The Design and Implementation of a Multimedia Data Management and Monitoring System for Digital Rights Protection Using Shared Key Pool

Jae-Pyo Park, Moon-Seog Jun

School of Computing, Soongsil University, Korea

Summary
In this paper we first propose an encryption scheme for encryption of MPEG video data using PKI, second, we propose a shared key pool and encryption/decryption of multimedia data. Third, to reduce huge play-out delay time which occurs when performing decryption on large capacity multimedia data, a double buffer configuration is adopted and a real-time decryption scheme using efficient buffer scheduling is proposed. After designing and implementing the proposed system, tests were then performed using video data files of various sizes for performance evaluation. We verified that the proposed system significantly reduces delay time, including decryption time, when playing back video data files in the client system compared with existing systems.
Key words:
DRM, PKI, Shared key pool, Licensing agent, Double buffer

1. Introduction

Recently, various researches are being performed to protect digital contents from intellectual property infringement. Further studies are concentrating on the implementation of comprehensive measures to manage the distribution of such contents using DRM (Digital Rights Managements) technology so that production, distribution and use of contents can be performed within a trusted environment[1, 2]. Existing DRM implementations do not take privacy protection into consideration for the reason that user privacy protection is not directly necessary for copyright protection. Therefore, user information leaked during user authentication for license issuing, and usage details reporting the process of monitoring against illegal content usage has caused user privacy infringements [3, 4].

Methods for implementing security for digital contents can be divided in two categories: upper-level security and lower-level security [5]. Upper-level security schemes relate to user authentication, while lower-level security relates to the protection of the data itself. User authentication means that an authenticated user is not subject to limitations in content usage [6]. Therefore, a function used for monitoring the amount of content usage in order to maintain information on the number of users accessing a specific multimedia content is required [7]. However, since in this scheme an authenticated user can obtain illegitimate copies of data that can be distributed, it cannot provide perfect protection of distributed data. As such, protection in the data itself performs encryption on the content to restrict user's access to the content. Therefore, in DRM, security is implemented by performing encryption on the content data itself.

The algorithms used for encryption are the private key algorithm and public key algorithm. The secret key algorithm is an algorithm which performs encryption at high speed by using a single key for encryption. However, in this method, there is the problem of key distribution; that is, the sender and recipient have to exchange their private keys beforehand. In addition, if large capacity moving images are transferred, a large amount of processing time is required if a transfer is carried out simultaneously with encryption. The public key algorithm offers the advantage of using separate keys for encryption and decryption so that the sender and recipient can safely exchange keys. However, its drawback is slow execution speed. Therefore, when encrypting large capacity moving image files, encryption on data using a secret key must be performed first, then the secret key is encrypted using the public key.

In this paper, we propose a shared key pool scheme which encrypts the secret key using each user's private information in order to prevent exposure of the secret key by the user while performing authentication for digital content users. This proposed scheme prevents the exposure of the secret key by the user while encrypting contents beforehand to improve transfer speed. In addition, by using a licensing agent, a license is downloaded from the licensing server which manages the licenses within a database when executing content so that offline execution is possible. For security of the transmitted key pool information, the key pool information is encrypted using a secret key and the relevant secret key is encrypted using the user's public key in order to improve the encryption speed and security of the encrypted key pool which is being transmitted.

Since the agent needs large amounts of time for performing decryption of high capacity moving image data, we propose a comprehensive DRM system which implements a buffer control for seamless replay of moving images, enables real-time decryption by the user through efficient buffer scheduling, and prevents illegal execution of contents through online and offline-based user authentication on multimedia contents and encryption of the source data itself.

The rest of this paper organized as follows. Section 2 describes related work for DRM system. In Section 3, we propose an integrated DRM System using shared key pool. Section 4 presents implementation of system. Performance evaluation is explained in section 5, followed by conclusion in section 6.

