ICIC Express Letters
Volume **12**, Number **3**, March **2018**
ICIC International ©2018 ISSN 1881-803X
pp. 255–262

THE APPLICATIONS OF PROCESSING MESSAGE SECURITY IN COMPUTER SYSTEM

Lily Lin[1], Huey-Ming Lee[2] and Tsang-Yean Lee[2]

[1]Department of International Business
China University of Technology
No. 56, Sec. 3, Hsing-Lung Road, Taipei 116, Taiwan
lily@cute.edu.tw

[2]Department of Information Management
Chinese Culture University
No. 55, Hwa-Kung Road, Yang-Ming-Shan, Taipei 11114, Taiwan
{ hmlee; tylee }@faculty.pccu.edu.tw

Received September 2017; accepted December 2017

Abstract. *The message passing in the computer network should be processed correctly and kept security to protect against stealing or modifying. In this study, we set each message in different verification encryption tables, and use it to do encryption, verification and decryption algorithm. The verification encryption table contains verification code and encryption data. We use encryption data to encrypt the message and verification code to produce verification bytes and then we insert them to the encrypted message. When the sending message will be processed, it should be decrypted the encrypted message and extracted the verification bytes to verify first. If verification bytes are changed, the file is not correct. In this paper, we presented the processes of encryption, verification and decryption algorithm to protect the message.*
Keywords: Cipher text, Decryption, Encryption, Security data, Verification

1. Introduction.

Shannon [12] discussed the theory of security system in 1949. The functions of security system are security, authenticity, integrity, non-repudiation, data confidentiality and accessed control [11]. NBS (National Bureau of Standards, USA) [9] proposed data encryption standard (DES) in 1977. McEliece [6] used algebraic coding theory to present public key. Merkle [7] presented "One way hash function" and used for digital signature. Miyaguchi [8] developed fast data enciphered algorithm (FEAL-8). NIST (National Institute of Standards and Technology) [10] presented secure hash standard (SHS). Matsui [5] presented linear cryptanalysis to attack DES type security system.

Lee et al. [2] used insertion, rotation, transposition, shift and pack of basic computer operation and encryption data to design encryption and decryption algorithms and store the encryption data in the cipher text. It is different each time and more difficult to do cryptanalysis. Chen et al. [1] presented inserting verification bytes to check its correction. Lin et al. [3] presented encryption and verification to process security data to be correct. Lin et al. [4] also presented processing method of security data in computer system.

In this paper, we present how to process message security from sender to receiver via server. Section 2 is the proposed applications description. We present the proposed model in Section 3. Section 4 shows the algorithms and relative description. We make the conclusion in Section 5.

2. The Proposed Applications Description.

The proposed applications are to process the message in security. The sender site has encryption application to encrypt message to encrypted message first, then sends encrypted message to the server. In the network, the encrypted message may be interrupted or modified. The server decrypts the encrypted message to original message and verified. The server checks the verification whether it is correct. If the verification is correct then we encrypt message to different encrypted message and sends to receiver site again. Via the decryption and verification, the receiver site can receive correct message. We set up sender-receiver user information data base in the server. It contains sender and receiver information. Also, it contains ID and location point of sender and receiver. Each message has a verification encryption table which contains encryption data and verification code. We use encryption data to encrypt message to produce encrypted message and use verification code to produce verification bytes and then insert verification bytes to encrypted message. Each message has a location point to be stored in sender-receiver user information database (SRUIDB). We use the location point and the length of message data to compute the entry point and insert verification bytes and verification encryption table to the entry point of encrypted message. If we want to get the message, we should get the length of this file in the EMDB and location point in the SRUIDB. From this entry point, we extract the verification encryption table and verification bytes. We use verification code of the verification encryption table to produce verification table, use verification bytes and verification table to check the correction, use the encryption data of verification encryption table to decrypt the remaining encrypted message to get the original message.

We explain files, database, tables, programs and operations as follows:

A. Files.

(1) Message file: Length is LM.

