Data Transmission Encryption and Decryption Algorithm in Network Security

TSANG-YEAN LEE, HUEY-MING LEE, HOMER WU, JIN-SHIEH SU
Department of Information Management, Chinese Culture University
55, Hwa-Kung Road, Yang-Ming-San, Taipei (11114), TAIWAN

Abstract: - In this paper, we proposed the encryption algorithm to encrypt plaintext to cipher text and the decryption algorithm to do the reverse. We applied the basic computing operations to design these algorithms in this study. We used the inserting dummy symbols, rotating, transposition, shifting, complement and inserting control byte to build the data and tables in the encryption algorithm. These operations are simple and easily to implement. Without knowing the data and tables of encryption, it is difficult to do cryptanalysis. In the decryption algorithm, we used these data and tables to decrypt cipher text to plaintext. We can easily apply these algorithms to transmit data in network and the data transmission is secure.

Key-Words: - Encryption; Decryption; Data transmission; Cipher text; Plaintext; Network security

1 Introduction

In 1949, Shannon [19] discussed the theory of security system. In general, the functions of security system are security, authenticity, integrity and non-repudiation [1-3]. Stallings [21-22] increased data confidentiality and accessed control. Traditional encryption operations are transposition and substitution. These basic operations solve complex encryption problems. But, the problems are easily to do cryptanalysis. Diffie and Hellman [5] proposed the concept of pubic key. Rivest *et al* [18] proposed public cryptosystem. From 1960, IBM started the planning of computer cryptology research, and in 1974, IBM proposed an algorithm to review. In 1977, NBS (National Bureau of Standards, U.S.A.) [13-14] suggested this algorithm as data encryption standard (DES). McEliece [10] used algebraic coding theory to propose public key. Merkle [11] presented “One way hash function” and used for digital signature. Miyaguchi [12] developed fast data encipherment algorithm (FEAL-8). NIST (National Institute of Standards and Technology) [15-16] proposed secure hash standard (SHS). Biham and Shamir [1-3] proposed differential attack, Matsui [9] proposed linear cryptanalysis to attack DES type security system. When new encryption is proposed, cryptanalysis starts to develop to attack.

Stallings [21-22] presented the model of encryption and decryption as shown in Fig. 1.

In this paper, we proposed the encryption and decryption algorithm in Section 2 and Section 3

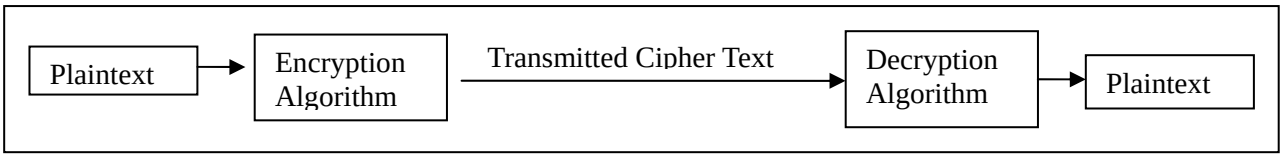

Fig. 1 Simplified model of convention encryption [21-22]

respectively. The decryption is the reverse of encryption. We used insertion, rotation, transposition, shift, complement and pack of computer operation in these two algorithms of this paper. We implemented these algorithms with C language and got the result of processing time.

2. Encryption algorithm

The steps of encryption algorithm are in the following steps:

Step 1: Set the number

The plaintext has the following combinations.

(1) all numerical characters;

(2) all alphabetic characters;

(3) numerical and alphabetic mixed characters together.

If the plaintext is all numeric characters, we can pack and decrease the length of text. If the plaintext is numeric and alphabetic mixed together, we can pack numeric characters first. Finally, we let the length of the plaintext be N characters and store them in the symbol table (ST) as follows:

Symbol Tables (ST): $S_1S_2\ldots S_N$

Step 2: Set dummy symbol

We can use any function, e.g. date and time, to pack dummy symbol table (DST) with *dddhhmmss* and length 9 as follows:

Dummy Symbol Table (DST): $D_1D_2\ldots D_9$

Step 3: Combine symbol table and dummy symbol table to symbol table with dummy (STWD)

Get any M (<=9) characters from dummy symbol table (DST).