2. Related Works

Using DRM technology, international companies such as InterTrust and Microsoft, as well as Korean companies such as Digicap, offer various types of DRM solutions [8]. However, since existing DRM solutions perform encryption using secret keys when the user downloads files, a large amount of encryption time is required. In addition, decryption must be performed first for large capacity contents so that the user cannot perform the playback of the file in real-time. Besides, if the key used for encryption and decryption is exposed by the user, the copyrighted content can no longer be protected.

Existing DRM solutions perform static copyright management by inserting information such as data protection conditions or copyright management into moving image data. As such, dynamic copyright control is difficult and data needed to prove illegal activities in case of copyright infringement is difficult to obtain. Therefore, existing DRM solutions use software agents to monitor a user's data usage in order to solve this problem, but this solution is subject to the functional constraints present in off-line usage environments. Hence, a digital copyright management technology which is applicable to all types of contents in both online and offline environments, and is capable of dynamic copyright management and real-time management and tracing, should be developed.

Microsoft's WMRM (Windows Media Rights Manager) is an end-to-end DRM system which provides safe distribution of digital media files to content providers and consumers [9, 10]. WMRM delivers media such as music, video, etc., that is protected in encrypted file format, to content providers through the Internet. In WMRM, each server or client instance receives a key pair through an individualization process, and instances that are determined to be cracked or unsafe are excluded from service through the certificate cancellation list. WMRM is widely used in embedded form with the Windows Media Player, but it shows limited adaptability to dynamic environment changes, is only applicable to the Windows Media Player, and only supports a limited range of file types. In addition, it has the disadvantage of potential leakage of user information such as user ID or e-mail address since no particular protection technology is applied on the certification stage for issuing licenses.

3. Integrated DRM System Using Shared Key Pool

- ### 3.1 System Architecture

For data protection and authentication of original content, data should be protected not only by simple access restriction or password authentication, but also through user authentication and data encryption implemented by PKI technology. The proposed system is a client/server configuration and its overall layout is illustrated in Figure 1.

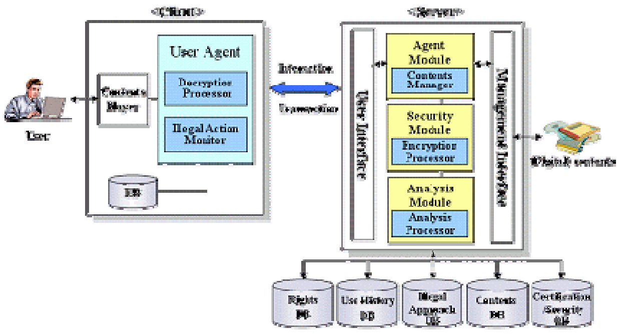

Fig. 1 System Architecture

When content is registered on the system server through the external interface, content monitoring processing is performed by the agent module and an encryption is placed on the content. When the user accesses the content, user authentication is performed by the licensing agent that is dispatched by the server. If the user is authenticated, the content is executed by an application program; otherwise, a warning message is displayed. Monitoring against illegal usage is performed on the content by the licensing agent, and all illegal user activities are stored on the server interface through the monitoring interface. Even in the case of authenticated users, content is protected by encryption of the content itself to restrict content usage according to access privilege levels.

3.2 Encryption and Decryption of Video Data

The content's author sends the generated content to the server. The server then encrypts that content using an arbitrary secret key (Ks) and stores the encrypted content

C together with the secret key (Ks) on the server's content database.

$$C = EKs[data] \tag{1}$$

The user can download a desired content from the server through the authentication process or copy it from another user. However, downloaded content is encrypted and therefore has to be executed through the agent. The server generates a shared key pool for encrypting the secret keys in order to prevent leakage of the secret key through the user. As shown in Figure 2, the server encrypts the I-frame of the content's GOP using a secret key through either AES or SEED algorithm and stores that content's ID and secret key on the server's database. An arbitrary shared key pool applicable in the encrypted content is then generated and is also stored on the database. If a user is registered, the server performs user authentication using the user's certificate, and then extracts the user's information from the private key pool using the user's certificate in order to generate the user's key.