(2) Encrypted message: The encrypted message contains cipher text (CT), verification encryption table (VET) and verification bytes (VB) as shown in Table 1. From entry point (EP), we begin to insert VET and VB to cipher text (CT).

Table 1. Encrypted message

CT	VET	VB	CT	VB	CT
	EP				

B. Database.

(1) Sender-receiver user information database (SRUIDB). SRUIDB contains sender-id (SENDER-ID), sender location point (SLP), length of verification encryption table (LVET), receiver-id (RECEIVER-ID) and receiver location point (RLP) as shown in Table 2. SENDER-ID and RECEIVER-ID are used as keys.

Table 2. Sender-receiver user information database (SRUIDB)

SENDER-ID	SLP	LVET	RECEIVER-ID	RLP

(2) Encrypted message database (EMDB). EMDB contains encrypted message filename (EMFN), sender-id (SENDER-ID), receiver-id (RECEIVER-ID), checksum (CKSUM) and length of message (LM) as shown in Table 3. We use EMFN and SENDER-ID as keys.

Table 3. Encrypted message database (EMDB)

EMFN	SENDER-ID	RECEIVER-ID	CKSUM	LM

C. Tables.

(1) Verification encryption table (VET). VET contains verification code (VC) and encryption data (ED) as shown in Table 4.

Table 4. Verification encryption table (VET)

Verification code (VC)	Encryption data (ED)

(2) Verification code (VC). VC contains code (CODE), code increase (CODEI), and number of inserted bytes (NUM) and offset (OFFSET) as shown in Table 5. NUM is number of verification bytes and OFFSET is the distance between two bytes.

Table 5. Verification code (VC)

CODE	CODEI	NUM	OFFSET

(3) Encryption data (ED). ED contains format code (FC), length of left shift table (LLST), number of block (NB), rotated byte (RB), left shift table (LST) and flag (FLAG) as shown in Table 6.

Table 6. Encryption data (ED)

FC	LLST	NB	RB	LST	FLAG

where

FC: Format Code, the value decides the order of NB, RB, LST and FLAG;
LLST: Length of LST, number bytes of LST;
NB: No. of Blocks, dividing the file to NB blocks;
RB: Rotated Byte, rotating RB bytes of the block;
LST: Left Shift Table, each byte has two half bytes, the value of half byte is between 0 and 7;
FLAG: Flag, if FLAG is true then rotates left first else rotates right first;
FC and LLST are in the fixed position. The length of ED (LED) is LLST + 5.

(4) Verification bytes (VBs). We use VC to produce VBs as shown in Table 7.
Let EPL = EP + LVET. If EPL + 1 + (NUM – 1)*OFFSET > LM change NUM or OFFSET as shown in Table 5.

Table 7. Verification bytes (VBs)

Address	EPL + 1	EPL + 1 + OFFSET	...	EPL + 1 + (NUM – 1)*OFFSET
VBs	CODE	CODE + CODEI	...	CODE + (NUM – 1)*CODEI

(5) Verification encryption table (VET) stored as shown in Table 8. LVET is in Table 2 and OFFSET is in Table 5. VET is stored in encrypted message data.

Table 8. Verification encryption table (VET) stored

Address	EP	EP + 1	...	EP + (LVET – 1)
Bytes Stored	VET1	VET2	...	VET (LVET)

D. Programs.

(1) Encry-encryption program: The encrypted encryption executing program has fixed VET and VB to do decryption processing.
(2) Encry-decryption program: The encrypted decryption executing program has fixed VET and VB to do decryption processing.

E. Operations.
The methods of processing message have the following operations:

(1) Build sender-receiver user information database operation;
(2) Build encrypted message database operation;
(3) Encryption operation. Following encryption algorithm (MDEC), we encrypt message to encrypted message file;
(4) Produce verification operation. From verification code produce verification bytes and insert verification bytes to encrypted message;
(5) Verification operation. Following verification algorithm (MDVC), we check the checksum and verification bytes;
(6) Decryption operation. Following decryption algorithm (MDDC), we decrypt the encrypted message to original message.