Combine symbol table and above M characters to symbol table with dummy (STWD) as follows:

Symbol Table with Dummy (STWD):

$$S_1S_2\ldots S_ND_1D_2\ldots D_M$$

Step 4: Set rotated byte and rotate symbol table with dummy

Set rotated byte, RB, as RB = D_9 mode (N+M).

If RB is odd, we rotate symbol table with dummy to left RB times. If RB is even, we rotate symbol table with dummy to right RB times. Insert RB to the trailer of above symbol table after rotation. Get symbol table after rotation (STAR).

For example: if RB=5 then we have symbol table after rotation (STAR) as follows:

$$S_6\ldots S_ND_1D_2\ldots D_M\ S_1S_2\ S_3S_4\ S_5RB$$

Step 5: Transpose the symbol table after rotation

Set the position table (PT) as follows:

Position Table (PT): $P_1P_2\ldots P_{N+M+1}$

Following position table (PT), we change the location of the symbol table after rotation.

Get the symbol table after transposition.

Symbol Table after Transposition (STAT):

$$SP_1SP_2\ldots SP_{N+M+1}$$

Step 6: Shift the symbol table after transposition

Set shift left table (SLT) of each byte, the contained value of shift left table is below 8, as shown below: Shift Left Table (SLT):

$$F_1F_2\ldots F_{N+M+1}$$

Shift each byte of symbol table after transposition (STAT) according to the contained value of shift left table (SLT). Get symbol table after shift (STAS) as follows:

Symbol Table After shift (STAS): $SS_1SS_2 \ldots SS_{N+M+1}$

Step 7: Complement the symbol table after shift

Set control bit table (CBIT) to all 0 and byte length is L= [(N+M)/8+1]. If the value of symbol table after shift is below the certain value (ex. 20_{16}), we complement the symbol of symbol table after shift and set the relative bit of control bit to 1. The result of control bit table is as follows:

Control Bit Table (CBIT): $C_1C_2 \ldots C_L$

Get symbol table after complement (STAC) as follows:

Symbol Table After Complement (STAC):

$SS_1SC_2 \ldots SS_{N+M+1}$

Step 8: Packed control byte table

To form control byte table (CBT), we take each 7 bits of control bit table (CBIT) from left and set bit 0 to 1. The length of control byte table is K=[(N+M)/7] +1. Control Byte Table is as follows:

Control Byte Table (CBT): $(CB1_1)(CB1_2)\ (CB1_K)$

Step 9: Shift the control byte table

Set control byte shift table (CBST) as follows:

Control byte shift table (CBST): $CF_1CF_{2,\ldots}CF_K$

Shift left of each byte of control byte table (CBT) according to control byte shift table (CBST). Get the Control Byte after shift (CBAS) as follows:

Control Byte after shift (CBAS): $CS_1CS_2 \ldots CS_K$

Step 10: Combine symbol table after complement and control byte after shift to get cipher text (CT)

Set control byte position table (CBPT).

Control Byte Position Table (CBPT):

$CP_1CP_2 \ldots CP_K$

Insert each control byte after shift into symbol table after complement (STAC) according to control byte position table (CBPT). Get the cipher text (CT).

Cipher text (CT): $SS_1\ SC_2\ CS_1 \ldots SS_{N+M+1+K}$

3. Decryption algorithm

The steps of decryption algorithm are as follows:

Step 1: Get the cipher text (CT)

We get the cipher text (CT) as follows:

Cipher text (CT): $C_1C_2 \ldots C_{N+M+1+K}$

Step 2: Separate cipher text into control byte after separation (CBAS) and symbol table after separation (STAS)

Get control byte position table (CBPT)

Control Byte Position Table (CBPT):

$CP_1\ CP_2 \ldots CP_K$

Following control byte position table (CBPT), we separate cipher text (CT) into control byte after separation (CBAS) and symbol table after separation (STAS) as follows: Control Byte After Separation

(CBAS): $CS_1CS_2 \ldots CS_K$

Symbol Table After Separation (STAS):

$SS_1SS_2 \ldots SS_{N+M+1}$

Step 3: Shift control byte after separation

Get control byte shift table (CBST).

Set the value of shift table of each byte as 8 minus control byte shift table (CBST) as follows:

Control Byte Shift Table (CBST): $CF_1CF_2 \ldots CF_K$

We left shift each byte of control byte after separation (CBAS) according to above shift table. Get control byte after shift (CBAS) as follows: Control Byte After Shift (CBAS): $CB_1CB_2 \ldots CB_K$

Step 4: Pack control byte after shift

We retrieve 7 bits from each control byte after shift (CBAS). We pack above bits to form control bit table (CBIT).