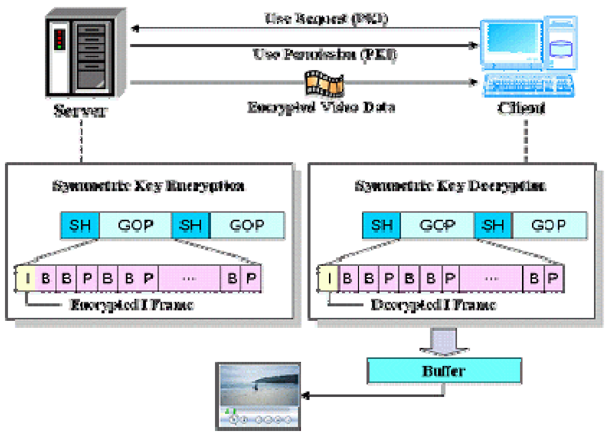

Fig. 2. Encryption & Decryption Processing

In addition, an encrypted shared key pool is generated by performing a bitwise XOR operation on each bit column of the secret key (Ks) and the shared key pool. Private information and the encrypted shared key pool are stored on the server's database and also on the user's database using the user agent.

3.2.1 Configuration of the Shared Key Pool

To generate a shared key pool, the content producer can encrypt the content to be distributed using a secret key (Ks) and divide it into k bit columns as in Equation 2.

$$Ks = Ks_1 \mid Ks_2 \mid \dots \mid Ks_k \tag{2}$$

Generally, in secret key encryption, a secret key (Ks) with a length of 128 bits is used. The shared key pool consists of $k * 2^{\frac{n}{k}}$ bits and is generated according to Equation 3.

$$\left\{ a_1^0, a_1^1, a_1^2, \dots, a_1^{2^{\frac{n}{k}-1}}, a_2^0, a_2^1, a_2^2, \dots, a_2^{2^{\frac{n}{k}-1}}, \dots, a_k^0, a_k^1, a_k^2, \dots, a_k^{2^{\frac{n}{k}-1}}, \right\} \tag{3}$$

In order to adapt to the key's size, an array with k rows and $2^{\frac{n}{k}}$ columns can be expressed as in Table 1.

a_1^0	a_1^0	…	$a_2^{2^{\frac{n}{k}-1}}$
a_1^0	a_1^0	…	$a_2^{2^{\frac{n}{k}-1}}$
:	:	…	:
a_1^0	a_1^0	…	$a_2^{2^{\frac{n}{k}-1}}$

Table 1. Shared Key Pool of Ks

The private key of each user, Kp, is a set of bit columns consisting of k bits as in Equation 4.

$$Kp = a_1^{b_1} \mid a_2^{b_2} \mid \cdots \mid a_k^{b_k} \tag{4}$$

Here, b_i corresponds to the value of each i th row of the key pool, as shown in Equation 5, and it is an important value determining each user's private key. b_i is extracted from the public key of the user certificate.

$$B = b_1 \mid b_2 \mid \dots\dots \mid b_k \tag{5}$$

The length of the user's public key is 512 bits if required secrecy is low. For critical information requiring high secrecy, a key length of 1024 bits is used. In general, if a public key of n bits length is used, the value of each item within the key pool falls in the range of $2^{\frac{n}{k}}(0 \sim 2^{\frac{n}{k}} - 1)$. For example, if the public key's length is 512 bits while the private key's length is 128 bits, 512/128=4; therefore, each item is 24=16 which corresponds to a value range of 0-F in hex. Therefore, the individual rows of the actual private key are determined by the public key's value which is in the range of 0-F. The range of each key's value according to the length of the secret key and the length of the public key is summarized in Table 2.

