

Resisting the replay attack is just one case where file physical information can be made use of to improve the system security. The value of file physical information in system security can be further developed in other applications.

1.3 Organization

The following sections will introduce the replay attack resistance scheme in more details. Section 2 elaborates on the technical details. Section 3 introduces the experimentation along with the results. Section 4 discusses the system security, system overhead and user experience of the scheme. Conclusions are drawn in Section 5.

2 OUR SCHEME

2.1 Initial Idea

There exist a variety of ways to carry out the replay attack, but the core idea is the same, that is recovering the state record from the backup copy. Figure 1 shows a simple replay attack model. In this model the state record is in file F, which is stored in a storage device. The attacker makes a backup copy of file F before he uses stateful rights. The state record is changed after stateful rights usage, and we call the updated file F as F* (we just use a new mark to indicate that the state record in it is different from before, and the real name of the file is not changed). Then he recovers the state record by replacing file F* with the backup copy of file F. Thus the replay attack is carried out. We can judge whether file F is fresh (not replaced with an old backup copy) through an extra log file, but the attacker can also backup and replace the log file.

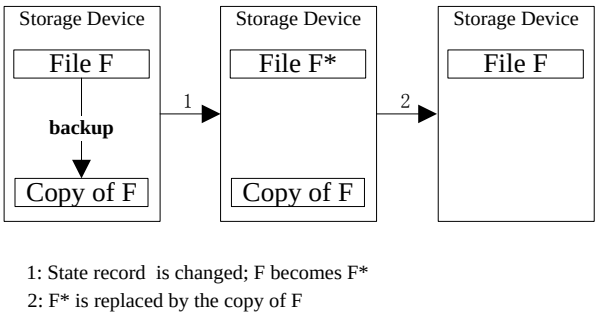
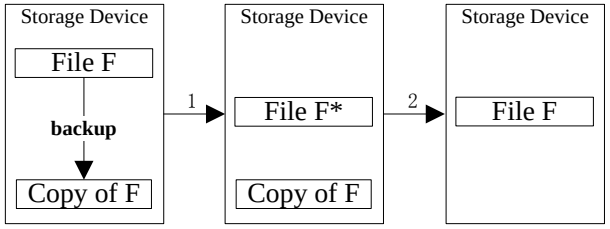


Figure 1. A Simple Replay Attack Model

The replay attack cannot be resisted by just analyzing the content in local files. Therefore tamper-proof hardware and continuous online connection are adopted to resist the replay attack. However, besides the file content, a digital file also has physical information, which was not fully made use of in previous digital rights management (DRM) systems.

Replacing a file with its old backup copy can recover its content, but the physical level file location, controlled by the file system, cannot be set according to one's wish. Our initial idea to resist the replay attack by making use of file physical information is shown in Figure 2. The attacker carries out the attack in the same way as in Figure 1. However, file F will be moved when the state record is changed. Therefore, the location of file F* is different from that of the original file F. When file F* is replaced with the backup copy of file F, the recovered file F does not return to the original location. We can record the mapping relationship between the state record and the original file F's location in an extra log file, thus we can easily discover the mismatch

between the state record and the recovered file F's location. Even if the attacker backups and replaces the log file, the mapping relationship won't change. We can attach a MAC to the log file to proof its integrity.



1: State record is changed; F becomes F*; F* is moved
2: F* is replaced by the copy of F; F does not return to its original location

Figure 2. A Simple Replay Attack Resistance Model

We remove the dependency on continuous online connection from the client-side to the server-side or the usage of tamper-proof hardware, such as Trusted Platform Module (TPM). Therefore, our scheme is appropriate for offline digital content usage. Before offering a whole DRM scheme, we must consider some details: where should we store the state record? How many types of files we should introduce into the DRM system? How to move files when the state record is changed? What if we place the same file (a file like file F shown in Figure 2) in the same location two times? The rest of Section 2 and Section 3 will describe these details.

2.2 Components

There exist two apparent weaknesses in the model shown in Figure 2: the first one is that file F contains the state record, which means file F is meaningful and we cannot change either the content or the size of it at will. The second one is that we only move one single file each time, and there exists a medium probability event that file F comes back to one of its previous location.

