Send Orders for Reprints to reprints@benthamscience.ae

The Open Cybernetics & Systemics Journal, **2015**, *9*, **2677-2680**

Open Access

Multiple Binary Trees Encryption Principle

Yong Wei[*]

Software School, Shenzhen Institute of Information Technology, Shenzhen, 518117, P.R. China

Abstract: Through block cipher method, this paper demonstrates using preorder and post traversal sequence of a binary tree with 64 nodes to implement symmetric encryption/decryption. The different traversal sequences may determine a binary tree, so the method is vulnerable to be attacked. The paper's further deduction involves improvement using multiple binary trees, so that more secure and efficient symmetric encryption algorithms can be obtained.

Keywords: AES, binary tree, DES, padding, symmetric encryption.

1. INTRODUCTION

DES and AES are approved by the federal government block cipher standard [1-6]. Despite the repeal, DES is still very popular (3DES deformation is still quite safe). However, AES is the Advanced Encryption Standard, its speed and high security level can replace the DES algorithm.

Rijndael block cipher algorithm is an iterative, its packet length is 128bit, the key length is 128/192/256bit, and the corresponding number of rounds is 10/12/14. In comparison, AES 128bit keys are 1021 times more than 56bit DES keys.

If a binary tree is considered as the key, the binary tree traversal can be used to encrypt [7]. For example, the string "ABCDEFGH" is the plaintext, the binary tree in Fig. (**1a**) is the key. Firstly the string is loaded into the binary tree according to preorder traversal, then the re-export of the tree postorder traversal sequence EDCBGHFA is regarded as cipher text. When decrypting, according to postorder traversal order, the encrypted data is loaded into the key, then the plaintext data is restored from the preorder traversal sequence.

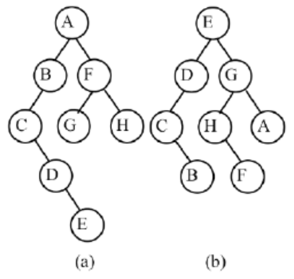

Fig. (1). Binary trees as key.

The number of binaries which consist of n nodes is:

$$\mathrm{B(n)} = \sum_{\mathrm{k=0}}^{\mathrm{n-1}} \mathrm{B(k)B(n-k-1)} \tag{1}$$

As the number of the binary trees consisting of n nodes is sufficiently large, the binary trees as the key for encryption/decryption operations are more than the AES encryption algorithm keys. For example, in the AES algorithm, when the key length is 128, there are 3.4×10^{38} keys. If a binary tree is used as a key when the number of bits n = 128, there are $B(n) = 4.4718285453094634 \times 10^{73}$ keys, which are much higher than the AES key 35 orders of magnitude, and the key exhaustive search is harder.

Binary tree has six kinds of traversal solutions, DLR, LDR, LRD, DRL, RDL and RLD. Whatever the traversal, its time complexity is O(n). So binary encryption/decryption efficiency is made high. Contrastingly, AES is more complex.

As some binary tree traversal sequence uniquely determines the binary tree [8-12], if only one binary tree to encrypt/decrypt is used, it is vulnerable to be attacked. For this reason, it is necessary to use multiple binary trees to achieve more secure and efficient encryption algorithm.

2. BINARY ENCRYPTION/DECRYPTION PROCESS

The binary tree with 64 nodes as shown in Fig. (**2**) is used as the key. Where "n#" indicates the nodes number, which is precisely the preorder sequence of the binary tree. "l"