Size of Secret Key / Size of Public Key	128	256	512
512	0 ~ F	0 ~ 3	0, 1
1024	0 ~ 255	0 ~ F	0 ~ 3
2048	0 ~ 65535	0 ~ 255	0 ~ F

Table 2. Value of Key Pool

Therefore, since each user's private key is determined by each user's unique public key, the key value selected from each row by the public key guarantees that each user is assigned a different key. When the shared key pool is generated, a bitwise XOR is executed for encryption of the secrete key (Ks) on $a_i^0, a_i^1, ..., a_i^{2^{\frac{n}{k}-1}}$ ($2^{\frac{n}{k}}$ bits) with each bit of Ks_i for each i th row of the shared key pool, as shown in Equation 6.

$$a'^0_i = Ks_i \oplus a_i^0, \quad a'^1_i = Ks_i \oplus a_i^1, \quad \ldots\ldots\ldots.,$$
$$a'^F_i = Ks_i \oplus a_i^F \qquad (6)$$

The shared key pool in an encrypted form can be obtained through the calculation of Equation 6 as shown in Table 3.

Table 3. Encrypted Shared-key Pool by Ks

a'^0_1	a'^0_1	$\ldots$	$a'^{2^{\frac{n}{k}-1}}_2$
a'^0_1	a'^0_1	$\ldots$	$a'^{2^{\frac{n}{k}-1}}_2$
:	:	$\ldots$	:
a'^0_1	a'^0_1	$\ldots$	$a'^{2^{\frac{n}{k}-1}}_2$

Table 3. Encrypted Shared-key Pool by Ks

Then, the encrypted shared key pool is forwarded to the user agent through the network. The agent finds the secret key (Ks) using the user's public key (Kp) from the encrypted key pool in order to decrypt the encrypted content. The moving image content file is decrypted using this secret key (Ks) and then displayed to the user. How the agent finds the secret key (Ks) by using the encrypted key pool and the private key (Kp) is summarized in Equation 7.

$$\text{Since } Kp = a_1^{b_1} \mid a_2^{b_2} \mid ... \mid a_k^{b_k} \text{ and } a'^j_i = Ks_i \oplus a_i^j;$$
$$a_1^{b_1} \oplus a'^{b_1}_1 = a_1^{b_1} \oplus Ks_i \oplus a_1^{b_1} = Ks_i \qquad (7)$$

For example if n=4, k=4, $b = 2^{\frac{4}{4}} = 2^1 = 2$, which has the value ranging between 0 and 1. The key pool is shown in Table 4.

	0	1
a_1	1	1
a_2	0	1
a_3	0	0
a_4	1	1

Table 4. Secret Key Values

If b=0,0,1,0, the user's private key is { $a_1^0, a_2^0, a_3^1, a_4^0$ } = {1,0,0,1}. If the secret key Ks={1,1,1,0}, the encrypted key pool can be obtained as shown in Equation 8, through a bitwise XOR with the secret key Ks_i. The encrypted key pool is shown in Table 5.

$$\{1, 1, 1, 0\} \oplus \{1, 0, 0, 1\} = \{0, 1, 1, 1\}$$
$$\{1, 1, 1, 0\} \oplus \{1, 1, 0, 1\} = \{0, 0, 1, 1\} \qquad (8)$$

0	0
1	0
1	1
1	1

Table 5. Shared-Key Pool

The user agent can be obtained using the encrypted key pool and the private key (Ks) as in Equation 9.

$$\{1, 0, 0, 1\} \oplus \{0, 1, 1, 1\} = \{1, 1, 1, 0\} \qquad (9)$$

3.2.2 License Certification Method

(1) User Authentication Protocol

In order to use copyrighted content, user authentication is required as illustrated in Figure 3. User authentication is carried out through member enrollment, and members can log in to download files. However, users who have received moving image data from other users can also redistribute contents, and therefore, anyone can join and login through PKI-based certificates. While a separate login scheme using ID and password is also supported, even in this process, the PKI-based certificate is always verified for login. If authentication is made through ID/password- or certificate-based login processes, moving images can be downloaded.