In our scheme, the DRM agent mainly manages four types of files (Figure 3), i.e., stateful license file, digital content file, Placeholder File (PF) and Log File (LF). A digital content file is associated with a stateful license file through the content identifier (CID); each stateful license file has an entry in the LF and the license identifier (LID) is recorded in the entry; the one-way hash output of the concatenation of the PFs' locations is also recorded in the LF.

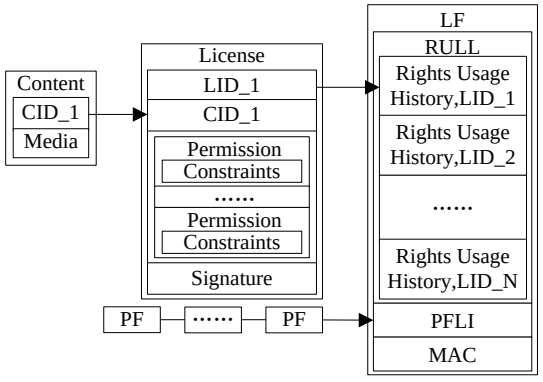


Figure 3. Relationships between Four Types of Files

The stateful license file has a signature, which is computed from all the data in the license except the signature itself, form its issuer. The DRM agent can proof the integrity of a license by verifying the signature. We centralize the management of each license's state information by recording the stateful rights usage history into the LF, so we do not update the license along with the stateful rights usage.

Any kind of digital media may suffer from the replay attack, and our protected object is not confined to some kinds of digital media.

On the contrary, the digital content file can contain any kind of digital media.

The main idea of this paper is making use of file physical information to improve system security, and we introduce the PF to support this main idea. We adopt pure text as the PF's format. But, The PF's format can also be other common file format, such as bitmap. The content of a PF is not what we concern, thus we can change the size of a PF by adding or emptying file content according to our wish. A PF's location is measured by the addresses of the PF's storage cells in the stable storage device. But, recording all the storage cells' addresses will bring large system overhead and is necessary. Usually, we measure a PF's location by the address of the PF's first storage cells or the addresses of the PF's first and last storage cells. The user cannot change the count of PFs. PFs are usually operated in group, and our experimentation proves that three PFs are enough to guarantee the security.

The DRM agent initializes and maintains the LF, which consists of the following parts:

(1) Rights Usage Log List (RULL): RULL is a list containing entries logging the usage history of stateful rights. Each entry is bounded to a stateful license, and the entry contains the corresponding license identifier (LID). The usage history records the accumulative used count or time of stateful rights, and the state information in stateful licenses indicates the usable amount of count or time. If the accumulative used count or time has reached the usable amount, the corresponding stateful rights cannot be used any more. When a user use a license's stateful rights, transfers a license's rights from a device called source device) to another device (called destination device), adds or deletes a digital content file through the DRM agent, the DRM agent will update the list by updating, adding or deleting a corresponding entry.

(2) Placeholder File Location Information (PFLI): PFLI is used to record the one-way hash output of the concatenation of the PFs' locations. The DRM agent can verify whether the LF is fresh (not replaced by an old backup copy) and whether the PFs are legal (not moved by something other than the DRM agent) through the information stored in PFLI, which can detect the replay attack. Whenever a user opens a digital content file, the DRM agent will perform the coherence check, i.e., to check whether the one-way hash output of the concatenation of current PFs' locations is equal to the hash value stored in PFLI. If the two values are different, the DRM agent will prevent the user from the subsequent operations. Whenever a user uses a license's stateful rights, transfers a license's rights from the source device to the destination device, adds or deletes a digital content file through the DRM agent, the DRM agent will firstly do the check mentioned above, and if the check is passed successfully, the DRM agent will move the PFs and replace the hash value stored in PFLI with the one-way hash output of the concatenation of current PFs' locations after the execution of the user's operation.

(3) Message Authentication Code (MAC): This part is the MAC value of the data saved in both RULL and PFLI. Whenever a user opens a digital content file, uses a license's stateful rights, transfers a license's rights from the source device to the destination device, adds or deletes a digital content file through the DRM agent, the DRM agent will verify the integrity

of the LF by checking the coherence between the MAC value recorded in the LF and the MAC value of the current data saved in both RULL and PFLI. When any item in RULL or PFLI is modified by the DRM agent, the MAC value stored in the LF will be updated accordingly.

