Chapter 34
Dynamic USBKEY System on Multiple Verification Algorithm

Yixiang Yao, Jinghua Gao and Ying Gong

Abstract On account of the closed products and other defective products in the current market, this paper puts forward and carries out the Dynamic USBKEY System. This system is based on Multiple Verification Algorithm and is able to verify the validity of users' identity in a high-strength dynamic channel. Firstly, the security of the entire system is based on the strength of the random key. The overall design and the adopted algorithm are open. Secondly, it can solve the problems within the channel, the verification method and the program's self-preservation. Thirdly, the system provides a more secure solution under the rapid programming mode. The developers can apply the system on their own programs through the opened cross-language interface. As a result, the development cycle can be shortened and the security strength of their program can be improved.

Keywords USBKEY · One-time encryption · Network security · Software security · Rapid programming

34.1 Introduction

The developers of application software have paid close attention to the encryption-protection of commercial software. In order to protect intellectual property and avoid piracy, a variety of encryption technologies have emerged. USBKEY, one of these technologies, has taken over the market with its extraordinary superiority [1].

The current market is dominated by the third (programmable) and the fourth generation (smart card) USBKEY systems. However, the seemingly high security strength of these systems depends on encrypting the structure of the system itself so that they are insecure when the structural information are let out or cracked. The

Y. Yao (✉) · J. Gao · Y. Gong
Department of Information Security, Computer Science and Technology Institute,
Civil Aviation University of China, Tianjin, China
e-mail: bigyyx@gmail.com

main methods to crack the USBKEY are as follows: (1) cloning or copying, (2) using the debugging tools to trace and decrypt, (3) using blocking procedure to modify the communication between the software and USBKEY so as to acquire the communication data. All these attacking methods are great threatens to the USBKEY systems.

The USBKEY systems are usually used in protecting some specific software or system like CAD and there is no commonly used USBKEY system that can adapt to every user's application. It is necessary to design a new type of USBKEY system which is more secure, universally adaptable and easily installed.

In order to solve the problems described above, this paper comes up with a Dynamic USBKEY System based on Multiple Verification Algorithm. In this system, Multiple Verification is embodied in the diversity of authentication tokens (such as the digital certificates) and the dynamic feature manifests in one-time encryption used in the channel of transmission.

34.2 System Structure and the Principle of Module Design

The design of the USBKEY system depicted in this paper follows the principles shown below: (1) it can be quickly put into use by developers (2) the function of it can be expanded according to users' demand (3) its structure is open and the security strength is entirely based on the random key in transmission (4) the security of channel (5) the safety of its components can be verified (6) the authentication tokens have multiple dimensional patterns.

The entire USBKEY system includes four components: application programs (CreateKey Application, which is designed to create keys, VerifyKey Application, which is designed to verify the created keys, ConfMgr Application, which is designed to manage configurations), USBKEY hardware, the database of the server and PwdGuard Control (the security password control which is used as user interface).

Based on the four components, the system is divided into two states, namely Creating State and Verification State. The Creating State is designed to initialize the USBKEY and update the database. The Verification State mainly deals with cross-validation and controls users' action as is shown in Fig. 34.1. When a user tries to operate the application and meet the verification point, the Verification State will be triggered.

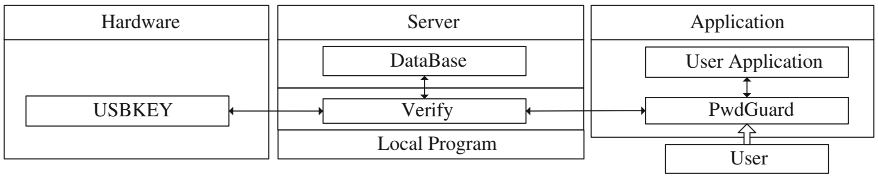


Fig. 34.1 The main structure of the Verification State

All communications of the system are based on their own communication protocols. Therefore, users can develop more plug-ins according to their own demand or adopt rapid programming by using PwdGuard.

34.2.1 Design of Dynamic Transmission Channel

It is generally believed that the security of transmission channel depends on the complexity of the protocol [2]. On contrary, cryptographers believe that any algorithm based on the complexity of protocols can be inducted and conjectured through statistical regularities. However, there is one exception that no one-time encryption can be cracked down even with infinite computing resources [3].