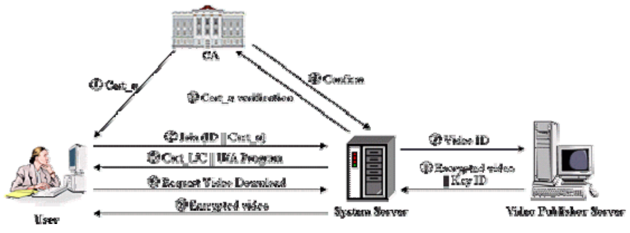

Fig. 3. User Verification Protocol

The user connects to the system server and transmits his certificate Cert_u. The system server verifies the user's certificate Cert_u through the authentication path in the CA server. If the certificate is correct, the user agent program and the server's certificate are transmitted to notify the user that authentication has been completed successfully. However, the moving image data is encrypted, and therefore cannot be executed directly after a download. The agent downloaded during the authentication process must be installed so that the execution of the moving image content can be requested through the agent. When the user executes copyrighted content, the licensing agent verifies the user's license. If a license exists, it is authenticated through the server; if there is no license, a license is issued.

(2) License Issuance Protocol

As shown in Figure 4, the user installs the licensing agent (LA) program and runs the licensing agent. The licensing agent installed on the user's PC verifies the user's license if the user executes an encrypted content. If there is a license present, the license is authenticated using the server; if not, a license is issued by connecting to the server.

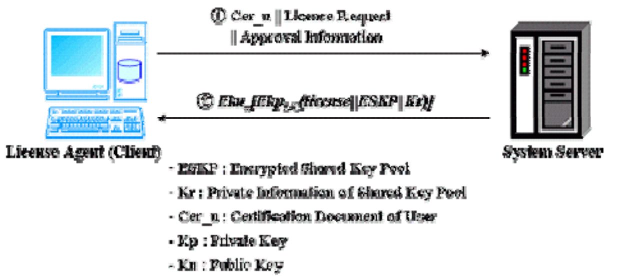
① Cer_u || License Request || Approval Information
② Eku_[Ekp_s(license||ESKP||Kr)]
License Agent (Client)
System Server
- ESKP : Encrypted Shared Key Pool
- Kr : Private Information of Shared Key Pool
- Cer_u : Certification Document of User
- Kp : Private Key
- Ku : Public Key

Fig. 4. License issue Protocol

The server extracts personal information from the shared key pool according to the relevant certificate's information extracted from personal information and the encrypted moving image's secret key (Ks) the encrypted shared key pool is then sent to the user together with the license.

As illustrated in Figure 5, the license is a digital document in which license ID, user ID, content ID, expiration date, license verification period, the user's public key information, server identification information, privileges etc., are stored; it is signed by the server using its private key to guarantee validity.

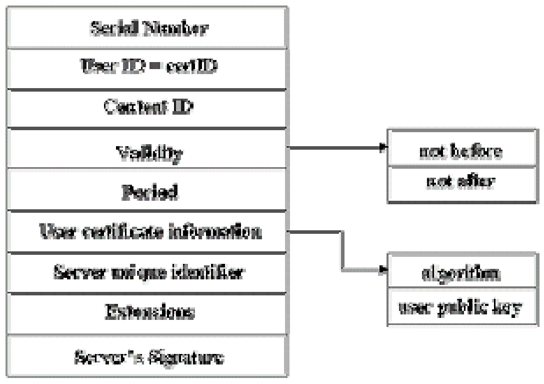

Fig. 5. Architecture of License

The serial number is the serial number of the license. The user ID is information for identifying individual users. Since users are identified by the user's public key certificate, the same value as the cert ID, which is the certificate's identification number, is used. Validity is the period of validity for the license: it specifies the license start date and the license end date. Period is the verification period for verifying the validity of the license for the client system. User certificate information shows the public key and the public key algorithm used for the public key certificate. The server unique identifier is information needed for identifying the server, while extensions refer to the value of the extended area. The server's signature is the value signed by the server to authenticate the license.
The user agent that receives a license then executes the moving image if the license is valid, and the license is then authenticated using the server's public key. Here, for security, the data to be transferred is signed by the server's private key and encrypted using the user's public key as follows so than exposure during transfer can be prevented.

$$Ekp\_s\ [Eku\_u\ (licence||ESKP||Kr)]$$

Here, ESKP is the encrypted shared key pool, and Kr is the user's private information extracted from the user's certificate and the shared key pool. Kp and Ku are the public key system's private key and public key, respectively, and Kp_s is the server's private key while Ku_u is the user's public key.