2.3 Functional Modules

When making use of file physical information to improve system security, we must design some special functional modules to operate on file physical information, and these functional modules should cooperate with traditional security functional modules. In our scheme, six functional modules (Figure 4) are combined to resist the replay attack.

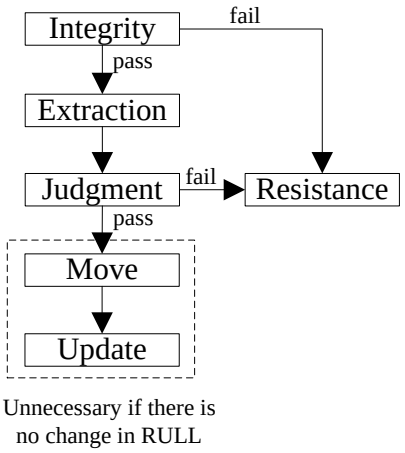


Figure 4. Cooperation between Function Modules

Integrity Module is in charging of proofing the integrity of the LF by checking the MAC in it. In addition, this module is in charge of proofing the integrity of licenses by checking the signatures in them.

Extraction Module extracts the locations of PFs and making a concatenation of them.

Judgment Module judges whether the LF is fresh and whether the PFs are legal. In addition, this module judges whether the right to be used is usable by checking whether its accumulative used count or time has reached the usable amount.

Move Module can move PFs and change their sizes. As to one single PF, we move it and change its size in four steps as follows:

Step 1: Generate a copy of the original PF and give the copy a different name.

Step 2: Change the size of the copy by means of adding or emptying the content in the copy.

Step 3: Delete the original PF.

Step 4: Rename the copy and give it the same name as the original PF.

Update Module can update the RULL by adding, deleting or modifying an entry in this list. What is more, after PFs are moved, this module will update the PFLI. This module will re-compute the MAC value whenever the content in the RULL or the PFLI is updated.

Resistance Module will stop the usage of digital content when any illegal action (e.g., tampering the LF or licenses, bypassing the DRM agent to move PFs) is discovered.

2.4 Solution

The integrity of the licenses and the LF are proofed when they are in use. The LF may be replaced by an old back copy at any moment, if we do not discover the replacement before Update Module updates the LF, we will never discover the replacement, which is replay attack. Therefore, in order to resist the replay attack, before the Update Module updates the LF, we must judge its freshness. In addition, we must judge whether the right to be used is usable.

In our scheme, four kinds of operations will lead to the update on the LF. These operations include adding a stateful license file, deleting a stateful license file, using a license's stateful rights and transferring a license from the source device to the destination device.

When a user opens a digital content file, Integrity Module will proof the integrity of both the LF and the file's license; Judgment Module will judge whether the LF is fresh and whether the PFs are legal. Opening a digital content file will not change the content in RULL, so Move Module and Update Module will do nothing in this situation.

When a user adds/deletes a digital content file, Integrity Module will only proof the integrity of the LF; Judgment Module will judge whether the LF is fresh and whether the PFs are legal. After downloading/removing the digital content file's license, Move Module will move PFs and change their sizes. At last, Update Module will add/delete the corresponding entry in RULL, record the one-way hash output of the concatenation of the PFs' locations into PFLI and re-compute the MAC value of the data saved in both RULL and PFLI.

When a user transfers a license from a source device to a destination device, the process control involves source-side control, destination-side control, the protocol between source-side and server-side, and the protocol between destination-side and server-side. The source-side control is as same as the process control that a user deletes a digital content file, and the destination-side control is as same as the process control that a user adds a digital content file. As to the protocol between source-side and server-side, the server-side extracts the original license's stateful rights' usable amount minus the accumulative used amount from the source-side. As to the protocol between destination-side and server-side, the server-side generates a new license containing the remaining stateful rights' usable amount, and transfers the license to the destination-side.

We centralize the management of every license's state information by means of recording the consumption history of stateful rights into the RULL. Therefore, when a user uses a license's stateful rights, the license will not be changed. Instead, Update Module will modify the corresponding entry in the RULL. The operation of using a license's stateful rights is more common than the other four kinds of operations, and we describe the cooperation between function modules shown in Figure 4 in detail as follows:

Step 1: Integrity Module proofs the integrity of the LF by checking the coherence between the MAC value recorded in the LF and the MAC value of the current data saved in both RULL and PFLI. If the integrity is verified, go to Step2, else stop.