The one-time encryption scheme of the channel is based on the key exchange algorithm under Public-Key Encryption Infrastructure (PKI) and a symmetric encryption algorithm of higher security level. However, Man-In-The-Middle Attacks exist in some of the key exchange algorithms so that the channel needs a trust-worthy Certification Agency (CA) to issue certificates. Those CAs mentioned are beyond the protocols [3]. Since the USBKEY hardware only communicates with VerifyKey, the USBKEY cannot build a CA and is impossible to adopt a complete procedure of exchanging keys directly. Here, we put forward a scheme independent of certificates and CAs: assume that the Creating State is safe, PK and SK, a pair of public and private keys created in the Creating State, are saved in database and USBKEY respectively. Then the Creating State will use the PK and SK directly to complete the key exchange.

For the sake of safety, we set a safety time threshold t. When a time period exceeds t, the VerifyKey will generate a new pair of PKnew and SKnew which will be updated to USBKEY by the old pair of PK and SK.

The procedure is divided into four phases of communications as below:

1. VerifyKey generates message p which will be encrypted into c by using the following formulas then VerifyKey will send c to USBKEY.

$$k = Random() \tag{34.1}$$

$$c = E_k(p) \tag{34.2}$$

2. USBKEY returns ACK after receiving the message.
3. After receiving ACK, VerifyKey changes k to k', then send it to USBKEY.

$$k' = PKA(k,\ SK) \tag{34.3}$$

4. USBKEY gets k and p via decryption.

$$k = PKA(k',\ PK) \tag{34.4}$$

$$p = D_k(c) \tag{34.5}$$

Then v, the data need to be sent back, will be encrypted into v' and then send back to VerifyKey.

$$v' = E_k(v) \tag{34.6}$$

Hereto, the entire procedure of the communications completes.

Besides, we use SSL directly to guarantee the communications between the database and the VerifyKey.

In order to fight against reply attacks, we need to use time-stamps with timeout range in all data packets. Here, we leave out unnecessary details.

34.2.2 The Design of Multiple Verification Algorithm

The traditional mode of token protection is possible to be cracked down. Therefore, it is essential to design a new kind of multi-dimensional transformation method for tokens.

In Creating State, CreateKey needs to produce several units of various verification tokens T_i. For all kinds of tokens, they have their own different ways to transform. In other words, $T_i' = F(T_i)$ in which F may be a one-way function. Then T_i and T_i' will be saved respectively in the USBKEY and the database.

Assume $\Omega\{T\}$ is the sample space of the token in traditional protection mode, $\Omega\{F\}$ is the sample space for the verification algorithm. Computed by this algorithm, the sample space will become $\Omega\{T_i\}$ (i = 1, 2…,n). The sample space computed by the verification algorithm will become $\Omega\{F_j\}$ (j = 1, 2,…,m). X, Y is random variables, then

$$\sum_{i=1}^{n} P(X = T_i) = P(X = T) \tag{34.7}$$

$$\sum_{j=1}^{m} P(Y = F_j) = P(Y = F) \tag{34.8}$$

So under the protection of Multiple Algorithm, the possibility of tokens which may be cracked down is:

$$P(X = T_i,\ Y = F_j) = P(X = T_i)P(Y = F_j) = \frac{1}{nm}P(X = T,\ Y = F) \tag{34.9}$$

Besides, tokens cannot be saved continuously in the memory and they can only be stored separately though self-defined structure of data.

34.2.3 The Design of Mutual Verification and Self-Protection

Since the USBKEY components may be replaced or modified, Mutual Verification is particularly important. Before transmission, both sides which are reciprocally the subject and the object should verify each other through hash verification. The information can only be transmitted when the verifications are passed.

The system also needs to prevent itself from decompiling and tracing. Without the anti-tracing technologies, the software will be exposed by the cracker using debugger and monitor [4]. The common protective measure in the market is packing. We recommend the high strength virtual machine packer which has a more remarkable protective ability.

In addition, we can adopt self-made methods to protect the core codes. For example, the common 0xCC breakpoint can be detected by acquiring the machine code during execution and we can also use the debugging mark in the memory of Windows to judge or avoid debugging etc.

34.3 Module Design

34.3.1 The Design of Program Module

Here is an overview of the design for CreateKey and VerifyKey. ConfMgr is used to manage the configuration. The related parameters can be redefined so as to be compatible with different environment.

