ISBN 978-981-11-0008-6
Proceedings of 2016 6th International Workshop on Computer Science and Engineering (WCSE 2016)
Tokyo, Japan, 17-19 June, 2016, pp. 143-149
doi: 10.18178/wcse.2016.06.024

A Key Distributed Storage Mechanism Based on User Credibility

Yuxi Lin [1], Zhaoheng Wang [2], Xingjun Wang [1+], XinYue Che [2], Feng Tian [2], Chengyu Li [2]

[1] Department of Electronic Engineering/Graduate School at Shenzhen, Tsinghua University, China
[2] Northern United Broadcasting Television Network Corporation, China

Abstract. Nowadays, more and more people are accustomed to using smart mobiles devices. According to NetMarketShare, in September 2015, Android occupies 53.54% of the smart mobile devices market shares. Digital Right Management (DRM) is recognized as an effective technology to protect the copyright of users. In DRM, the protection of private key is the basis of the system security. This paper presents an improved key distributed storage solution based on user credibility to protect user private key.

Keywords: DRM, private key, distributed storage, user credibility, Android, security.

1. Introduction

Nowadays, Android is facing increasingly serious security threats. The reason Android security accidents caused frequently are mainly as follows: first, Android provides an open environment, its flexibility and various customization requirements are achieved in the loss of part of security under the premise. Second, developers focus more on functional development, and lack of attention to software security. Also, many existing Android-based security mechanism cannot meet a variety of specific application requirements.

DRM is a widely used technology in digital product copyright protection [1]-[3]. Many specifications have been launched, such as Marlin DRM [4], OMA DRM [5] and China DRM [6]. In DRM, private key ensures the security of the communication between server and client. The client's private key faces more danger than server's. Researchers have proposed many key protection mechanisms to protect the private key of client, storing the private key in unreadable hardwires can effectively improve the security of user private key but has high cost, Z. Tan proposed a CA system based on server-end private key storage method in [7]. In this paper, we will propose an improved key distributed storage mechanism based on user credibility.

This paper is structured as follows: the next section describes some related technology and my previous work, and then we will proposes our improved solutions, the last section is our conclusions.

2. Related Work

In this section we will first introduce the establishment of security channel based on China DRM technical specification. And then, we will introduce an existing key distributed storage technology which needs improvement. After that is our previous improvement program.

2.1. Secure Channel Establishment

Referring to China DRM Technical specification, we introduce the process of secure channel establishment which is divided into three parts: certificate exchange, key agreement and challenge response.

- Certificate exchange

In this process, client and server verify each other's identity by checking each other's certificate.

- Key agreement

+ Corresponding author. Tel.: + 18038153071.
E-mail address: wangxingjun2015@163.com.

In the process of key agreement, client and server negotiate a session key, which is used encrypt the session content after the secure channel is established.

- Challenge response

After the session key is generated, client and server will need to validate the correctness of the session key through the process of challenge response.

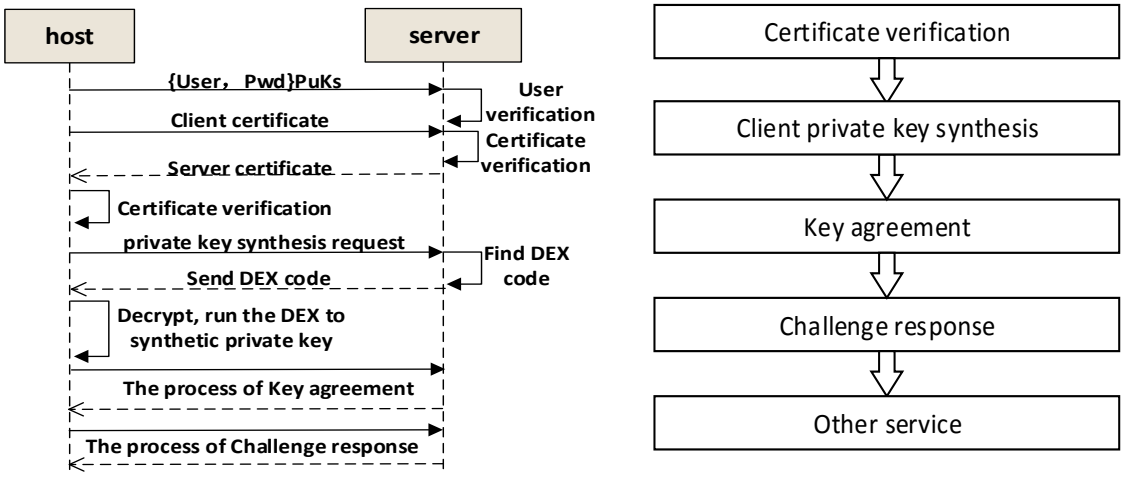
In the above processes, server and client's information transmitted are encrypted with each other's public key. And only the one with the corresponding private key can decrypt the information. So, the security of private key is the most important part of DRM system.