Step 2: Integrity Module proofs the integrity of the license by verifying the signature. If the integrity is verified, go to Step3, else stop.

Step 3: Extraction Module extracts the locations of current PFs.

Step 4: Judgment Module judges whether the LF is fresh and whether the PFs are legal by checking whether the one-way hash output of the concatenation of current PFs' locations

is equal to the hash value stored in PFLI. If the two values are equal, go to Step 5, else stop.

Step 5: Judgment Module locates the corresponding entry in RULL according to the LID of the license mentioned in Step 2, and then judges whether the right to be used is still usable. If the right's accumulative used count or time recorded in the entry has reached the usable amount recorded in the license, the right cannot be used any more, and else the right is still usable. If the right is still usable, go to Step 6, else stop.

Step 6: The right is used. Meanwhile, Move Module moves the PFs and changes their sizes one by one.

Step 7: Update Module modifies the entry mentioned in Step 5 by updating the value of the right's accumulative used count or time.

Step 8: Update Module replaces the hash value stored in PFLI with the one-way hash output of the concatenation of current PFs' locations.

Step 9: Update Module computes the MAC value of the current data saved in both RULL and PFLI, and then replaces the old MAC value stored in the LF with the new one.

2.5 Security Analysis

In our scheme, there is a medium probability event that a PF comes back to one of its previous location, and we call this event as "regress". Figure 5 shows an example of regress: a PF's first storage cell and last storage cell separately located in A and B at a certain moment in the past, and a regress means that the PF's first storage cell and last storage cell separately comes back to A and B at the same time. This is a potential threat to our scheme.

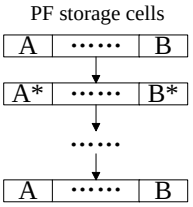


Figure 5. An Example of Regress

If the DRM agent only uses one PF, the scheme is still vulnerable to replay attacks. When the regress occurs in the PF, an attacker can replace the current LF with an old backup copy with the same content in PFLI to carry out the replay attack. Unfortunately, Judgment Module cannot discover the replay attack on this occasion.

Compared to the regress, there is a small probability event that the current content in PFLI is same to a previous one. We call this event as "repetition". Let us consider the situation that the count of PFs is two. PF_1's location was C and PF_2's location was D at a certain moment in the past, and a repetition means that PF_1 comes back to C and PF_2 comes back to D at the same time. Figure 6 shows an example of repetition. When a repetition occurs, regresses must occur to all of the PFs. However, regresses occurring to all of the PFs will not surly lead to a repetition.

When a repetition occurs, an attacker can replace the current LF with an old backup copy, which contains the PFLI as same as the current one, to carry out the replay attack. Judgment Module cannot discover the replay attack on this occasion.

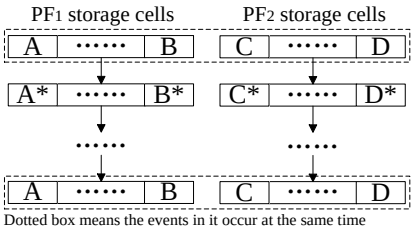


Figure 6. An Example of Repetition

A PF's location is controlled by the file system, and there exists big difference between different file systems. We can hardly make a quantitative analysis to the probability of the repetition in a file system, but, in theory, we can make the probability of the repetition keep zero within a certain amount of PF moves.

Let us consider the situation that the count of PFs is three. We assume that all the initial sizes of PFs are zero and the PF size changing pace is as same as the size of a file storage cell, thus the count of storage cells occupied by a PF can be increased during a PF move. We choose three different prime numbers C_1, C_2 and C_3 as the size change cycles of PF_1, PF_2 and PF_3, thus the PFs size change cycle is $C_1 \times C_2 \times C_3$.

A PF is a small file and its storage cells have consecutive numbers, so a repetition means not only the locations return but also the storage cell counts return. When a repetition occurs, we assume that the current counts of storage cells occupied by the three PFs are S_1, S_2 and S_3. In the last PFs size change cycle, there was only one time that the counts of storage cells occupied by the three PFs were S_1, S_2 and S_3 concurrently. When the storage cell counts return occurs, the locations return may not occur at the same time. Therefore, in the first PFs size change cycle, the probability of the repetition is zero, and in the latter cycles, the probability is very low. We can extend the PFs size change cycle by increasing the count of PFs and choosing larger prime numbers as the PF size change cycles. The strategy mentioned above does not rely on specific file systems.