(3) License Authentication Protocol

As illustrated in Figure 6, when the user executes an encrypted content, the licensing agent checks whether a license is present. If it is not present, a license is issued according to the license issuing protocol. If a license is present and the client is on-line, authentication for the relevant license is requested to the server. If off-line, the license is authenticated for the client according to the information stored in the user's database, supporting the execution of up to a specific number of content.

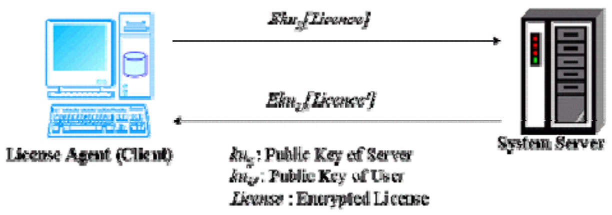
$Eku_s[Licence]$
$Eku_u[Licence']$
License Agent (Client)
System Server
ku_s : Public Key of Server
ku_u : Public Key of User
License : Encrypted License

Fig. 6. Licence Authentication Protocol

The system server, when it receives a license authentication request from a licensing agent, compares the relevant license's information stored in the database with the client's license information, and then corrects and authenticates the license information. The license storage database status is shown in Table 6.

Licence ID	user ID	Data ID	Auth.	Auth. Value	Conn. Count	System num.	Private Info.
1	11111	s11111	1	10	2	203.253.21.174 162.192.56.39	12345678
2	22222	a11111	2	04-3-12	1	203.253.27.162	87654321
3	33333	k11111	1	5	1	223.65.198.45	33333333
:	:	:	:	:	:	:	:

Table 6. License Information in Database

The system information of the user's license is verified and added to the server's database. If the license is time-limited up to a certain date, it checks whether the license has expired, and if the license is a count-limited one, the license information is corrected, the corrected license is encrypted using the user's public key, and it is then transmitted.

Eku_u [Licence']

The user agent which has received the corrected license from the server corrects the client side database information based on that license and generates the secret key (Ks) based on the license's private information and the shared key pool's value. It then decrypts the encrypted content using the secret key (Ks) for display to the user.

4. Implementation

The proposed environment for development is based on an Intel(R) Pentium-IV CPU (2.4GHz), 512MB RAM, and the MS Windows 2000 Server operating system. The programming languages used for implementation were Visual C++ 6.0 and Delphi 7.0; we also used MS-SQL 2000 as the DBMS. The server can view the client side moving image file storage folder as in Figure 7, and real-time monitoring is possible through the user agent. Therefore, it is possible to monitor illegal usage and calculate statistical information on illegal activities.

Fig. 7. Data Monitoring of Client Side

The encryption of moving image data is performed as shown in Figure 8. The encryption key and the position of the moving image information to be encrypted are entered to perform encryption. The information used for encryption is stored on the DBMS. The proposed system was implemented so that either the SEED or AES can be selected as the encryption algorithm.

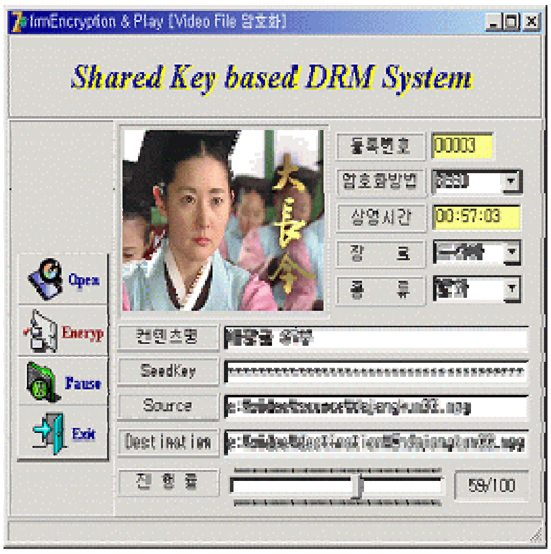

Fig. 8. Encryption Processing for Video Data

The decryption of the moving image is performed as illustrated in Figure 9, and is done by the agent when the user executes the moving image. When executing the user's content, the user agent decrypts the moving image using a key.