3. **The Proposed Model.** We present the processing message model (PMM) as shown in Figure 1. We send message from sender to receiver via server. The processes of each site are as shown in Figure 1.

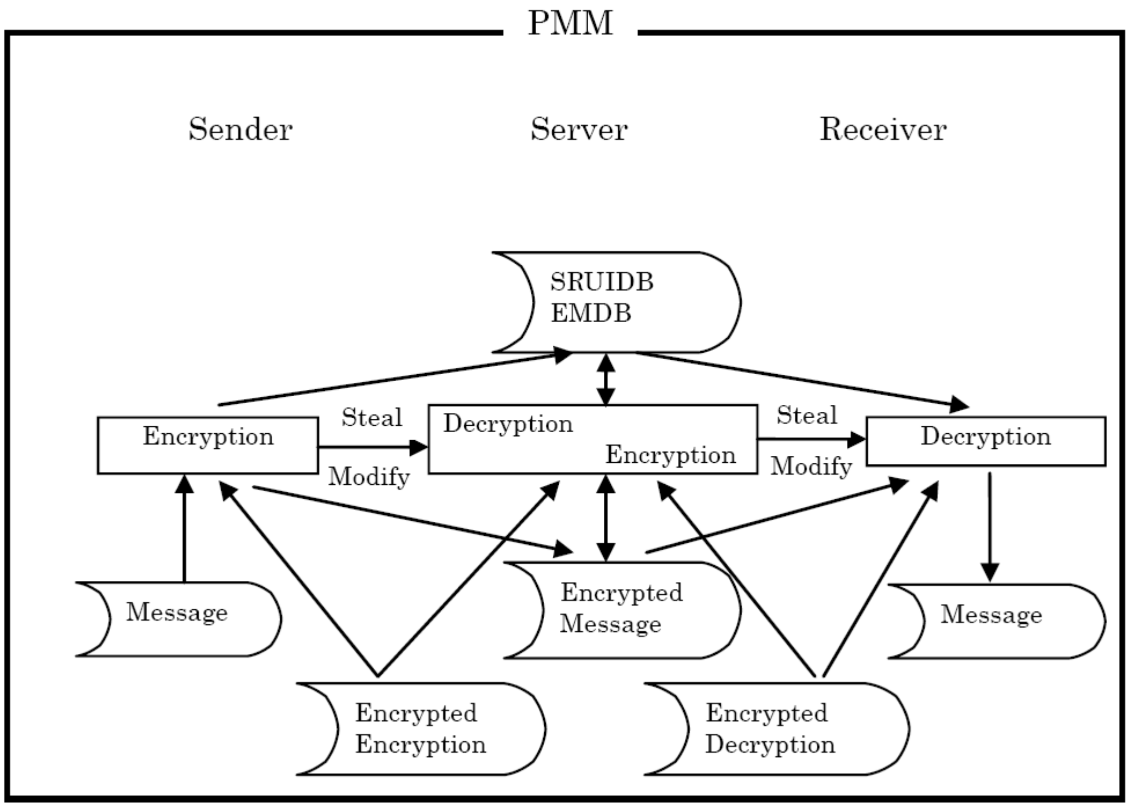

Figure 1. Framework of the proposed processing message model (PMM)

A. Sender.

(1) Get encryption program module:
Get encry-encryption program;
Decrypt encry-encryption program to get encryption program and verification

code;
We use verification code to check encryption program whether it is correct.

(2) Set encrypted-message database module:
Store encrypted message filename (EMFN), sender-id (SENDER-ID), receiver-id (RECEIVER-ID) to EMDB.

(3) Set sender-receiver user information database (SRUIDB) module:
Set sender-id (SENDER-ID), sender location point (SLP), length of verification encryption table (LVET), receiver-id (RECEIVER-ID) and receiver location point (RLP) to SRUIDB.

