Tweakable Block Mode of Operation for Disk Encompression Using Cipher Text Stealing

Rashmita Padhi(✉) and B. N. B. Ray

Department of Computer Science and Applications, Utkal University, Vani Vihar, Bhubaneswar, India

Abstract. In this paper, we study a particular class of symmetric algorithms that aim to ensure confidentiality by using a functionality that is tweakable enciphering scheme. A tweakable enciphering scheme is a length preserving encryption protocol which can encrypt messages of varying lengths. The security goal is to satisfy the notion of the tweakable strong pseudorandom permutation (SPRP). Our proposed work is a modified version of XTS that is Xor-Encrypt-Xor with Cipher Text Stealing. This work includes a Galois Field multiplier GF (2^{128}) that can operate in any common field representations. This allows very efficient processing of consecutive blocks in a sector. To handle messages whose length is greater than 128-bit but not a multiple of 128-bit.

Keywords: Block cipher · XTS (XOR Encrypt Xor with ciphertext stealing) · Galosis Field multiplier GF (2^{128}) · Strong Pseudorandm Permutation (SPRP) · Tweakable enciphering

1 Introduction

Explosive growth of the digital storage and communication of data require adequate security. Cryptology is the science that aims to provide information security in the digital world. Information security comprises many aspects, the most important of which are confidentiality and authenticity. *Confidentiality* means keeping the information secret from all except those who are authorized to learn or know it. *Authenticity* involves both ensuring that data have not been modified by an unauthorized person (*data integrity*) and being able to verify who is the author of the data (*data origin authentication*). In this paper we provide data encryption with compression by focusing on the tweakable encipher scheme as these appear to offer the best combined security and performance. Our proposed work is a modified version of XTS that is Xor-Encrypt-Xor with Cipher Text Stealing. This work includes a Galois Field multiplier GF (2^{128}) that can operate in any common field representations. This allows very efficient processing of consecutive blocks in a sector. To handle messages whose length is greater than 128-bit but not a multiple of 128-bit.The objective of the work is to develop a fast data encryption system. The requirement is actually to achieve security, speed and error propagation with less consumption of space, i.e., the size of hardware implementation and the amount of secure storage space required.

P. K. Behera and P. C. Sethi (Eds.): CSI 2020, CCIS 1372, pp. 100–108, 2021.
https://doi.org/10.1007/978-981-16-2723-1_11

Data Encryption

Hard disk encryption is usually used to protect all the data on the disk by encrypting it. The whole disk is encrypted with a single/multiple key(s) and encryption/decryption are done on the fly, without user interference. The encryption is on the sector level, that means each sector should be encrypted separately. There are two ways to encrypt a hard disk: at the file level and at the driver level. Encryption at the file level means that every file is encrypted separately. To use a file that's been encrypted, that file must be first decrypted, and then it is used, and then re-encrypts it. Driver-level encryption maintains a logical drive on the user's machine that has all data on it encrypted. In this paper we used AES.The AES is a symmetric block cipher i.e., encryption rule e_k is either the same as decryption rule d_k, or easily derived from it. During one round of AES the entire traffic is divided into fixed block of size 128 bits which is known as a State. AES is an iterated cipher, i.e., ciphers frequently incorporate a sequence of permutation & substitution operations. There are three allowable key lengths, namely 128 bits, 192 bits, and 256 bits. It follows a number of rounds N_r, depends on the key length.$N_r = 10$ if the key length is 128 bits, and $N_r = 12$ if the key length is 192 bits, and $N_r = 14$ if the key length is 256 bits.

2 Existing Work

LRW Mode of Encryption in AES

In LRW mode of AES encryption two keys are used i.e. primary and secondary key. These keys are independent to each other. Each key length is 128 0r 256 bits. In this paper Key_1 and Key_2 are Primary and Secondary keys respectively. The entire message is divided into fixed size blocks which are known as Plain text P. The encryption process is applied to each plaintext block and corresponding cipher text block C is obtained. I is the index of the block.

Input:
P: Fixed size 128 bits Plaintext Block
Key_1: 128 or 256 bits Primary or Cipher Key
Key_2: 128 bits Secondary or Tweak Key
I: Index of data in 128 bit representation
Output:
C: Corresponding Cipher text Block
The following sequence of steps are applied to each plain text block to obtain the Corresponding Cipher text Block :
1. If $I > 2^{128}$-1 or I , exit and output ERROR
2. $T \leftarrow Key_2 \otimes I$
3. $PP \leftarrow P \oplus T$
4. $CC \leftarrow$ AES-enc(Key_1,PP)
5. $C \leftarrow CC \oplus T$
6. Return C

LRW Mode of Decryption in AES

In LRW mode of AES dencryption two keys are used i.e. primary and secondary key. These keys are independent to each other Each key length is 128 0r 256 bits. In this paper Key_1 and Key_2 are Primary and Secondary keys respectively. The decryption process is applied to each Cipher text block C and corresponding Plain text block P is obtained. I is the index of the block.