2.2. Key Distributed Storage Technology

Tian proposed a key distributed storage mechanism in [8] based on the secure channel establishment in China DRM Technical specification.

In Tian's solution, client private key is divided in two parts, 20% of private key is stored in server, and the other 80% private key message is stored in a 50k file in client. When client need to synthesize its private key, it apply to server to get the private key synthesis message, and then synthesize its private key.

Here, we introduce the process of Tian's solution. The first step is user verification and certification verification. Secondly, client send private key synthesis request to server, server find the corresponding DEX code, and send it to client, client decrypt the DEX code [9], and run it to get the 80%private key message and then synthetic client private key. The protocol descripted above can be descripted as (a) in Fig. 1. In the figure, User refers to user's account, Pwd refers to corresponding password of the account, PuKs refers to the public key of server, the form of {User, Pwd}PuKs refers to user's account and password is encrypted by server's public key. The structure of Tian's solution can be expressed as (b) in Fig. 1.



(a) (b)

Fig. 1: Tian's key distributed storage protocol (a) and its scheme frame (b).

Tian divides client private key into two parts, and stores them separately. Thus, hackers will have to attack server and client in order to steal client private key. Tian's ideal is a great innovation of the private key protection. However, one point we need to point out, Tian did not design a secure transmission scheme for DEX code transmission.

2.3. User Credibility

Users will have to carry out some necessary action to connect to server. As is known to us, there are certain difference between the operating behaviour of legitimate users and illegal users. On the other hand, we cannot clearly determine whether a user is legitimate or not by some of his behaviour. Therefore, we should provide a computable metric method for users, rather than simply defining the user as trusted or non-trusted. Yan has proposed a good solution for user credibility calculation [10].

Firstly, a table of user credibility will be maintained (see Table I), in this table, some of the user behaviour we care about are recorded. Certificate ID refers to user's Certificate ID. Credibility is the measure value of user credibility, and it ranges from 0 to 1. Logon Counts refers to the logon frequency of user in unit time (one hour). Pause Counts refers to pause times of user in one hour. Pause Interval refers to the total length of pause time in one connection. Logon Client Counts refers to the number of logon clients per unit of time. Authenticate Interval refers to the time spent on Authentication. Request Counts refers to the number of videos requested by user per hour.

Table I: User Credibility

No.	Field	Type	Length
1	Certificate ID	CHAR	20
2	Credibility	FLOAT	
3	Logon Counts	UNSIGNED BIGINT	
4	Pause Counts	UNSIGNED BIGINT	
5	Pause Interval	UNSIGNED DOUBLE	
6	Logon Client Counts	UNSIGNED BIGINT	
7	Authenticate Interval	UNSIGNED DOUBLE	
8	Request Counts	UNSIGNED BIGINT	

According to the Table of User Credibility, some illegal behaviour and corresponding illegal probability can be defined (see Table II).

