ICIC Express Letters
Part B: Applications
Volume **7**, Number **3**, March **2016**

ICIC International ©2016 ISSN 2185-2766
pp. **583–589**

THE PROCESSING METHOD OF SECURITY DATA 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 August 2015; accepted October 2015

Abstract. *The security data in the computer system must be processed correctly and kept security to protect against stealing or modifying. In this study, each security data has set different verification encryption table. We 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 security data and verification code to produce verification bytes and then we insert them to the encrypted security data. When the security data are processed, it should decrypt the encrypted security data and extract the verification bytes to verify first. If verification bytes are changed, the file is not correct. In this paper, we present the processing method of encryption, verification and decryption algorithm to protect the security data.*
Keywords: Cipher text, Decryption, Encryption, Security data, Verification

1. **Introduction.** Shannon [11] discussed the theory of security system in 1949. The functions of security system are security, authenticity, integrity, non-repudiation, data confidentiality and accessed control [10]. NBS (National Bureau of Standards, U.S.A) [8] proposed data encryption standard (DES) in 1977. McEliece [5] used algebraic coding theory to propose public key. Merkle [6] presented "One way hash function" and used for digital signature. Miyaguchi [7] developed fast data enciphered algorithm (FEAL-8) in 1988. NIST (National Institute of Standards and Technology) [9] presented secure hash standard (SHS). Matsui [4] 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] also presented encryption and verification to process security data in computer system.

In this paper, Section 2 is the proposed method description. We present the proposed model in Section 3. Section 4 shows the algorithms and relative description. We make the conclusions in Section 5.

2. **The Proposed Method Description.** The proposed method is to process the security data in security. Each security data has different verification encryption table. The verification encryption table contains encryption data and verification code. We use

encryption data to encrypt security data to produce encrypted security data and use verification code to produce verification bytes and then insert them to encrypted security data. Each security data has location point and is stored in security information database. We use the location point and the length of security data to compute the entry point. We insert verification bytes and verification encryption table to the entry point of encrypted security data. When we want to get the security data, we should get the length of this file and location point in the security information database. 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 their correction, use encryption data to decrypt the remaining encrypted security data to get the original security data to process.

We explain files, database, tables and processes as follows:

A. Files

(1) Security data file.

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

Table 1. Encrypted security data

CT	VET	VB	CT	VET	VB	CT	...	VET	VB	CT	VB	CT

B. Database

(1) Security information database (SIDB). SIDB contains security file name (SFN), user-id (USER-ID), location point (LP), offset (OFFSET), length of verification encryption table (LVET) and length of security data file (LS) as Table 2. SFN and USER-ID are used as keys.

Table 2. Security information database (SIDB)

SFN	USER-ID	LP	OFFSET	LVET	LS

(2) Encrypted security database (ESDB). ESDB contains encrypted security file name (ESFN), user-id (USER-ID), directory (DIR) and checksum (CKSUM) as Table 3. We use ESFN and USER-ID as keys.

Table 3. Encrypted security database (ESDB)

ESFN	USER-ID	DIR	CKSUM

C. Tables

(1) Verification encryption table (VET). VET contains Verification Code (VC) and Encryption Data (ED) as 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), offset (OFFSET1) and number of inserted bytes (NUM) as Table 5.

(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 Table 6.

Table 5. Verification code (VC)

CODE	CODEI	OFFSET1	NUM

Table 6. Encryption data (ED)

FC	LLST	NB	RB	LST	FLAG

In Table 6
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, and the value of half byte is between 0 and 7;
FLAG: Flag, if FLAG is set then rotate left first; else rotate 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 Table 7.

Table 7. Verification bytes (VBs)

Address	EP	EP + OFFSET1	...	EP + (NUM − 1) ∗ OFFSET1
VBs	CODE	CODE + CODEI	...	CODE + (NUM − 1) ∗ CODE1

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

Table 8. Verification encryption table (VET) stored

Address	EP	EP + OFFSET	...	EP + (LVET − 1) ∗ OFFSET
Bytes Stored	VET1	VET2	...	VET (LVET)

D. Processes
The processes of security data file have the following operations.
(1) Store operation. Store the original security data file to private device and encrypt it to encrypted security data file, and then store encrypted security data file in computer system.
(2) Verification operation. Check the checksum and verification bytes.
(3) Update operation. Verify the encrypted security data file. Decrypt the encrypted security data file. Update the security data file. Backup the security data file. Encrypt the security data to encrypted security data file and store in computer system.

3. **The Proposed Model.** We present the process security data model (PSDM) as Figure 1. It contains process store security data module (PSSDM), process verification security data module (PVSDM) and process update security data module (PUSDM).
A. Process store security data module (PSSDM). PSSDM has the following components.
(1) Backup security data component (BSDC). It does backup to private device.
(2) Create security information database (SIDB) component (CSIDBC). It creates SIDB. Use SFN and USER-ID as keys to create an entry in SIDB and set location point (LP), offset (OFFSET) and length of security data file (LS) to SIDB.
(3) Create verification encryption table component (CVETC). It sets different verification encryption table (VET) as Table 4. Store the length of LVET to SIDB.