n#	l	r	n#	l	r	n#	l	r	n#	l	r
00	nl	01	16	nl	17	32	nl	33	48	nl	49
01	02	08	17	18	19	33	nl	34	49	50	53
02	nl	03	18	nl	nl	34	nl	35	50	nl	51
03	04	06	19	nl	20	35	nl	36	51	52	nl
04	nl	05	20	21	27	36	37	39	52	nl	nl
05	nl	nl	21	nl	22	37	nl	38	53	nl	54
06	07	nl	22	23	nl	38	nl	nl	54	55	57
07	nl	nl	23	24	25	39	nl	40	55	nl	56
08	09	10	24	nl	nl	40	41	45	56	nl	nl
09	nl	nl	25	26	nl	41	nl	42	57	58	59
10	11	13	26	nl	nl	42	43	44	58	nl	nl
11	nl	12	27	28	29	43	nl	nl	59	60	nl
12	nl	nl	28	nl	nl	44	nl	nl	60	61	62
13	nl	14	29	nl	30	45	46	48	61	nl	nl
14	15	16	30	nl	31	46	nl	47	62	63	nl
15	nl	nl	31	nl	32	47	nl	nl	63	nl	nl

Fig. (2). Binary tree with 64 nodes.

*Address correspondence to this author at the Software School of Shenzhen Institute of Information Technology, Shenzhen, Guangdong, 518172, P.R. China; Tel: +86 755 89226180; Fax: +86 755 89226555;
E-mail: tsh-xyz@163.com

indicates the associated number of left-son, "r" represents its right son.

The postorder traversal sequences are:

```
05 04 07 06 03 02 09 12 11 15 18 24 26 25 23 22
21 28 38 37 43 44 42 41 47 46 52 51 50 56 55 58
61 63 62 60 59 57 54 53 49 48 45 40 39 36 35 34
33 32 31 30 29 27 20 19 17 16 14 13 10 08 01 00
```

Fig. (3). Postorder traversal sequences

2.1. The Encryption Process

The following string "shenzhen, China" was divided into encrypted blocks in accordance with PKCS7 padding manner. Fig. (**4**) shows node number and the corresponding data loaded according to the preorder sequence of the first block of 8-byte.

The first block of encrypted data is obtained from postorder in accordance with Fig. (**3**): 00111111 00101110 11100111 00001011 10110101 10000111 10011000 10001010. Fig. (**5**) represents postorder traversal sequences of encrypted data and the corresponding node number.

```
  's'  'h'  'e'  'n'  'z'  'h'  'e'  'n'
00 0 08 0 16 0 24 0 32 0 40 0 48 0 56 0
01 1 09 1 17 1 25 1 33 1 41 1 49 1 57 1
02 1 10 1 18 1 26 1 34 1 42 1 50 1 58 1
03 1 11 0 19 0 27 0 35 1 43 0 51 0 59 0
04 0 12 1 20 0 28 1 36 1 44 1 52 0 60 1
05 0 13 0 21 1 29 1 37 0 45 0 53 1 61 1
06 1 14 0 22 0 30 1 38 1 46 0 54 0 62 1
07 1 15 0 23 1 31 0 39 0 47 0 55 1 63 0
```

Fig. (4). By way of preorder sequence data into the first block.

```
  63   46  -25   11  -75  -121 -104 -118
05 0 11 0 21 1 47 0 61 1 49 1 33 1 17 1
04 0 15 0 28 1 46 0 63 0 48 0 32 0 16 0
07 1 18 1 38 1 52 0 62 1 45 0 31 0 14 0
06 1 24 0 37 0 51 0 60 1 40 0 30 1 13 0
03 1 26 1 43 0 50 1 59 0 39 0 29 1 10 1
02 1 25 1 44 1 56 0 57 1 36 1 27 0 08 0
09 1 23 1 42 1 55 1 54 0 35 1 20 0 01 1
12 1 22 0 41 1 58 1 53 1 34 1 19 0 00 0
```

Fig. (5). Encrypted data and the corresponding node number.

```
  ','  'C'  'h'  'i'  'n'  'a'   2    2
00 0 08 0 16 0 24 0 32 0 40 0 48 0 56 0
01 0 09 1 17 1 25 1 33 1 41 1 49 0 57 0
02 1 10 0 18 1 26 1 34 1 42 1 50 0 58 0
03 0 11 0 19 0 27 0 35 0 43 0 51 0 59 0
04 1 12 0 20 1 28 1 36 1 44 0 52 0 60 0
05 1 13 0 21 0 29 0 37 1 45 0 53 0 61 0
06 0 14 1 22 0 30 0 38 1 46 0 54 1 62 1
07 0 15 1 23 0 31 1 39 0 47 1 55 0 63 0
```

Fig. (6). Loading the second 8 bytes into the key by preorder traversal way.