Control Bit Table (CBIT): $CI_1CI_2 \ldots CI_L$

Step 5: Complement symbol table after separation

From the relative bit position of control bit table

(CBIT), if bit =1 then we complement the symbol of symbol table after separation (STAS) else no change.

Get the symbol table after complement (STAC) as follows:

Symbol Table After Complement (STAC):

$SC_1SC_2 \ldots SC_{N+M+1}$

Step 6: Shift symbol table after complement

Get shift left table (SLT).

Set each byte of shift table as 8 minus shift left table (SLT) as follows:

Shift Left Table (SLT): $F_1 F_2 \ldots F_{N+M+1}$

We left shift each byte of the symbol table after complement (STAC) according the above shift table.

Get symbol table after shift (STAS) as follows:

Symbol Table After Shift (STAS): $ST_1ST_2 \ldots ST_{N+M+1}$

Step 7: Transpose the symbol table after shift

Get the position table (PT).

Position Table (PT): $P_1P_2 \ldots P_{N+M+1.}$

According to the position table, we transpose the symbol table after shift (STAS).

Get symbol table after transposition (STAT) as follows:

Symbol Table After Transposition (STAT):

$SP_1SP_2 \ldots SP_{M+N+1}$

Step 8: Rotate symbol table after transposition

Get the rotated value from trailer byte of STAT and call it as RB. If RB is odd, rotate symbol table after transposition without trailer byte to right RB times. If RB is even, rotate symbol table after transposition without trailer byte to left RB times.

Get symbol table after rotation (STAR) as follows:

Symbol Table After Rotation (STAR):

$SR_1SR_2 \ldots SR_{N+M+1}$

Step 9: Get plaintext (PT).

Get the plaintext from first N symbols of symbol table after rotation (STAR) as follows:

Plaintext (PT) $S_1S_2 \ldots S_N$

4. Implementation

In this section, we implemented the proposed algorithms in Section 4.1 - 4.2, and presented the discussion of implementation in Section 4.3.

4.1 Computing environment

Computer: type: INTEL, Pentium D830

Memory size: DDR 512 MB * 2

Computer Language: C Language

4.2 Executing results

The processing time of the difference combinations of symbol size and executing times are as follows: Table 1 is the encryption processing time and Table 2 is the decryption processing time

Table 1 Encryption processing time

Encryption	Symbol table: size(Bytes)			
Times[*1]	8	16	24	32
1M	9.75[*2]	12.19	13.98	15.98
4M	36.98	49.33	56.39	64.31
8M	79.20	98.36	112.94	128.64
16M	168.84	192.42	226.70	256.97

[*1]M=1000000 processing times.

[*2] processing time in second

Table 2 Decryption processing time

Decryption	Symbol table: size(Bytes)			
Times*	8	16	24	32
1M	4.73	7.06	9.47	11.95
4M	18.77	28.73	38.30	47.63
8M	38.55	56.67	75.89	95.47
16M	77.14	114.14	151.69	192.70

[*1]M=1000000 processing times.

[*2] processing time in second

4.3 Discussion of implementation

(1) Encryption takes more time because it calls DATE and TIME function to produce dummy symbols. We can choose the other function to replace.

(2) Number of symbols increases, time is linearly increases.

(3) Number of executing times increases, time linearly increases.

5. Conclusion and Discussion

In this study, we used the basic computing operations to design these encryption and decryption algorithms. We don't need any special hardware. Finally, we made some comments about this study.

(1) To do the decryption, we should have the following tables and values to decrypt.

Position table (PT)
Shift Left Table (SLT)
Control Byte Sift Table (CBST)
Control Byte Position Table (CBPT)
Length of plaintext to encryption
Length of dummy symbols

(2) The length of cipher text is the length of plaintext and dummy symbols and control bytes (about (plaintext + dummy symbol)/8)

(3) The same plaintext may encrypt to different cipher text, when the transfer time is different.

(4) If plaintext acts as key, we must decrypt the received cipher text and check the key whether existed or not.

(5) The reasons of difficult cryptanalysis are as follow;:

(a)Through complement, we can avoid control codes of transmission.