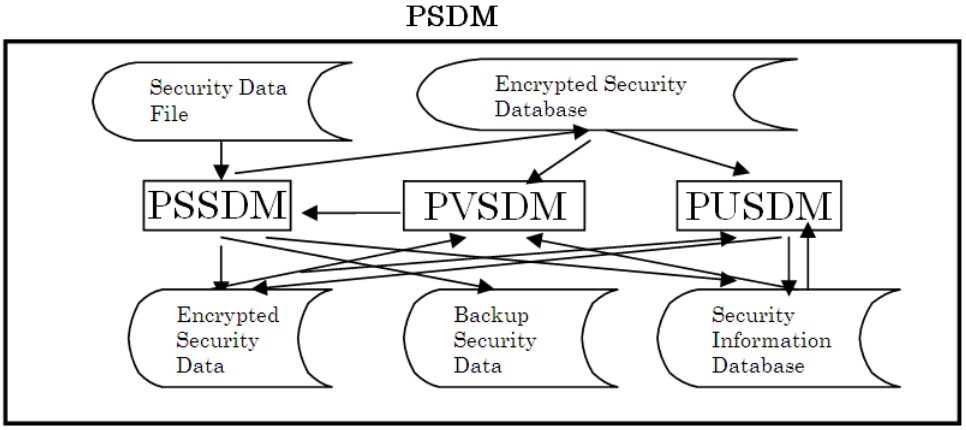

FIGURE 1. Framework of the proposed PSDM

(4) Encrypt security data file component (ESDFC). It follows encryption algorithm (SDEC), uses ED to encrypt the security data file to encrypted security data and uses VC to produce verification bytes.

(5) Insert verification bytes and verification encryption table component (IVBVETC). It inserts verification bytes and then inserts verification encryption table to encrypted security data.

(6) Store encrypted security database component (SESDBC). It stores encrypted security data to create ESDB, gets checksum of encrypted security data and stores checksum to CKSUM of encrypted security database (ESDB).

B. Process verification security data module (PVSDM). PVSDM has the following components:

(1) Check checksum encrypted security data component (CCESDC).
  (a) Get the encrypted security data file and checksum (CSK);
  (b) Use USER-ID and SFN as keys to get LP, OFFSET, LVET and LS from SIDB;
  (c) Use ESFN and USER-ID as keys to get CKSUM from ESDB;
  (d) If CSK and CKSUM are not the same, it is error and exits.

(2) Verify encrypted security data component (VESDC). VESDC follows verification algorithm (SDVC) to verify encrypted security data to be correct or not.
  (a) From LP and LS to compute entry point (EP);.
  (b) From EP, extract verification encryption table and verification bytes (VB);
  (c) Use verification code to build verification table (VT).
  (d) If VB and VT are the same
    then the encrypted security data is correct and exits,
    else processes the following steps:
      1) Delete the encrypted security data;
      2) Load the security data from private device;
        If PVSDM is called by PUSDM then return;
      3) Process store security data module (PSSDM) again.

C. Process update security data module (PUSDM). PUSDM has the following components:

(1) Process PVSDM to check the encrypted security data correct or not.
  If it is error then go to (4);

(2) Extract verification encryption table and verification bytes component (EVETVBC). It extracts verification encryption table (VET) and verification bytes from encrypted security data file. Verification encryption table contains encryption data and verification code;

(3) Decrypt encrypted security data component (DESDC). DESDC follows decryption algorithm (SDDC) and uses encryption data to decrypt the encrypted security data. Get LS from SIDB. The first LS bytes are the original security data;

(4) Process update security data component (PUSDC). PUSDC processes to update security data;

(5) Process security data store component (PSDSC). PSDSC processes to backup and processes PSSDM again.

4. **Algorithms and Relative Description.**

A. Encryption algorithm (SDEC security data encryption component)

Based on Lee et al. [2], we proposed the encryption algorithm as the following steps.

(1) Get file. Get security data file and its length LS;

(2) Set location point. Set location point of the user and security data and number of blocks (NB). NB saves to encryption data (ED);

(3) Insert dummy data;

(a) Insert any characters to trailer of security data file, the length of security data is multiplier of NB and produce temp file to process;

(b) Divide temp file to NB blocks.

(4) Set verification encryption table (VET);

(a) Set value of code (CODE), code increase (CODEI), offset (OFFSET) and number (NUM) to the fields of verification code (VC);

(b) Set value of FC, LLST, RB, LST and FLAG to the fields of encryption data (ED).

(5) Left shift each byte;

(a) Get left shift table (LST) and LLST from ED;

(b) The length of LST is LLST;

(c) From the first block to NB blocks;

```
Repeat
    Get first byte of LST.
    From first byte to end byte of block;
    Repeat
        Get the shift value SV of each half byte of LST.
        Left shift each next byte of block SV bits;
        Get next half byte of LST;
        If end of LST, then reset to first LST
    Until end of block.
    Get next block
Until over NB block.
```

(6) Rotate the security data;

(a) Get rotated byte RB from ED;

(b) From the beginning block, repeat to rotate each block left or right RB bytes depending on the sign of RB.