1. CreateKey is responsible for Creating State. The main functions include: creating, transforming and saving the verification token, generating and storing the initial PKI parameters, writing the USBKEY and updating the database.
2. As the center of entire Verification State, VerifyKey is transparent to users. When PwdGuard receives a request, the Verify will transfer the request to USBKEY which will return the ID and the relevant token T. According to the ID, a duplication of verification token T' will be granted from the database. Finally, VerifyKey will compare T' and F(T) and use the result to generate execution event which will be executed by PwdGuard.

34.3.2 The Design of Hardware Module

As the carrier of the system's hardware, USBKEY is responsible for storing the authentication tokens and assisting VerifyKey to complete verification process. As is shown in Fig. 34.2, when data arrives at the communication interface, it should be decrypted and decoded by Cryptographer and Protocol Explanation respectively. Then the data has three routes: (a) acquiring a random number (b) being created or acquiring a verification token (c) calling other functions. Particularly, the operations of verification token and function must go through internal memory and then return. Here, the operation of verification token means the creation of verification token (Creating State) and the random reading of verification token (Verification State). While the function is used in two ways: (a) part of the function (used to running in VerifyKey) is executed in the USBKEY and it will return the result (b) the function only used by USBKEY itself.

34.3.3 The Design of Interface Module

The Interface Module is used to connect the user's applications, its function includes monitor and control the user's operation. Based on the ActiveX control, we designed a special InputBox to fight against detecting of asterisk password and KeyLogger. It also has the function of verification.

1. Measures to prevent asterisk password from being detected: under the Windows system, the password InputBox only changes password into asterisk. However, the cracker can use the Handle of the InputBox to get the password. Our new control is designed to avoid this problem by caching the password indirectly and forging a fake display, as is shown in Fig. 34.3a.
2. Ways to prevent the KeyLogger: Message Hook is a mechanism provided by Windows and it can allow the system to monitor the processing of Message [5].

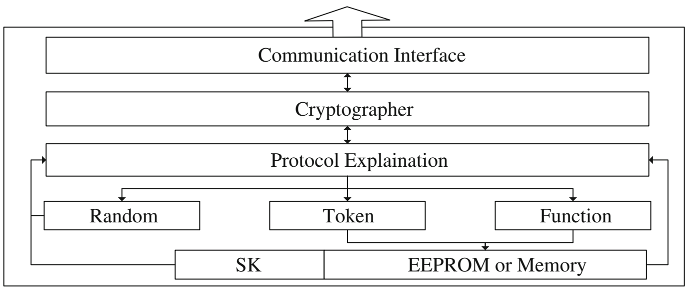


Fig. 34.2 The schematic of hardware module

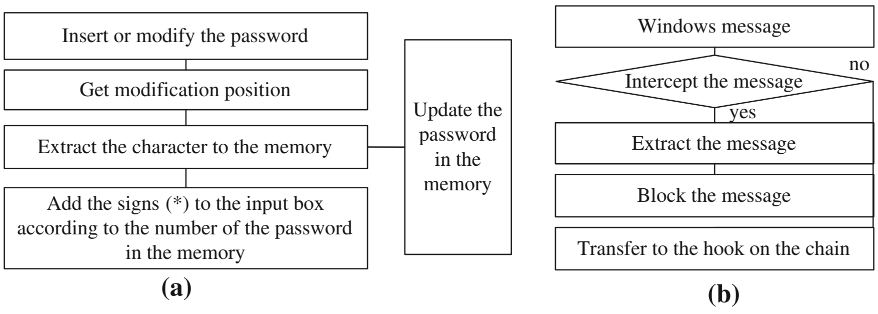

Insert or modify the password
Get modification position
Extract the character to the memory
Add the signs (*) to the input box according to the number of the password in the memory
Update the password in the memory
(a)
Windows message
Intercept the message
no
yes
Extract the message
Block the message
Transfer to the hook on the chain
(b)

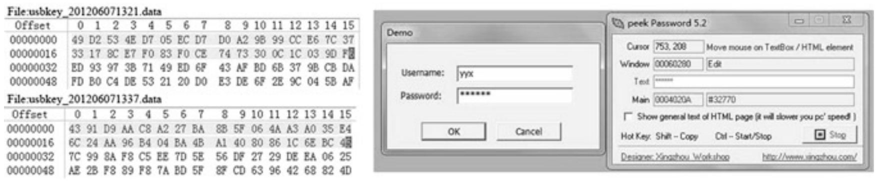
Fig. 34.3 Two processes of PwdGuard. **a** The process of anti-asterisk detect. **b** The process of anti-keylogger

When a new Hook is created, it will be placed at the top of the Hook Chain. As shown in Fig. 34.3b, we can load and unload Hooks repeatedly in small intervals to ensure that all Messages (the Messages of WH_KEYBOARD_LL and WH_DEBUG) will reach the top of the Hook Chain before being transmitted.