(4) Get encrypted-message module:
Get message;
Use encryption program to encrypt message to produce encrypted-message;
Set verification code;
We use verification code to produce verification bytes and insert verification bytes to encrypted-message;
Set CKSUM to EMDB;
We send encrypted-message to server.

B. Server.

(1) Get decryption program module:
Get encry-decryption program;
Decrypt encry-decryption program to get decryption program and verification code;
We use verification code to check decryption program whether it is correct.

(2) Get encrypted-message database module:
From EMDB, get sender-id (SENDER-ID), receiver-id (RECEIVER-ID).

(3) Get sender-receiver user information database (SRUIDB) module;
Get sender location point (SLP), length of verification encryption table (LVET), from SRUIDB.

(4) Run decrypt module:
Use SLP and LVET;
Run decryption program to decrypt encrypted-message to the original sending message;
We use verification code to check the sending message whether it is correct.

(5) Get encryption program module:
Get encry-encryption program;
Decrypt encry-encryption program to get encryption program and verification code;
We use verification code to check encryption program whether it is correct.

(6) Run encryption module:
Use encryption program encrypt message to produce encrypted-message;
Set verification code;
We use verification code to produce verification bytes and insert verification bytes to encrypted-message;
Set CKSUM to EMDB;
We send encrypted message to receiver.

C. Receiver.

(1) Get decryption program module:
Get encry-decryption program;
Decrypt encry-decryption program to get decryption program and verification code;
We use verification code to check decryption program whether it is correct.

(2) Get encrypted-message database module:
From EMDB, get sender-id (SENDER-ID), receiver-id (RECEIVER-ID).
(3) Get sender-receiver user information database (SRUIDB) module:
Get receiver location point (RLP), length of verification encryption table (LVET), from SRUIDB.
(4) Run decrypt module:
Use RLP and LVET;
Run decryption program to decrypt encrypted-message to original message;
We use verification code to check the sending message whether it is correct;
We get original message.

4. **Algorithms and Relative Description.**

A. Encryption algorithm (MDEC, message data encryption component).
Based on Lee et al. [2], we proposed the encryption algorithm as the following steps:
(1) Get message file: The length of message is LM;
(2) Set location point (LP): Set SLP and RLP to Table 2;
(3) Set verification encryption table (VET): Set VET of Table 4;
(4) Insert dummy data (LD): Let LM + LD be divided to NB blocks;
(5) Left shift each byte: Use LST in Table 6 to left shift each byte of each block;
(6) Rotate the data: Use RB in Table 6 to rotate each byte of each block;
(7) Change the order of message data: If FLAG is true in Table 6, then we change direction of message;
(8) Create encrypted message data: Insert VET and verification bytes (VB) as Table 1.

B. Verification algorithm (MDVC, message data verification component).
The steps of verification algorithm are as follows:
(1) Get encrypted message data checksum (CSK) and CKSUM from EMDB;
(2) Compare checksum: If CSK and CKSUM are not the same then it is error and exit;
(3) Get verification encryption table (VET):
(a) Get LM from EMDB and LP from SRUIDB, compute entry point (EP);
(b) Extract verification encryption table (VET) from EP of encrypted message data;
(c) Extract verification bytes from verification code;
(d) Produce verification table from verification code.
(4) Compare verification bytes and verification table:
If verification bytes and verification table are not the same then it is error.

C. Decryption algorithm (MDDC, message data decryption component).
Decryption algorithm is the reverse of encryption algorithm. The steps of decryption are as follows:
(1) Extract verification encryption table (VET) and verification bytes (VB);
(2) Get the fields of ED;
(3) Change the order: If FLAG is true, we change the direction of encrypted message data;
(4) Divide blocks: Divide the encrypted message data to NB blocks;
(5) Rotate the encrypted message data: Each block uses RB to rotate different direction;
(6) Left shift each byte: Get LST. According to LST, we left shift each byte as 8-LST bits;
(7) The first LM bytes are the original sending message data.