3 EXPERIMENTATION AND RESULTS

3.1 Verification of Theory

In theory, in the first PFs size change cycle, the probability of the repetition is zero. This conclusion does not rely on specific file systems, and we experimented with tens of groups of PFs to verify this conclusion.

All the experiments were performed in a platform, where the CPU main frequency is 2.50GHz, with Windows OS. In the experiment platform, the hard disk sector cluster is the file storage cell, and the size of a sector cluster is 4KB. A PF's location was measured by its first and last sector cluster numbers. We changed a PF's size by adding and emptying the file content cyclically.

In each experiment, all the initial sizes of PFs were set as zero and all the PF size changing pace were set as 4KB. We chose n different prime numbers C_1, C_2, …, C_n as the size change cycles of PF_1, PF_2,…, PF_n when the count of PFs is n. We recorded the count of repetitions as CRP, and computed the repetition rate as the following equation:

$$\text{Repetition Rate} = \text{CRP} / C_1 \times C_2 \times \ldots \times C_n \quad (1)$$

We moved them $C_1 \times C_2 \times \ldots \times C_n$ times for each group of PFs, and all the repetition rates were zero, which proved the conclusion that the probability of the repetition is zero in the first PFs size change cycle.

A repetition means both the locations return and the storage cell counts return. The conclusion mentioned above bases on the fact that the storage cell counts return cannot occur in the first PFs size change cycle. We did further experimentation to the study the probability of the locations return.

In previous experiments, a PF's location was measured by its first and last sector cluster numbers. However, in next experiments, a PF's location was measured only by its first sector cluster number, thus the storage cell counts return can be ignored. In other words, when judging whether a regress or a repetition occurs, we only check whether the PF's first storage cell comes back to one of its previous location. When regresses result in the current content in PFLI becoming as same as a previous one, a repetition occurs.

We mainly studied five factors which may make a difference to the probability of the locations return. Factor (1) is the count of PFs; factor (2) is the PF size change pace; factor (3) is the PF size change cycle; factor (4) is the sequence of moving PFs; factor (5) is the size of free space in the stable storage device.

3.2 Factor Analysis

3.2.1 Count of PFs

When we studied how the count of PFs influence the probability of the locations return, we made five groups of PFs. Group 1 had 1 PF; group 2 had 2 PFs; group 3 had 3 PFs; group 4 had 4 PFs; group 5 had 5 PFs. We did one experiment with each of these five groups of PFs. All the PF size change paces were set as 4KB, and all the PF size change cycles were set as 128. Thus the PFs size change cycle was only 128. In each experiment, the initial size of each PF was set as zero, and we moved a group of PFs 50,000 times in a hard disk partition with 50GB free space. We recorded the count of repetitions as CRP, and computed the repetition rate as the following equation:

$$\text{Repetition Rate} = \text{CRP}/50{,}000 \quad (2)$$

The statistics (Figure 7) show that a larger count of PFs leads to a lower repetition rate. As to the count of PFs, the repetition rate has dropped to a very low level (0.03%) when the count of PFs reaches three.

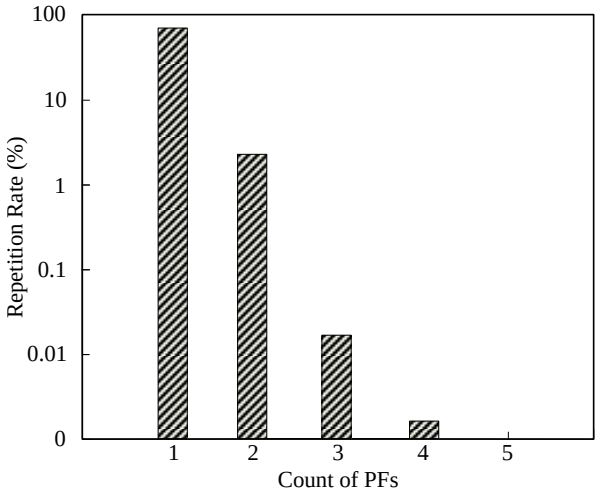


Figure 7. Repetition Rate with Different Count of PFs

3.2.2 PF size change pace

When we studied how the PF size change pace influences the probability of the regress, we speeded up the size change pace of a PF step by step. The conclusion is: in order to reduce the probability of the regress, the pace should be quick enough to change the count of the sector clusters occupied by the PF within several moves, under which precondition a quicker pace does not reduce the probability further. However, with the same PF size change cycle, a quicker pace means a larger

average size, which will cost more time to move the PF.