Table II: Probability Setting of Illegal Behavior

No.	Illegal behavior	Probability
1	In unit time to disconnect > 30 times	0.4
2	Pause during watch video in unit time > 30 times	0.05
3	More than 5 terminals log in in unit time	0.1
4	Time interval between authentications > 30 seconds	0.4
5	More than 30 videos are requested in unit time	0.05

Through this table, we can calculate the value of user credibility. Now, we will show the method of calculation, N species of user illegal behaviour were defined, and the severity of each behaviour is P1, P2,……PN, the range of them is 0 to 1. We statistics the number of these N kind of illegal behaviour in a certain period of time. The value are T1, T2… TN. By these data, we can calculate the illegal probability of user as in (1).

$$P = \frac{\sum_{i=1}^{N} T_i \times P_i}{\sum_{i=1}^{N} T_i} \tag{1}$$

And then, we can calculate the value of user credibility as in (2).

$$K = 1 - P \tag{2}$$

By now, we can get user credibility of each user. User credibility is a good measure to user, in this paper, we will introduce it to our solution.

2.4. Our Previous Improvement Solution

In my previous work, I have proposed a secure transmission protocol to transfer the private key information from server to client.

The complete process of the improved program is shown in Fig. 2.

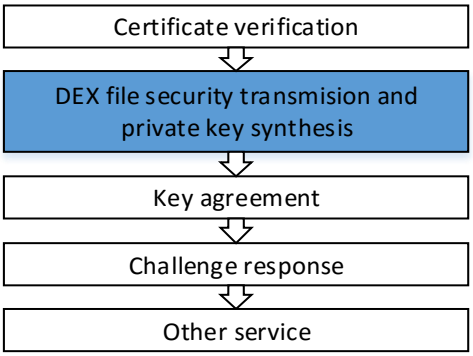


Fig. 2. Flow chat of improved key distributed storage system.

The process of temporary session key negotiation is as follows:

- Client send private key synthesis request to server.
- Server send random number R_0 to client.
- Client uses its username, password and R_0 to make a salted hash[11], and send the result to server.
- Server do the same calculation to user as client did, and compare the result with the result received from client for user verification. Then send the verification result to client.
- Client send client certificate to server.
- Server verify client certificate, and send server certificate to client.
- Client verify server certificate.
- After certificate verification, client generate a random number R_1 and one-time session key TK and encrypt them with server public key, and then send them to server.
- Server generate two random number R_2, R_3, then encrypt R_2 with TK and send it to client, and send R_3 to mobile number the user binding.
- Client calculate the hash of $R_2||R_3$, and send the result to server, and then calculate the temporary session key SK as in (3).

$$SK = Hash(r_1 \,||\, (r_2 \,||\, r_3)) \tag{3}$$

- Server check the received hash value from client, if correct, server calculate the temporary session key SK as in (3).

The process mentioned above can be descripted in UML as in Fig. 3.

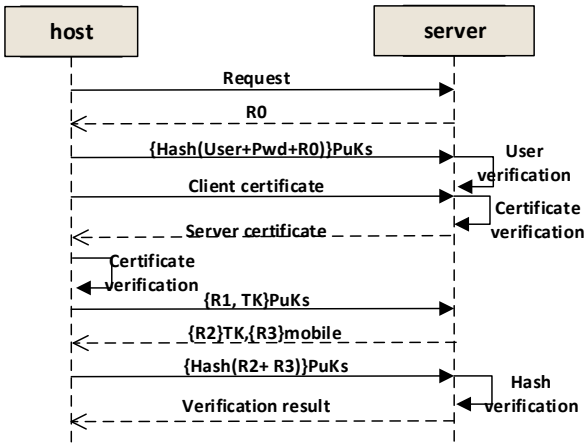


Fig. 3: Improved key distributed protocol.

We adopted the two factor authentication and the idea of "salt" to generate the temporary session key. The temporary session key is generated in two session channels and the random number R_3 are the "salt" value of R_2, which makes it not possible to steal random number R_2 and R_3 in methods such as Brute force.