The postorder way to get the encrypted byte is shown in Fig. (**5**): 63 46 -25 11 -75 -121 -104 -118. Then by the preorder traversal sequence, the second 8 bytes are put into the key as shown in Fig. (**6**).

According to PKCS7 padding way, due to 6 bytes, less than 8 bytes, the last two bytes are filled 0x2.

The second block of encrypted data is obtained according to the postorder traversal: 11000110 01101100 01110011 10000000 00100010 00000101 10100010 10100000 [Fig. (**7**)].

```
 -58  108  115 -128  34    5   -94  -96
05 1 11 0 21 0 47 1 61 0 49 0 33 1 17 1
04 1 15 1 28 1 46 0 63 0 48 0 32 0 16 0
07 0 18 1 38 1 52 0 62 1 45 0 31 1 14 1
06 0 24 0 37 1 51 0 60 0 40 0 30 0 13 0
03 0 26 1 43 0 50 0 59 0 39 0 29 0 10 0
02 1 25 1 44 0 56 0 57 0 36 1 27 0 08 0
09 1 23 0 42 1 55 0 54 1 35 0 20 1 01 0
12 0 22 0 41 1 58 0 53 0 34 1 19 0 00 0
```

Fig. (7). Postorder traversal sequence of encrypted data and the corresponding node number.

Fig. (**7**) shows the postorder way to get the encrypted byte: -58 108 115 -128 34 5 -94 -96. Finally two encrypted block data are completed: 63 46 -25 11 -75 -121 -104 -118 -58 108 115 -128 34 5 -94 -96.

2.2. Decryption Process

The following description of the decryption process shows that first 8 bytes are loaded according to postorder traversal: 63 46 -25 11 -75 -121 -104 -118, as shown in Fig. (**8**).

```
  63   46  -25   11  -75 -121 -104 -118
05 0 11 0 21 1 47 0 61 1 49 1 33 1 17 1
04 0 15 0 28 1 46 0 63 0 48 0 32 0 16 0
07 1 18 1 38 1 52 0 62 1 45 0 31 0 14 0
06 1 24 0 37 0 51 0 60 1 40 0 30 1 13 0
03 1 26 1 43 0 50 1 59 0 39 0 29 1 10 1
02 1 25 1 44 1 56 0 57 1 36 1 27 0 08 0
09 1 23 1 42 1 55 1 54 0 35 1 20 0 01 1
12 1 22 0 41 1 58 1 53 1 34 1 19 0 00 0
```

Fig. (8). First 8 bytes of data loaded according to the postorder traversal.

The way of preorder traversal to get the data decryption block data is shown in Fig. (**9**).

```
  's'  'h'  'e'  'n'  'z'  'h'  'e'  'n'
00 0 08 0 16 0 24 0 32 0 40 0 48 0 56 0
01 1 09 1 17 1 25 1 33 1 41 1 49 1 57 1
02 1 10 1 18 1 26 1 34 1 42 1 50 1 58 1
03 1 11 0 19 0 27 0 35 1 43 0 51 0 59 0
04 0 12 1 20 0 28 1 36 1 44 1 52 0 60 1
05 0 13 0 21 1 29 1 37 0 45 0 53 1 61 1
06 1 14 0 22 0 30 1 38 1 46 0 54 0 62 1
07 1 15 0 23 1 31 0 39 0 47 0 55 1 63 0
```