(7) Change the order of data;

If FLAG was set then change the order of temp file.

(8) Create encrypted security data;

(a) Produce verification bytes (VB) from verification code;

(b) Compute entry point (EP) from location point (LP) of SIDB;

(c) Insert VB and EVT into temp file from EP;

(d) Get the final encrypted security data from temp file.

B. Verification Algorithm (SDVC security data verification component)

The steps of verification algorithm are as follows:

(1) Get encrypted security data file and checksum (CSK) and get CKSUM from ESDB;

(2) Compare checksum. If CSK and CKSUM are not the same, it is error and exit;
(3) Get verification encryption table (VET);
  (a) Get LS and LP from SIDB, compute entry point (EP);
  (b) Extract verification encryption table (VET) from EP of encrypted security 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 (SDDC security 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);
  (a) Compute entry point (EP). From EP, extract VET;
  (b) VET contains verification code (VC) and encryption data (ED). Use VC to extract VB.
(2) Get the fields of ED. From FC of ED, we get LLST, NB, RB, LST and FLAG;
(3) Change the order. If FLAG is true, then change the order of data;
(4) Divide blocks. Divide input string to NB blocks;
(5) Rotate the security data;
  (a) Get rotated byte RB from ED;
  (b) From the beginning block, repeat to rotate each block right or left RB bytes depending on the sign of RB.
(6) Left shift each byte;
  (a) Get left shift table (LST) and LLST from EDT;
  (b) The length of LST is LLST;
  (c) From the first block to NB blocks,

```
Repeat
    Get first byte of LST.
    From first byte to end byte of block;
    Repeat
        Get the shift value SV of each half byte of LST;
        Set SV = SV + 6;
        Left shift each next byte of block SV bits;
        Get next half byte of LST;
        If end of LST, then reset to first LST;
    Until end of block.
    Get next block;
Until over NB block.
```

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) and length of security data (LS) in security information database (SIDB), the value EP can be computed as the following rules:

(1) If LS > LP then EP = LP;
(2) If LS <= LP then EP = mod (LP/LS).

The EP is the location to insert verification bytes and verification encryption table.

```
Get OFFSET1 and NUM from VC.
If EP + 2 + (NUM − 1) ∗ OFFSET1 > LS then
    Repeat
```

reset NUM and OFFSET1 of VC to smaller value
Until EP + 2 + (NUM − 1) ∗ OFFSET1 <= LS.
Get OFFSET and LVET from SIDB.
If EP + 2 + (LVET − 1) ∗ OFFSET > LS then
Repeat
reset OFFSET of SIDB to smaller value.
Until EP + 2 + (LVET − 1) ∗ OFFSET <= LS.

F. Encrypted security data file

Encrypted security data is produced by the following data:
(1) Security data;
(2) The dummy data;
(3) Location point of user and security data;
(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 security data sets different verification encryption table to do encryption, verification and decryption, and is stored in encrypted security data. The verification encryption table contains verification code and encryption data. We use verification code to produce verification bytes and insert to encrypted security data. When the verification bytes or checksum of file are changed, the encrypted security data is modified and error. When same security data runs many times to encrypt, we can get different encrypted security data. When security data file contains many records, we can process one record per time and store verification encryption table in security information database.

Acknowledgment. The authors gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] H.-S. Chen, T.-Y. Lee and H.-M. Lee, Verification of stored security data in computer system, *The 2nd International Conference on Intelligent Information and Database Systems*, 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] M. Matsui, Linear cryptanalysis method for DES cipher, in *Advances in Cryptology*, T. Helleseth (ed.), New York, Springer-Verlag Berlin Heidelberg, 1994.
[5] R. J. McEliece, A public-key system based on algebraic coding theory, *Deep Space Network Progress Report*, California Institute of Technology, pp.114-116, 1978.
[6] R. C. Merkle, One way hash function and DES, in *Advances in Cryptology*, G. Brassard (ed.), New York, Springer-Verlag Berlin Heidelberg, 1990.
[7] S. Miyaguchi, The FEAL-8 cryptosystem and a call for attack, in *Advances in Cryptology*, G. Brassard (ed.), New York, Springer-Verlag Berlin Heidelberg, 1990.
[8] National Bureau of Standards, *Data Encryption Standard*, U. S. A. Department of Commerce, 1977.
[9] National Institute of Standards and Technology (NIST), *Secure Hash Standard (SHS)*, 1993.
[10] W. Stallings, Cryptography and network security: Principles and practices, *International Journal of Engineering & Computer Science*, vol.11, no.7, pp.655-660, 2003.
[11] C. E. Shannon, Communication theory of security systems, *Bell System Technical Journal*, vol.28, pp.657-715, 1949.