Our improved scheme can effectively improve the security of key distribution storage system. However, even if we are able to provide a prefect transmission scheme that defence all network attacks, hackers can still get the user's account and password and other information through other ways, such as, keyboard monitor, screen record, and even password psychology. That means, we need a better improved solution.

3. Our Solution

This section will introduce user credibility to key distributed storage method. Our improved solution contains the following modules: one time password agreement module, reverse challenge-response module based on user credibility, private key information secure transmission module. As the first and the third module have been introduced in section 2.4, we will only do a brief introduction to them, and mainly introduce our improved solution: reverse challenge response based on user credibility.

The first step is temporary session key negotiation which is introduced in section 2.4. The temporary session key is used for the transmission of the client's private key synthesis message.

After the process of temporary session key agreement, server will calculate the value of user credibility.

The method of calculating user credibility has been carried out in section 2.3. The range of K is 0 to 1. Here, in our solution we divide user credibility into three sections, not-credible section, to-be-verified section and credible section, the boundary values of which are K_1 and K_2, as shown in Fig. 4.

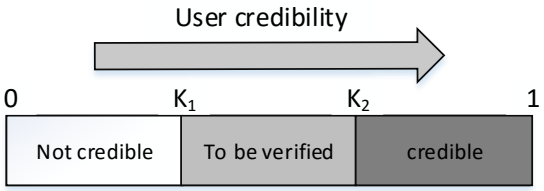


Fig. 4: User credibility interval.

If the value of user credibility is in the range of $[0, K_1]$, this user is supposed to have high possibility to thread the system, and server will finish the connection with client. If the value of user credibility is in the range of $[K_1, K_2]$, this user is supposed to have low possibility to thread the system, and server will start the process of reverse challenge response to client. The detail process of it will be introduced in the next paragraph. If the value of user credibility is in the range of $[K_2, 1]$, this user is supposed to be credible, and server will start the process of DEX file security transmission and private key synthesis. Different systems can adjust the range of the three section based on their own security request.

Now we will introduce the process of reverse challenge response. A table of user login history is maintained (see Table III) by server and client. Each time user log in, the table generates a log in history.

Table III: User Login History

No.	Field	Type	Length
1	Date	TIMESTAMP	
2	Device IMEI-ID	CHAR	15
3	User ID	CHAR	20
4	Content ID	CHAR	20
5	Action	CHAR	1
6	IP	VARCHAR	20
7	CONNECTION	TIMESTAMP	

The following is an introduction to the fields.

Date: the time user log in.

Device IMEI-ID: unique identification code for device

User ID: SHA-1 hash of user name.

Content ID: the ID of digital content that user operate on.

Action: This field records user's operation, 0 represents authentication request, 1~5 represents status of authentication process, 6 represents play, 7 presents pause, 8 represents download, 9 represents quit.

IP: the IP address of client.

CONNECTION: this field records a user's connection to server, time recorded when each connection is set up. As long as the connection continues, all user behaviours are belong to this connection.

Each behaviour of user will generate a user behaviour record, which store in database of both client and server respectively. The database stored in client is encrypted, and the encryption key is obtained by calculating the SHA-1 hash of IMEI-ID and MAC address. When server need to start the process of reverse challenge response to client, server sends its challenge content, the content of CONNTCTION field in user login history database. When client receive the challenge content, it decrypt the user log in history database, and find out the user behaviour records of the connection. Client make a hash of them and send it to server. Server finds user behaviour records of the same connection in server user log in history database, and make hash of them, then, server compare it with the received value, if equals, the protocol goes to the process of client private key information security transmission, else, server quit the connection.

The process of reverse challenge response is shown in Fig. 5.