Fig. (9). Restore the first one plaintext according to preorder traversal.

```
 -58  108  115 -128  34    5   -94  -96
05 1 11 0 21 0 47 1 61 0 49 0 33 1 17 1
04 1 15 1 28 1 46 0 63 0 48 0 32 0 16 0
07 0 18 1 38 1 52 0 62 1 45 0 31 1 14 1
06 0 24 0 37 1 51 0 60 0 40 0 30 0 13 0
03 0 26 1 43 0 50 0 59 0 39 0 29 0 10 0
02 1 25 1 44 0 56 0 57 0 36 1 27 0 08 0
09 1 23 0 42 1 55 0 54 1 35 0 20 1 01 0
12 0 22 0 41 1 58 0 53 0 34 1 19 0 00 0
```

Fig. (10). Loaded the second block ciphertext by postorder traversal.

The second block of 8 bytes is continued to load by following the postorder traversal: -58 108 115 -128 34 5 -94 -96 as shown in Fig. (**10**).

The second ciphertext is restored by preorder traversal, obtaining a decrypted data block, "China", as shown in Fig. (**11**) below.

Upon completion of decryption, the string is restored: "shenzhen, China".

```
   ','  'C'  'h'  'i'  'n'  'a'  '2'  '2'
00 0 08 0 16 0 24 0 32 0 40 0 48 0 56 0
01 0 09 1 17 1 25 1 33 1 41 1 49 0 57 0
02 1 10 0 18 1 26 1 34 1 42 1 50 0 58 0
03 0 11 0 19 0 27 0 35 0 43 0 51 0 59 0
04 1 12 0 20 1 28 1 36 1 44 0 52 0 60 0
05 1 13 0 21 0 29 0 37 1 45 0 53 0 61 0
06 0 14 1 22 0 30 0 38 1 46 0 54 1 62 1
07 0 15 1 23 0 31 1 39 0 47 1 55 0 63 0
```

Fig. (11). Restore the second ciphertext by preorder traversal.

3. MULTIPLE BINARY TREES ENCRYPTION PRINCIPLE

Some binary tree traversal sequences may determine the binary tree, for example:

The preorder and inorder sequence of the binary tree can uniquely determine it.

The inorder and postorder sequence of the binary tree can uniquely identify it.

A preorder and postorder sequence of the binary tree can only determine it.

Based on the above rules, using only one binary tree to encrypt/decrypt known plaintext attack or chosen plaintext attack is very easy to calculate the key. So multiple binary trees are needed to be used in order to achieve more secure and efficient encryption algorithm.

For this reason, not one binary, but multiple trees are encrypted. For example, above cipher text EDCBGHFA was encrypted again with the second tree as shown in Fig. (**1b**). EDCBGHFA is loaded into the tree by the preorder traversal. Output postorder nodes sequence: BCDFHAGE as the last encrypted sequence. The second tree was used to decrypt the BCDFHAGE to EDCBGHFA, then the first tree was used to restore EDCBGHFA to plaintext ABCDEFG, thus completing the decryption process.

By the adding principle, if the first tree has n_1 nodes, the number of the binary trees is configured by $B(n_1)$:

$$B(n_1) = \sum_{k=0}^{n_1-1} B(k)B(n_1 - k - 1) \tag{2}$$

The second tree has n_2 nodes, the number of the binary trees configured for $B(n_2)$.

$$B(n_2) = \sum_{k=0}^{n_2-1} B(k)B(n_2 - k - 1) \tag{3}$$

If the number of nodes of the two binary trees have the same nodes set then $n = n_1 = n_2$, and according to the multiplication principle, the number of combinations of two trees CB(n,2) is configured.

$$CB(n,2) = \left(\sum_{k=0}^{n-1} B(k)B(n - k - 1)\right)^2 \tag{4}$$

With two or more binary trees as the key, the number of keys can produce more than a single binary. For example, if the encryption key has 64 nodes of binary, the keys number is: 3.68E35. If 2 binary trees are encrypted with 64 nodes, the keys number can be increased: 1.35E71. In practice, to improve security, even more binary trees can be used for this encryption/decryption process.