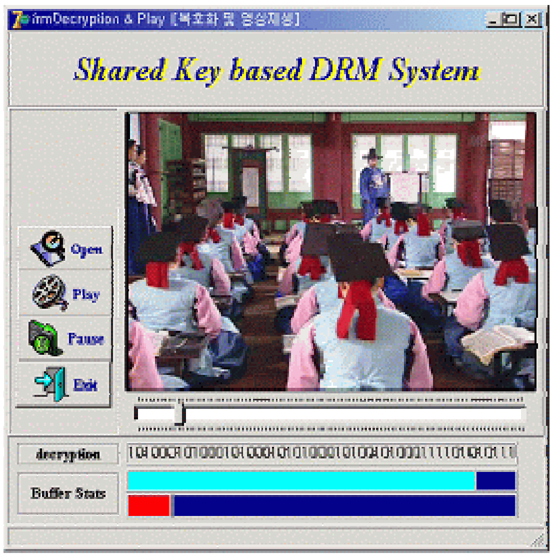
DrmDecryption & Play [복호화 및 영상재생]
Shared Key based DRM System
Open
Play
Pause
Exit
decryption
Buffer Stats

Fig. 9. Decryption Processing for Video Data

5. Performance Evaluation

To evaluate the performance of the system proposed in this thesis, we measured the encryption time of the video itself and the initial play-out delay time according to the decryption time. The conventional method of first decrypting an already encrypted video data file (non-real-time decryption method), and the method proposed in this thesis, that is, playing out while performing decryption in real-time (real-time decryption method), were implemented and their respective execution times were measured with the results shown in Table 7. For accurate time measurement, the video file was divided by minutes.

File Size (MBytes)	Execution Time (Minutes)	Decryption Time (Seconds)	Delay Time of Existing Method (Seconds)	Delay Time of Proposed Method (Seconds)
6.83	1	0.76	2.42	2.42
13.66	2	1.57	5.50	2.42
20.49	3	3.00	9.24	2.42
68.31	10	9.05	25.01	2.42
204.94	30	24.31	49.11	2.42
423.94	60	41.62	82.06	5.50
635.92	90	59.46	104.72	5.50

Table 7. Delay Time Comparison for Execution Time

Each delay time is the time required for decrypting the encrypted video data file and play-out. This time is the sum of the video data file's decryption time and the loading time. In general, the video file's loading time differs for each video player; therefore, in this experiment, the loading time was processed together with the decoding time to calculate the delay time. In the proposed method, video data files are played out concurrently while performing execution by using double buffer scheduling. Therefore, we can see that the play-out delay time significantly decreased compared with the conventional method. Figure 10 illustrates the measured encryption time and the play-out delay time of the conventional method and the proposed method in graph form

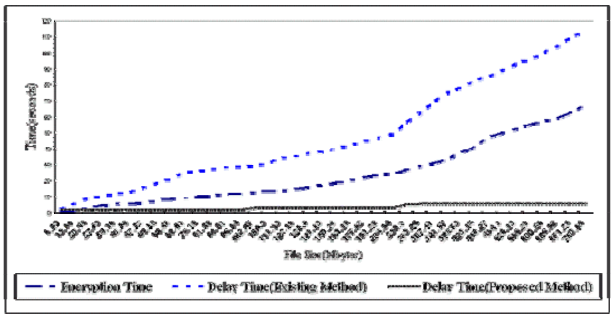

Fig. 10. Encryption Time and Delay Time Comparison.

We see that, in the conventional method, both decryption time and play-out delay time increase proportionally with file size. However, in the proposed method, the actual delay time is decreased since decryption is performed concurrently with play-out while the decryption time increases with file size.