Input:

C: Fixed size 128 bits Cipher text Block

Key_1: 128 or 256 bits Primary or Cipher Key

Key_2: 128 bits Secondary or Tweak Key

I: Index of data in 128 bit representation

Output:

P: Corresponding Plain text Block

The following sequence of steps are applied to each Cipher text block to obtain the Corresponding Plain text Block :

1. If $I > 2^{128}-1$ or I < Zero
Then exit and Display ERROR as Output
2. $T \leftarrow Key_2 \otimes I$
3. $CC \leftarrow C \oplus T$
4. $PP \leftarrow$ AES-dec (Key_1 ,CC)
5. $C \leftarrow PP \oplus T$
6. return P

Limitations: LRW-AES tweakable mode scope is limited.
Large volume of data storage cannot be possible using this procedure.

3 Proposed Work

XTS-AES Tweakable Block Cipher

The XTS-AES Tweakable Block Ciphers XEX(Xor-Encrypt-Xor, designed by Rogaway [26])-basd Tweaked Code Book mode (TCB) with Cipher Text Stealing (CTS). Although XEX-TCB-CTS should be abbreviated as XTC, “C” was replaced with “S” (for “stealing”) to avoid confusion with the abbreviated ecstasy. Cipher text stealing provides support for sectors with size not divisible by block size, for example, 520-byte sectors and 16-byte blocks.

Meaning of Used Symbols:
Symbols which are used in the equations has the following meaning:

α : Primitive element of Finite Field $GF(2^{128})$
$\oplus$: Bitwise XOR operation
$\otimes$: Two polynomials Multiplication in $GF(2^{128})$
$\leftarrow$: Assigning a value to the variable
| : Binary Concatenation
For example , if $N_1=1011_2$ and $N_2=1110011_2$,
then $N_1 | N_2 = 10111110011_2$.
$\lfloor X \rfloor$: Floor operation of String

Data Units and Tweaks
The size of each data unit must be greater than or equal to 128 bits. The number of blocks having length 128 bits must be less than or equal to $2^{128}-2$. The number of block of size 128-bit should be less than or equal to 2^{20}. A tweak value is assigned to each data unit which is a positive integer. The values of the tweak are assigned sequentially. The assignment of tweak value will be started from any arbitrary positive integer. In AES tweak encryption the tweak will be converted into array of little-endian byte. For example, $123456789A_{16}$ is a tweak value which is converted into byte array $9A_{16}$, 78_{16}, 56_{16}, 34_{16}, 12_{16}.

XEX Tweakable Mode Using Cipher Text Stealing Encryption (XTS-AES Encrypt)
XEX Tweakable Mode using Cipher text Stealing Encryption procedure, a single block of size 128-bit block is implemented by the following equation:

$$C \leftarrow \text{XTS-AES-Encrypt}(Key, P, i, j)$$

Input:
P: Fixed size 128 bits Plain text Block
Key: 512 or 256 bits Cipher Key
i: Tweak value of size 128 bit
j : Sequential number of the data unit

Output:
C: Corresponding Cipher text Block

The key is obtained by concatenating of two fields i. e called Key_1 and Key_2 i.e. Key = $Key_1 | Key_2$. In this case Key_1 and Key_2 are Primary and Secondary keys respectively and both are equal size . The following sequence of steps are perfomed to obtain the corresponding cipher text block .

1) $T \leftarrow \text{AES-enc}(Key_2, i) \otimes \alpha^{Js}$
2) $PP \leftarrow P \oplus T$
3) $CC \leftarrow \text{AES-enc}(Key_1, PP)$
4) $C \leftarrow CC \oplus T$

XTS-AES Encryption of a Data Unit

The encoding process of 128 or more bits plain text block can be implemented by using the following equation:

$$C \leftarrow \text{XTS}\textit{text}-\text{AES-Encrypt}(Key, P, i)$$

Input:

P: Fixed size 128 bits Plain text Block

Key: 512 or 256 bits Cipher Key

i: Tweak value of size 128 bit

Output:

C: Corresponding Cipher text Block

C is the cipher text which is obtained by using the above operation on the same block size of *P* *.The entire traffic or message is* partitioned into $m+1$ number of blocks:

$P = P_0 |\ldots P_{m-1} |P_m$

The value of *m* is the largest integer that is 128*m* P.

The size of the initial m blocks that is $P_0,\ldots, P_{m-1}$ are 128 bits long, and the size of the end block that is P*m* is between 0 and 127 bits long. P*m can be a null string that the size of the string is zero*. The key is obtained by concatenating of two fields i. e called $Key_1$and Key_2 i.e. Key = $Key_1| Key_2$. In this case Key_1 and Key_2 are Primary and Secondary keys respectively and both are equal size . The following sequence of steps are preformed to obtain the corresponding cipher text block .