D. Format code.
The fields in ED have set FC, LLST, NB, RB, LST and FLAG. The FC and LLST are in the fixed portion. The value of FC determinates the order of NB, RB, LST and FLAG. Store the order of FC = 1 to program. The different format of ED can be derived from the value of FC.

E. Compute entry point (EP).
From the location point (LP) in SRUIDB and length of message data (LM) in EMDB, the value EP can be computed as the following rules:
(1) If $LM > LP$ then $EP = LP$;
(2) If $LM < = LP$ then $EP = \mathrm{mod}\ (LP/LM)$.
The EP is the location to insert verification bytes and verification encryption table.

F. Encrypted message data file.
Encrypted message data is produced by the following data:
(1) Message data;
(2) The dummy data;
(3) Location point of sender and receiver;
(4) Encryption verification table;
(5) Verification bytes.
If the above data is more complicated, it will be more difficult to do cryptanalysis.

5. **Conclusions.** In this study, each message data sets different verification encryption tables to do encryption, verification and decryption and is stored in encrypted message data. The verification encryption table contains verification code and encryption data. We use verification code to produce verification bytes and insert to encrypted message data. If the verification bytes or checksum of file is changed, the encrypted message data is modified to be error. If same message data runs many times to encrypt, we can get different encrypted message data. We make the following suggestions.

A. All the important executed programs are encrypted to store in the system. They can protect against being modified or stolen.
B. All the messages are encrypted before sending out to protect against being modified or stolen.
C. We store source program to backup devices.
D. When the source program is updated, we can do the following:
(1) Using source program in the backup device to be updated and restored, we produce executed program and then encrypt to be encrypted program. We replace old encrypted program and delete execute program.
(2) We store the encrypted source program in the running disk and use it to update.

REFERENCES

[1] H.-S. Chen, T.-Y. Lee and H.-M. Lee, Verification of stored security data in computer system, *ACIIDS 2010, Part I, LNCS*, vol.5990, pp.426-434, 2010.
[2] H.-M. Lee, T.-Y. Lee, L. Lin and J.-S. Su, Cipher text containing data and key to be transmitted in network security, *Proc. of the 11th WSEAS International Conference on System*, Agios Nikolaos, Crete Island, Greece, pp.275-279, 2007.
[3] L. Lin, H.-M. Lee, J.-N. Chen and T.-Y. Lee, Processing security data in computer system, *Information Engineering Letters*, vol.3, pp.26-34, 2013.
[4] L. Lin, H.-M. Lee and T.-Y. Lee, The processing method of security data in computer system, *ICIC Express Letters, Part B: Applications*, vol.7, no.3, pp.583-589, 2016.
[5] M. Matsui, Linear cryptanalysis method for DES cipher, *Advances in Cryptology – EUROCRYPT'93*, no.765, pp.386-397, 1994.
[6] R. J. McEliece, A public-key system based on algebraic coding theory, *Deep Space Network Progress Report*, Jet Propulsion Laboratory, California Institute of Technology, vol.44, pp.114-116, 1978.
[7] R. C. Merkle, One way hash function and DES, *Advances in Cryptology – CRYPTO'89 Proceedings*, pp.428-446, 1990.

[8] S. Miyaguchi, The FEAL-8 cryptosystem and call for attack, *Advances in Cryptology – CRYPTO'89 Proceedings*, pp.624-627, 1990.

[9] National Bureau of Standards, *NBS FIPS PUB 46: Data Encryption Standard, National Bureau of Standards, USA Department of Commerce*, 1977.

[10] National Institute of Standards and Technology (NIST), *FIPS PUB 180: Secure Hash Standard (SHS)*, 1993.

[11] W. Stallings, *Cryptography and Network Security: Principles and Practices*, International Edition, Third Edition by Pearson Education, Inc. Upper Saddle River, NJ 07458, 2003.

[12] C. E. Shannon, Communication theory of security systems, *Bell System Technical Journal*, vol.28, pp.657-715, 1949.