CONCLUSION

In contrast to AES algorithm, when the key length is n, a binary tree with n nodes to encrypt/decrypt has more keys and key exhaustive search harder than AES algorithm.

However, binary tree can be inferred by its traversals of preorder, inorder and/or postorder. So using a known plaintext attack or chosen-plaintext attack to crack the key is easy. For this reason, the actual encryption/decryption is to be taken for multiple binary trees. That data continues to be encrypted/decrypted by using more binary trees which produces more keys.

B(n) represents the number of binary trees generated by n nodes. There are r binary trees with n nodes to generate CB(n, r) keys:

$$CB(n,r) = \left(\sum_{k=0}^{n-1} B(k)B(n - k - 1)\right)^r \tag{5}$$

Multiple binary trees are used to achieve symmetric encryption algorithm, the key of which is completely random and is not reused. If the violent attacks occur, the effort increases exponentially by the growing length of the key. Therefore, this encryption method is secure.

Because the binary tree traversal visits each node in turn, so the encryption/decryption time complexity is O(n). Thus this encryption/decryption method is also efficient.

CONFLICT OF INTEREST

The author confirms that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

This work was financially supported by Shenzhen Institute of Information Technology scientific research raising project (Special topic of school enterprise cooperation) "open mobile educational management system"(lg2014010).

REFERENCES

[1] M. Mitsuru, "Linear cryptanalysis method for DES cipher," *Lecture Notes in Computer Science*, Springer-Verlag, vol. 765, pp. 368-397, 1993.

[2] E. Biham, and A. Shamir, "*Differential Cryptanalysis of Data Encryption Standard*," New York: Springer-Verlag, 1993.

[3] J. Daemen and V. Rijmen, AES proposal: Rijndael. In AES Round 1 Technical Evaluation CD-1: Documentation. NIST, August 1998. See http://www.esat.kuleuven.ac.be/~rijmen/rijndael/ or http://www.nist.gov/aes.

[4] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, and D. Whiting, "Improved cryptanalysis of Rijndael," In: *FSE'00, volume 1978 of Lecture Notes in Computer Science*, pp. 213-230, 2000.

[5] J. Daemen, and V. Rijmen, "*The Design of Rijndael: AES - The Advanced Encryption Standard*," Springer, 2002.

[6] A. Biryukov, and D. Khovratovich, "Related-Key Cryptanalysis of the Full AES-192 and AES-256," In: *ASIACRYPT'09, of Lecture Notes in Computer Science*, Springer, 5912, pp. 1-18., 2009.

[7] Y. Wei, S. Xu, G. Deng, and T. He, "*Analysis of the Forming of Binary Trees with n Nodes*," ICITME, 2014.

[8] D. E. Knuth, "The Art of Computer Programming: vol. 1, Fundamental Algorithms," 2rd ed. Reading, MA: Addison-Wesley, vol. 329, pp. 347~351, 1973.

[9] Z. Tang, "Algorithms for constructing a strictly binary tree based on its traversals," *Journal of SooChow University(Natural Sciemce Edition)*, 2010.

[10] Z. Tang, "Methods for uniquely determining a tree or a binary tree based on its traversal sequences," *Mini-Micro System*, pp. 985-988, 2001.

[11] Z. Tang, "Algorithms for construction a tree based on its traversals," *Journal of SooChow University(Natural Science Edition)*, vol. 27, no. 3, 2011.

[12] E. Mäkinen, "Constructing a binary tree efficiently from its traversals," *International Journal of Computer Mathematics*, vol. 75, no. 2, pp. 143-147, 2000.

Received: June 10, 2015 Revised: July 29, 2015 Accepted: August 15, 2015