3.2.3 PF size change cycle

When we studied how the PF size change cycle influences the probability of the regress, we enlarged the size change cycle of a PF step by step. The conclusion is: compared to a small PF size change cycle, a larger one does reduce the probability of the regress. However, with the same PF size change pace, a larger PF size change cycle means a larger average size, which will cost more time to move the PF.

3.2.4 Sequence of Moving PFs

When we studied how the sequence of moving PFs influences the probability of the locations return, we made three groups of PFs. Group 1 had 1 PF; group 2 had 2 PFs; group 3 had 3 PFs. We did one experiment with each of these three groups of PFs. All the PF size change paces were set as 4KB, and all the PF size change cycles were set as 128. Thus the PFs size change cycle was only 128. In each experiment, the initial size of each PF was set as zero. Each experiment was divided into two parts. In the first part, the sequence of moving PFs is fixed; more specifically, we numbered the PFs and moved them in an ascending sequence. In the second part, the sequence of moving PFs is not fixed; more specifically, we numbered the PFs and moved them in varying sequences. For each part of an experiment, we moved PFs 50,000 times in a hard disk partition with 50GB free space and computed the repetition rate as Equation (2).

The statistics (Figure 8) show that whether moving PFs in a fixed sequence or in varying sequences almost makes no difference to the probability of the repetition.

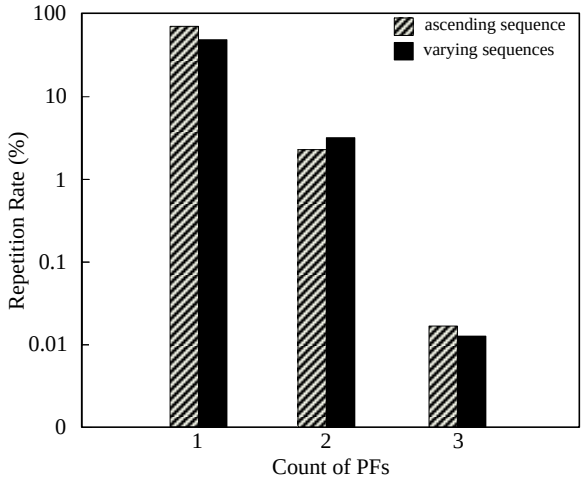


Figure 8. Repetition Rate with Different Sequence of Moving PFs

3.2.5 Free Space

When we studied how the free space influences the probability of the locations return, we changed the free space of the hard disk partition, where the PFs are stored, from 50GB to 5MB. All the PF size change paces were set as 4KB, and all the PF size change cycles were set as 128. Thus the PFs size change cycle was only 128. In each experiment, the initial size of each PF was set as zero, and the count of PFs was three. We moved PFs 50,000 times in each experiment and computed the repetition rate as Equation (2).

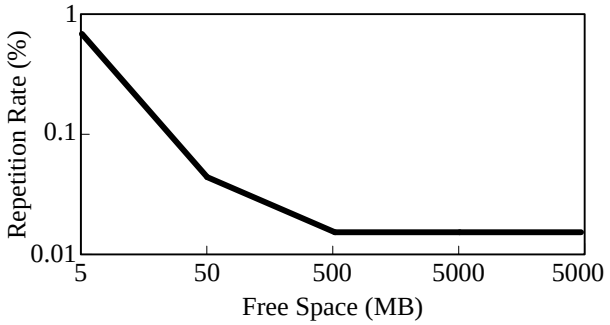


Figure 9. Repetition Rate with Different Free Space

The statistics (Figure 9) show that the repetition rate is a little high (0.92%) when the

free space has only 5MB, but when the free space increases to 50 MB the repetition rate becomes quite low (0.07%). The repetition rate keeps on a very low level (0.03%) when the free space is more than 500MB.