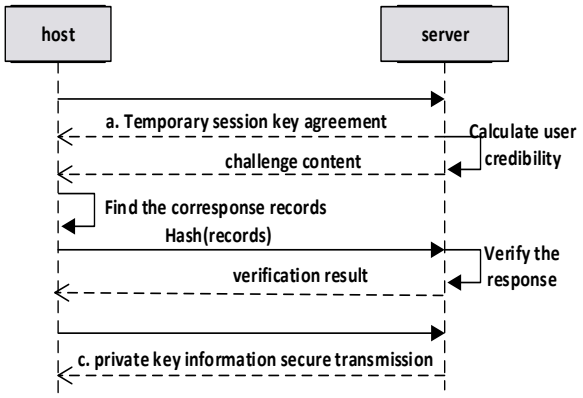


Fig. 5: Reverse challenge response.

After the process of reverse challenge response, the protocol goes into client private key information secure transmission. Server encrypt the 20% private key and the DEX code with the temporary session key and then send it to client. Client decrypt the information, run the DEX code to get 80% private key from 50k file. And then, client synthetic private key.

4. Conclusion

The security of private key is the most concerned of DRM. Hackers can not only attack the communication between client and server, but also attack terminal equipment to get useful information, such as, keyboard monitor, screen record, etc. Our previous improvement scheme introduced above has a good resistance to common network attack, such as network monitor, replay attack and MITM attack. In this paper, we introduce the concept of user credibility which is a good indicator to measure the credibility of user, user credibility is acquired by some user behaviours we concerned. If hackers attack terminal equipment to execute some illegal operations, these operations will be captured by system, thereby reducing the credibility of user. By this, we provide a new method for user authentication which can not only resist stronger network attacks, but also has a good defence against some terminal device attack.

In the future work, we will carry on social expansion to the measurement of user credibility. By analysing the social network of user, we hope to capture some data features associated with user's credibility to conduct a more comprehensive measure of user's credibility.

5. Acknowledgements

Here I want to take this chance to thanks to my tutor: Xingjun Wang, In the process of composing this paper, he gives me many academic and constructive advices, and helps me to correct my paper.

At the same time, I would like to appreciate Chen Tian senior, who guided me to start my subject from zero, and direct me to complete the paper patiently.

At last, I am very grateful of my dear friends, Zhiyong Li, Chao Cheng, who offer me the confidence and discuss with me about my paper.

6. References

[1] L. Xie, C. Tian, X. Wang A Device Management and Credit Evaluation System in Home Network Domain, *Lecture Notes on Information Theory* Vol, 2013, 1(3).

[2] Y. Zhang, C. Yuan, and Y. Zhong. Implementing DRM over Peer-to-Peer Networks with Broadcast Encryption. *Advances in Multimedia Information Processing – PCM 2007*. Springer Berlin Heidelberg, 2007:236-245.

[3] Y. Zhao. An implementation of Trusted Computing based on Trusted Application [D], Jilin, Jilin University, 2013

[4] Marlin Architecture Overview (2007). Marlin Developer Community. [Online]. pp. 6-20. Available: http://www.marlin-community.com/public/MarlinArchitectureOverview.pdf.

[5] OMA2.0 DRM Specifications, Open Mobile Alliance, OMA-TS-DRM-DRM-DRM-V2_1 -20070724-C, 2007.

[6] China DRM Forum Home Network Standard Architecture, China DRM Forum, unpublished V2.0, 2009.

[7] Z. Tan, T. Si, and Y. Dai. CA system in network computer environment based on server-end private-key storage mechanism. *Journal of Tsinghua University* 47.7(2007):1208-1211.

[8] C. Tian, L. Xie, and X. Wang. Combination of DRM and Mobile Code: A Practice to Protect TV Contents and Applications on Android Smartphone. *Networking and Distributed Computing (ICNDC), 2013 Fourth International Conference on* IEEE, 2013:89-93.

[9] Jack_jia, http://blog.csdn.net/androidsecurity/article/details/9674251.

[10] F. Yan, X. Wang. Research of PC Video Content Protection Method[D], Beijing, Tsinghua University, 2010

[11] Salted Password Hashing - Doing it Right, https://crackstation.net/hashing-security.htm