Future Decryption: Secure the Time Sensitive Information

Bessie C. Hu
Dept. of Computer Science
City University of Hong Kong
Hong Kong, China
bessiehu@cs.cityu.edu.hk

Anthony Y. Fu
Dept. of Computer Science
City University of Hong Kong
Hong Kong, China
anthony@cs.cityu.edu.hk

Xiaotie Deng
Dept. of Computer Science
City University of Hong Kong
Hong Kong, China
csdeng@cityu.edu.hk

ABSTRACT

In many real cases, private documents MUST be publicized later, when there is a requirement to protect it before a predefined time, such as testament, examination papers, bid proposals, and financial report, etc. We want to disclose them when it is valid time for other users to access the document. However, none of the existing document protection approaches can fulfill such requirement up to now. In this paper, we proposed an approach to address such a real-world security scenario.

1. INTRODUCTION

1.1 Motivation

Time is always an essential issue in cryptography and information protection. In digital world, given sufficient long time, any secret can be broken under some computation power. What is more, some of the secrets are required to be private ONLY within a specific period. Sometimes, the time to open the secret attracts quite a lot of public interests. Examination paper should be disclolsed to students at the time exam starts. No student has the privilege to access it in advance. A company's financial report should be publicized to all of its stock holders at the same time to ensure the fairness. Other time sensitive information protection requirements are not difficult to be found. Bid proposals from different competitors are required to be opened only in the bidding day. A testament is expected to be disclosed to relative persons after its owner passes away. All these issues have something in common: these secret documents require to be publicized at a pre-determined time. No user should be able to get the protected information in advance. These potential requirements in real world motivate us to consider it as an interesting problem. In this paper, we devise a method by designing a novel algorithm to fulfill such requirements.

1.2 Related Works

There are several approaches in the market to protect documents on different security aspects. MS Office offers a simple way[3] to protect documents from unauthorized access using password entry. However, this build-in MS Office security feature is not really secure. Many password crackers are available in the Internet, which can easily crack the protected documents, such as Advanced Office Password Recovery[2].

Rohos Company developed *Teslain Encryption Pack*[4] to protect MS Office documents using a strong encryption(AES 256 bit). This software is an MS Office toolbar. To save and encrypt a document, a password should be given. User also needs the password to decrypt and open the encrypted document. This approach is claimed to be very strong. It should take 1 year to perform a Brute Force attack against a well-chosen password at 6 symbols' length. However, in case the password is lost, the protected document is left to be irretrievable.

AegisDRM provides a more flexible solution: *PaM*(Protector add-in for Microsoft Office)[1]. Document authors control not only who may access a document, but what they can do with it, such as reading, editing, copying, etc. This approach uses a central RightServer to control all the access rights for different users. Along with the powerful functionality, it requires the complete trust to RightServer. For a company, archiving all essential documents in third party is not preferable. *PaM* also provides the options of *Start Time* and *End Time* to restrict user's access right within a time slot.

2. SYSTEM DESIGN

2.1 Security Requirements

Our system is designed to protect the documents before a specific time, while it must be publicized after that time. There are two requirements which should be achieved for this security purpose:

1. The owner/author who knows the secret key can decrypt the document at any time to modify the document as he wishes.

2. Other users who have no secret key can only open the document after a pre-determined time, which is set by the owner.

Most of the current solutions of document protection achieves the first requirement. However, to the best of our knowledge, none of them can solve the second one. For the *built-in MS Office security feature*, the time an attacker is able to break the password depends on the power of password crackers. For *Teslain Encryption Pack*, the security of document relies on choosing proper passwords. For *PaM*, the reliability of system depends on the complete trustworthiness of Right-Server. Document owners have no opportunity to control the time effect in these systems.

Our design aims at solving these side effects by adding light weight accessary computational effort, such that the owner of a document can actively control the disclosing time, rather than passively wait to be broken.

2.2 The Algorithm

The system will be installed in a device (like computer) with computational power and storage. It also requires an off-line time clock accessory to avoid time resetting.

Definition of System Parameters:

1. D: Document to be encrypted.
2. T_E: Proposed time to encrypt the document (default is current time).
3. T_D: Proposed time to decrypt the document (a time in future).
4. s: User Secret Value used to generate the keys for encryption and decryption, only known by owner.
5. n: The number of encryption operations user required between T_E and T_D.
6. $\{T_0, T_1, T_2, ...T_n\}$: the time set, where $T_0 = T_E$, and $T_n = T_D$. $\forall i \in \{1, ..., n\}$, $T_i - T_{i-1} = (T_n - T_0)/n$.
7. $\{K_0, K_1, K_2, ...K_n\}$: the key set, where $K_0 = Hash(s, T_0)$. $\forall i \in \{1, ..., n\}$,$K_i = Hash(K_{i-1}, s, T_i)$.