(b)Through rotation, when plaintext is on sequence numbers, it may be different position and difficult to process cryptanalysis.

(c) Through left shift, the content has changed.

(d) Insert dummy symbols, we'll be difficult to determine plaintext.

(6) Using basic operations, we don't need complex computation.

(7) By the algorithms described in Section 2, and Section 3, we can set up the encryption and decryption mechanism by computers as a useful and security procedure.

Acknowledgements

This work was supported in part by the National Science Council, Republic of China, under grant NSC 95-2745-M-034-007-URD.

References

[1] E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystem", Advances in Cryptology-CRYPTO'90 Proceedings, Berlin: Springer-Verlag, pp. 2-21, 1991

[2] E. Biham and A. Shamir, "A Differential cryptanalysis of the Data Encryption Standard", Springer, Berlin Heidelberg New York, 1993

[3] E. Biham and A. Shamir, "Differential Cryptanalysis of Data Encryption Standard", Berlin: Springer-Verlag, 1993

[4] D. Denning, "Cryptography and Data Security", Addison-Wesley, 1982

[5] W. Diffie, M. E. Hellman, "New Directions in Cryptography", IEEE Trans. On Inform. Theory, pp. 644-654, 1976.

[6] H. Gilbert and G.. Chase, "A Statistical Attack on the FEAL-8 Cryptosystem," Advances in Cryptology-CRYPTO'90 proceedings, Berln: Springer-Verlag, pp. 22-2, 1991

[7] Oded Goldreich, "Foundations of Cryptography : Basic Tools", Published by

the Press Syndicate of The University of Cambridge, The Pitt Building, Trumpington Street, Cambridge, United Kingdom, First published 2001

[8] Darel W. Hardy, Carol L. Walker, "Applied Algebra: Codes, Ciphers, and Discrete Algorithms", Library of Congress cataloging-in Publication Data, 2003 by Pearson Education, Inc. Pearson Education, Inc. Upper Saddle River, NJ 07458.

[9] M. Matsui, "Linear cryptanalysis method for DES cipher', In T. Helleseth, editor, Advances in Cryptology (CRYPT"O'90). Lecture Notes in Computer Science No. 765, pp. 386-397. Springer, Berlin Heidelberg New York, 1994

[10] R. J. McEliece, "A Public-Key System Based on Algebraic Coding Theory" pages 114-116, Deep Sace Network Progress Report, 44, Jet Propulsion Laboratory, California Institute of Technology, 1978

[11] R. C. Merkle, "One Way Hash Function and DES" Proc. Crypto'89, Berlin: Springer-Verlag, pp. 428-446, 1990

[12] S. Miyaguchi, "The FEAL-8 Cryptosystem and Call for Attack", Advances in Cryptology-CRYPTO'89 proceedings, Belin: Springer Verlag, pp. 624-627, 1990

[13] National Bureau of Standards, NBS FIPS PUB 46, "Data Encryption Standard," National Bureau of Standards, U. S. Department of Commerce, Jan., 1977

[14] National Bureau of Standards, NBS FIPS PUB 81, "Data Modes of Operation," National Bureau of Standards, U. S. Department of Commerce, Jan. 1980

[15] National Institute of Standards and Technology (NIST). FIPS PUB 180: Secure Hash Standard (SHS), May 11, 1993

[16] National Institute of Standards and Technology (NIST). NIST FIPS PUB 185, Escrowed Encryption Standard, February 1994

[17] Josef Pieprzyk, Thomas Hardjono, Jennijer Seberry, "Fundamentals of Computer Security", Springer-Verlag Berlin Heidelberg, 2003

[18] R. L. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public –Key Cryptosystems", Communications of the ACM, Vol. 21, No. 2, pp. 120-126, Feb. 1978

[19] C. E. Shannon.: "Communication Theory of security Systems", Bell System Technical Journal, Vol. 28, pp. 657-715, 1949

[20] A. Shimizu, S. Miyaguchi, "Fast Data Enciphrment Algorithm FEAL", Advances in Cryptology-EUROCRYPT'87 proceedings, Berlin: Springer-Verlag, pp.267-278, 1987

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

[22] William Stallings, "Network Security Essentials: Application and Standards", Second Edition 2003 by Pearson Education, Inc. Upper Saddle River, NJ 07458