6. Conclusion

In this paper, first, we have proposed a new encryption scheme which encrypts the video data's I-frame for encryption of moving image data. Second, a licensing agent which enables automatic user authentication and data decryption when multimedia data is encrypted at the system server is executed at the client system. The licensing agent performs PKI-based user authentication and encryption/decryption of moving image data using a shared key pool when executing the user's multimedia data. After encrypting a moving image file using a secret key, the user's private information is extracted from operations with the PKI certificate and the shared key pool, and then the secret key is transmitted to the user hidden within the shared key pool so that the user cannot expose the key to the outside by accessing the secret key. If a key is exposed, the path of exposure can be traced. The shared key pool system is a methodology for effectively countering the exposure of the key by a user. Third, to reduce the huge playout delay time occurring due to decryption when executing high capacity multimedia data, a double buffer configuration was implemented and a real-time decryption method using efficient buffer scheduling was proposed. If a user executes a moving image file, the licensing user authenticates the license at the system server, calculates the secret key based on the user's personal information and the shared key pool information, and then the moving image file can be decrypted for play-out. Here a double buffer is employed, which enables execution if a part of the file is decrypted, so that decryption can be performed in real-time for the user.

The proposed system was designed, implemented, and tested using video data files of various sizes for performance evaluation. We confirmed that the proposed system can significantly reduce delay time including decryption time compared with existing systems when playing out large capacity moving image files in the client system.

The proposed system will be effective if deployed as a DRM system for moving images such as protecting the copyright of Internet-based movies and music videos, and CF. In addition, further research needs to be done to improve support for wireless network devices such as PDAs.

Acknowledgments

Insert acknowledgment, if any.

References

[1] Joshua Duhl and Susan Kevorkian, "Understanding DRM system: An IDC White paper," IDC, 2001.
[2] Jai Sundar B., Spafford E., "Software Agents for Intrusion Detection," Technical Report, Department of Computer Science, Purdue University, 1997.
[3] J.Dubl,"Digital Rights Management: A Defination", IDC 2001.
[4] J.Dubl, S.Kevorkian, "Understanding DRM system: An IDC White paper", IDC, 2001.
[5] Kentaro Endo, “The Building up of national Regional and International Registers for works and objects of related rights," Proc. of International Conference on WIPO, Seoul, Korea October 25-27, 2000.
[6] V. K Gupta, "Technological measures of protection," Proc. of International Conference on WIPO, Seoul, Korea October 28-29, 2000.
[7] P. Vora, D, Reynolds, L. Dickinson, J. Erickson, D. Banks, "Privacy and Digital Rights Managements", A Position paper for the W3C Workshop on Digital Rights Management, January 2001.
[8] D. K. Mulligan and A. Burstein, “Implementing Copyright Limitations in Rights Expression Languages,” in 2002 ACM Workshop on Digital Rights Management, Washington DC, November 18 2002.
[9] J. S. Erickson, “Fair use, DRM, and trusted computing,” Communications of the ACM, vol. 46, no. 4, pp. 34–39, April 2003.
[10]Microsoft's press releases of the PocketPC 2002 launch, Oct 8, 2001. Available at www.microsoft.com/presspass/events/pocketpc2002/default.asp.
[11] John Linn, "Trust Models and Management in Public Key Infrastructures," Technical Notes and Reports of RSA Laboratories, November 2000.
[12] Russ Housley and Tim Polk, Planning for PKI, John Wiley & Sons, 2002.

Jaepyo Park received his BS, MS and PhD degrees in computer science from Soongsil University, Seoul, Korea, in 1996, 1998 and 2004, respectively. Currently he is a Part-time Instructor in Soongsil University, Seoul, Korea. His research interests include multimedia security, PKI, network security, cyber forensic, and mobile network.

Moon-Seog Jun received his B.S. at Soongsil Univ, M.S. and Ph.D degrees in computer science from University of Maryland, USA, in 1985, 1988. He taught computer Network at Morgan State University and researched Physical Science Lab. New Mexico, USA. He has been taught and researched as a full professor at Soongsil University. His research interests include Network Security, Cryptography, Computer Algoruthms, and Network Protocol.