3.3 Summary

In summary, we can measure a PF's location by its first and last storage cell addresses and choose different prime numbers as the size change cycles of PFs, thus the repetition rate is zero in the first PFs size change cycle. In latter cycles, the repetition rate is still very low because the probability of the locations return is small. We can use more PFs or set PF size change cycles with larger prime numbers to extend the PFs size change cycle.

4 DISCUSSIONS

4.1 System Security

The tamper behavior on the LF or licenses can be discovered by Integrity Module. Judgment Module can judge whether the PFs are legal and the freshness of LF. Move Module is in charge of moving PFs. A file's location is decided by the file system, so an attacker cannot move a PF to an expected location.

An attacker cannot carry out the replay attack until a repetition occurs. In theory, the repetition rate is zero in the first PFs size change cycle, and the experiments verify this conclusion. The repetition is still very low in latter cycles. What's more, an attacker may not know when a repetition occurs because the data in PFLI is hash value.

If the replay attack fails, all the digital contents will become unusable. Even through an attacker carries out the replay attack successfully with an old backup copy of the LF; the RULL in the copy will not contain entries for the recently added digital content files, so the attacker cannot use them any longer.

Our scheme cannot resist the attack in the form of hard disk clone. However, besides operating risk and great time cost, this type of attack will remove all the data added or updated since the cloning operation.

All DRM systems are designed for common users, not for hackers. If an attacker can fully control the file system, our scheme is also invalid. However, this type of attack is too difficult for the vast majority of users to carry out, and after judging and weighing operating risk, time cost and rewards of attack, most consumers will cancel the idea to carry out this attack.

4.2 System Overhead

The system overhead mainly concentrates on two aspects: The first one is some hash transformations and encryption/decryption; the second one is moving and changing PFs. Many DRM systems have proven that the first one is efficient. As to the second one, in our experiment platform, where the CPU main frequency is 2.50GHz, moving a PF within 512KB and changing its size cost less than 25 milliseconds. What's more, PFs are moved only before Update Module updates the LF, where the frequency is not high. A user may not be aware of the system overhead caused by the operation on PFs.

4.3 User Experience

Our scheme removes the dependency on continuous online connection from the client-side to the server-side or the usage of tamper-proof hardware, so it can be applied to

common personal computers. The offline digital content usage and the small system overhead will not bring inconvenience to user experience.

One limitation is that the user cannot defragment the hard disk partition where the PFs are stored. If the user defragments this partition, Judgment Module will judge that the PFs are moved illegally. As a result, the user cannot use the digital contents any longer. Although defragment on a hard disk is a low-frequency event, measures should be taken to solve this problem. The license issuer can offer the service to recover a user's rights. The server-side should offer data backup service and data recovery service.

5 CONCLUSIONS

In this paper, we introduce the idea of making use of file physical information to improve the system security. The way we make use of file physical information includes recording a file's size, changing a file's size, recording a file's location, changing a file's location, and so on. In order to give a real case to describe how to combine these operations, we provide a scheme based on our core idea to resist the replay attack in DRM systems. The theory analysis and experiments show that the proposed scheme is secure enough to be put into practice use. In addition, the scheme only brings little system overhead and does not bring inconvenience to user experience.

There is still some further work we can do in the future. We can find more application scenarios to make use of file physical information to improve the system security.

ACKNOWLEDGMENT

This work was supported by Major Scientific and Technological Project of GAPP in China (No. GXTC-CZ-1015004).

REFERENCES

1. Rosenblatt, W., Trippe, W., Mooney, S.: Digital Rights Management: Business and Technology. M&T Books, New York (2002).
2. Evered, M.: A Formal Semantic Model for the Access Specification Language RASP. International Journal of Cyber-Security and Digital Forensics (IJCSDF), vol. 1, no. 2, pp.152-159. SDIWC, Hong Kong (2012).
3. Smit, L., Stander, A., Ophoff, J.: An Analysis of Base Station Location Accuracy within Mobile-Cellular Networks. International Journal of Cyber-Security and Digital Forensics (IJCSDF), vol. 1, no. 4, pp.272-279. SDIWC, Hong Kong (2012).
4. Shapiro, W., Vingralek, R.: How to Manage Persistent State in DRM Systems. Security and Privacy in Digital Rights Management , vol. 2320, pp.176-191. Springer, Heidelberg (2002).
5. Yan, J., Peng, X.: Security Strategy of DRM Based on Trusted Computing. Journal of Computational Information Systems, vol. 7, no. 9, pp. 3226-3234, Binary Information Press, USA (2011).
6. Li, F., Yu, Y., Li, D., Zhang, J.: Research on Access Control Mechanism for Digital Products Trading. In: Proc. 7th International Conference on Computational Intelligence and Security, pp. 1041-1045, IEEE Press, New York (2011).
7. Abbadi, I.M., Alawneh, M.: Replay Attack of Dynamic Rights within an Authorised Domain. In: Proc. 3rd International Conference on Emerging Security Information, Systems and Technologies, pp. 148-154, IEEE Press, New York (2009).
8. Sadeghi, A.R., Scheibel, M., Stüble, C., Wolf, M.: Play It Once Again, Sam-Enforcing Stateful