Process of Encrypting and Decrypting Documents:

1. To encrypt D, its owner should upload D to the system, plug in the time clock accessory to synchronize the time, and input T_E, T_D, s, and n.
2. Upon receiving these parameters, system calculates the time set $\{T_0, T_1, T_2, ...T_n\}$, and key set $\{K_0, K_1, K_2, ...K_n\}$.
3. System then encrypts D by calculating $E(D) = E_{K_1}(E_{K_2}(...E_{K_n}(D)))$. We denote $E(D)$ as V_1.
4. System stores $E(D)$, time set $\{T_0, T_1, T_2, ...T_n\}$ as well as key set $\{K_0, K_1, K_2, ...K_n\}$, and destroys D,T_E, T_D, s, n.
5. To decrypt $E(D)$ and retrieve D, a user needs to plug in the time clock accessory, and provide T_E, T_D, s, n regarding to D.
6. The system will take the parameters to calculate a new time set $\{T'_0, T'_1, T'_2, ...T'_n\}$ and key set $\{K'_0, K'_1, K'_2, ...K'_n\}$. If they are matched with stored time set $\{T_0, T_1, T_2, ...T_n\}$ and key set $\{K_0, K_1, K_2, ...K_n\}$, system will decrypt $E(D)$ by calculating $D = D_{K_n}(D_{K_{n-1}}(...D_{K_1}(E(D))))$. If they are not matched, system will find out T_i in time set such that $T_i \leq T_C < T_{i+1}$, where T_C is the current time of time clock. Decrypted value $V_{i+1} = D_{K_i}(D_{K_{i-1}}(...D_{K_1}(E(D)))) = E_{K_{i+1}}(...E_{K_n}(D))$ will be provided for the user.

This design is to ensure that author of the document who knows T_E, T_D, s, n can retrieve the document at anytime. While other users who have no knowledge of T_E, T_D, s, n can only wait until T_D to open the document.

2.3 Security Analysis

This design is mainly used to avoid unauthorized users to open the document within some time period. Considering with the efficiency of symmetric key cryptography, as well as the small key size applied, n decryption operations are affordable for legal user. However, the cost for an attacker to decrypt the document is very high, about $2^{k(n-i)-1}$ decryption functions on average(*see details below), where T_C is the current time, and $T_C \in [T_i, T_{i+1})$, for all $i \in \{1, 2, 3, ..., n\}$. In fact, cost of attacker will be reduced when current time T_C is closer to T_D. That also reflects the time effect in real-world scenario that value of protected information decreases along with the time passes away.

* Analysis of cost for attacker:
We assume that T_D is a public knowledge, and an attacker can always run decryption oracle to ask for the current decrypted value of D, which appears to be a random string before T_D. To obtain the interval between T_{i-1} and T_i, a positive attacker can keep requesting the decrypted value from system, and check when the random string is changed from V_i to V_{i+1}, say at T_i. He can also get K_i by brute force to try all 2^k possible values of K_i and check whether $D_{K_i}(V_i) = V_{i+1}$. However, without the knowledge of s, the attacker is not able to calculate K_{i+1} based on K_i. Thus, to calculate V_{i+2}, the attacker also needs to check all possible 2^k values of K_{i+1}. Since V_{i+2} is still a random string, and only V_{n+1} has meaningful content which is finally decrypted document D, the attacker should try $n - i$ rounds 2^k keys at time T_C where $T_C \in [T_i, T_{i+1})$, and on average needs to do $2^{k(n-i)-1}$ decryption functions.

3. REFERENCES

[1] AegisDRM. *protector^{TM}* add-in for ms office. *http://www.aegisdrm.com/*, 2005.

[2] Elcomsoft. Advanced office password recovery. *http://www.crackpassword.com/products/prs/integpack/aopr/*, 2004.

[3] Microsoft. Ms office architecture:security features in office. *http://www.microsoft.com/technet/archive/office/office97/reskit/office97/034.mspx?mfr=true*, 1997.

[4] Rohos. Teslain encryption pack. *http://www.rohos.com/encryption-pack/*, July 2005.