34.4 Experimental Tests and Comparison

We installed the database into a testing server and the other components in a common PC. Limited by the space of this paper, we only give the screenshots of the data flow and the detecting result of the asterisk viewer. In particular, Fig. 34.4 (left) shows the data flow in the transmission channel when the system is doing the same operation.

Table 34.1 presents the comparison between the USBKEY system introduced in this paper and a certain one from current market.



```
File:usbkey_201206071321.data
 Offset    0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15
00000000  49 D2 53 4E D7 05 EC D7  D0 A2 9B 99 CC E6 7C 37
00000016  33 17 8C E7 F0 83 F0 CE  74 73 30 0C 1C 03 9D F[illegible]
00000032  ED 93 97 3B 71 49 ED 6F  43 AF BD 6B 37 9B CB DA
00000048  FD B0 C4 DE 53 21 20 D0  E3 DE 6F 2E 9C 04 5B AF
File:usbkey_201206071337.data
 Offset    0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15
00000000  43 91 D9 AA C8 A2 27 BA  8B 5F 06 4A A3 A0 35 E4
00000016  6C 24 AA 96 B4 04 BA 4B  A1 40 80 86 1C 6E BC 4[illegible]
00000032  7C 99 8A F8 C5 EE 7D 5E  56 DF 27 29 DE EA 06 25
00000048  AE 2B F8 89 F8 7A BD 5F  8F CD 63 96 42 68 82 4D
```

Demo
Username: yyx
Password: ******

peek Password 5.2
Cursor: 753, 208 Move mouse on TextBox / HTML element
Window: 00060280 Edit
Text: ******
Main: 0004020A #32770
Show general text of HTML page (it will slower you pc' speed!)
Hot Key: Shift -- Copy Ctrl -- Start/Stop
Designer: Xingzhou Workshop http://www.xingzhou.com/

Fig. 34.4 Data flaw in transmission channel and Asterisk Viewer’s detecting result

Table 34.1 Function comparisons between USBKEY systems

Function	Testing system	Comparison system
Channel transmission	Different data flow each time.	When execute the same function, the data flow is same.
Program modification	No disassembling information in OllyDbg1.1.	Disassembling the core code is possible.
Password detecting	The password cannot be cracked down through password viewer and KeyLogger.	The password can be cracked down.
Customizing components	Each part can be developed independently and allows self-defined interface.	Cannot be extended
System structure	Open, the security is based on random key.	Confidential, part of the security strength is based on protocols

34.5 Conclusion

In this study, researchers came up with the theory of Dynamic USBKEY System on Multiple Verification Algorithm which can reduce the threats of cracking and eavesdropping. Through the protocol's decoupled structure, self-verification and mutual verification, the system achieves custom extension while ensuring safety. Developers can deploy the system into their own applications through simple configuration. How to guarantee the safety of the system when the server is cracked down will be future top priority in researches.

Acknowledgments This work is supported by National Training Programs of Innovation and Entrepreneurship for Students No. 201210059060. We wish to thank Prof. Guo Li for his valuable instructions on earlier drafts of this paper.

References

1. Li, M., Shen, T.: Design of a USB software dog. Chin. J. Electron Devices **29**(1), 205–208 (2006)
2. Xu, M., Zhuge, Z.: Reliability analysis and design of a USB softdog. Chin. Mech. Electr. Eng. Mag. **24**(7), 47–49 (2007)
3. Schneier, B.: Applied cryptography-protocols, algorithms, and source code in C, 2nd edn. China Machine Press, Beijing (2004)
4. Duan, G.: Encryption and decryption, 3rd edn. Chinese Publishing House of Electronic Industry, Beijing (2008)
5. Richter, J.: Programming application for Microsoft Windows, 4th edn. Microsoft Press, Washington (2000)