Licenses on Open Platforms. 2nd Workshop on Advances in Trusted Computing (2006).

9. Yu, A., Feng, D., Liu, R.: TBDRM: A TPM-Based Secure DRM Architecture. In: Proc. 2009 International Conference on Computational Science and Engineering, pp.671-677, IEEE Press, New York (2009).
10. Meng, C., He, Y., Zhang, Q.: A Dependable, Scalable Maintenance Scheme for Stateful License in DRM. In: Proc. 2009 International Symposium on Information Engineering and Electronic Commerce, pp.754-758, IEEE Press, New York (2009).
11. Delaune, S., Kremer, S., Ryan, M.D., Steel, G.: A Formal Analysis of Authentication in the TPM. Formal Aspects in Security and Trust, vol. 6561, pp. 111-125. Springer, Heidelberg (2010)..
12. Yang, C., Jiang, Z., Yang, J.: Novel Access Control Scheme with User Authentication Using Smart Cards. In: Proc. 3rd International Joint Conference on Computa- tional Science and Optimization, pp.387-389, IEEE Press, New York (2010).
13. Elmadani, A.B.: Trusted Document Signing Based on Use of Biometric (Face) Keys. International Journal of Cyber-Security and Digital Forensics (IJCSDF), vol. 1, no. 4, pp.289-296. SDIWC, Hong Kong (2012).
14. Jung, H., Kim, H.S.: Secure Hash-based Password Authentication Protocol Using Smartcards. Computa- tional Science and Its Applications-ICCSA 2011, vol. 6786, pp.593-606. Springer, Heidelberg (2011).
15. Hao, Z., Yu, G.: A Security Enhanced Remote Password Authentication Scheme Using Smart Card. In: Proc. 2nd International Symposium on Data, Privacy, and E-Commerce, pp.56-60, IEEE Press, New York (2010).
16. Sood, S.K., Sarje, A.K., Singh, K.: A Secure Dynamic Identity Based Authentication Protocol for Multi-server Architecture. Journal of Network and Computer Applications, vol. 34, no. 2, pp.609-618. Elsevier (2011).
17. Savari, M., Montazerolzhour, M.: All about Encryption in Smart Card. In: Proc. 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic, pp.54-59, IEEE Press, New York (2012).
18. Ghosh, S., Hiser, J.D., Davidson, J.W.: A Secure and Robust Approach to Software Tamper Resistance. Information Hiding, vol. 6387, pp.33-47. Springer, Heidelberg (2010).
19. Byun, J.W., Sohn, Y., Bertino, E.: Systematic Control and Management of Data Integrity. In: Proc. 11th ACM Symposium on Access Control Models and Technologies, pp.101-110, ACM Press, New York (2006).
20. Blietz, B., Tyagi, A.: Software Tamper Resistance through Dynamic Program Monitoring. Digital Rights Management, Technologies, Issues, Challenges and Systems, vol. 3919, pp.146-163. Springer, Heidelberg (2006).
21. Ghaeb, J.A., Smadi, M.A., Chebil, J.: A High Performan- ce Data Integrity Assurance Based on the Determinant Technique. Journal of Future Generation Computer Systems, vol. 27, no. 5, pp.614-619. Elsevier (2011).
22. Jin, P., Myles, P., Lotspiech, P.: Key Evolution-based Tamper Resistance: A Subgroup Extension. In: Proc. 2nd ACM Symposium on Information, Computer and Communications Security, pp.321-328, ACM Press, New York (2007).