1)for q ← 0 to m-2 do

a) Cq ← XTS-AES-Encrypt(Key, P_q, i, q)

2)b ← bit-size of Pm

3)if b=0 then do

a) XTS-AES-Encrypt(Key, P_{m-1}, i, m-1)

b) C_m ← empty

4)else do

a) CC ← XTS-AES-Encrypt(Key, P_{m-1}, i, m-1)

b) C_m ← first b bits of CC

c) CP ← last (128-b) bits of CC

d) PP ← P_m | CP

e) C_{m-1}← XTS-AES-Encrypt(Key, PP, i, m)

5) C ← $C_0|\ldots | C_{m-1}|C_m$

XEX Tweakable Mode Using Cipher Text Stealing Decryption (XTS-AES Decrypt)

XEX Tweakable Mode using Cipher text Stealing Decryption procedure, a single block of size 128-bit block is implemented by the following equation:

$$\text{P} \leftarrow \text{XTS}\textit{text}-\text{AES-Decrypt}(\text{Key, C, i, j})$$

Input:
C: Fixed size 128 bits Cipher text Block
Key: 512 or 256 bits Key
i: Tweak value of size 128 bit
j : Sequential number of the data unit

Output:
P : Corresponding Plain text Block

The key is obtained by concatenating of two fields i.e called $Key_1$and Key_2 i.e. Key = Key_1| Key_2. In this case Key_1 and Key_2 are Primary and Secondary keys respectively and both are equal size . The following sequence of steps are perfomed to obtain the corresponding plain text block .

1) $T \leftarrow$ AES-dec (Key_2, i) $\otimes \alpha^{Js}$
2) $CC \leftarrow C \oplus T$
3) $PP \leftarrow$ AES-dec(Key_1, PP)
4) $P \leftarrow PP \oplus T$

XTS-AES Decryption of a Data Unit

The decoding process of 128 or more bits cipher text block can be implemented by using the following equation:

$$P \leftarrow \text{XTS-AES-Decrypt}(Key, C, i)$$

Input:
C: 128 bits Cipher text Block
Key : 512 or 256 bits Cipher Key
i: Tweak value of size 128 bit

Output:
P: Corresponding Plain text Block

P is the plain text which is obtained by using the above operation on the same block size of C *.The entire cipher text is* partitioned into m+1 number of blocks:

$$C = C_0 | \ldots | C_{m-1} | C_m$$

The value of m is the largest integer that is 128m P.
The size of the initial m blocks that is $C_0, \ldots, C_{m-1}$ are 128 bits long, and the size of the end block that is C_m is between 0 and 127 bits long. C*m can be a null string that the size of the string is zero*. The key is obtained by concatenating of two fields i. e called Key_1 and Key_2 i.e. Key = $Key_1 | Key_2$. In this case Key_1 and Key_2 are Primary and Secondary keys respectively and both are equal size . The following sequence of steps are performed to obtain the corresponding plain text block .

1) for q ← 0 to m-2 do
   a) Pq ← XTS-AES-Decrypt(Key, Cq, i, q)
2) b ← bit-size of Cm
3) if b = 0 then do
a) P_{m-1}← XTS-AES-Decrypt(Key,C_{m-1}, i, m-1)
b) P_m ← empty
4) else do
a)PP ← XTS-AES-Decrypt(Key, C_{m-1}, i, m)
b)P_m ← first b bits of PP
c) CP ← last (128-b) bits of PP
d) CC← C_m | CP
e) P_{m-1}← XTS-AES Decrypt(Key, CC, i, m-1)
5. P← $P_0 | \ldots | P_{m-1} | P_m$

4 Performance Analysis

With the wide spread of multi-core processors, speeding up encryption using parallelization is made possible and parallelization is not a luxury anymore and can increase the performance significantly. Encryption mode of operation should support parallelization. CBC and CFB cannot be parallelized, while XTS can be parallelized on the sector level as each sector is encrypted independently to other sectors. Also a plaintext can be recovered from just two adjacent blocks of cipher text. As a consequence, decryption *can* be parallelized.

5 Conclusion

In this paper a highly secure XTS-based Tweaked Block Enciphering scheme with Cipher text Stealing has been proposed for hard disk encryption. The important features of this scheme are the use of Cipher block chaining mode like operations to gain the error propagation property. A one-bit change in a plaintext affects all following cipher text blocks in a sector. The tweak T is calculated by encrypting (using AES) the block address (after being padded with zeros) with the tweak key due to this step the value of the tweak is neither known nor controlled by the attacker. Any difference between two tweaks result full diffusion in both the encryption and decryption directions. All these factors improve security. It has been shown that the proposed mode possesses a high throughput as compression is done before enciphering scheme. Only standard shift and add (xor) operators have been used for the non-linear multiplication function in the finite field GF(2^{128}) having O(1) time complexity, therefore gives better resistance against linear cryptanalysis without degradation in performance. This proposed mode has ability to encrypt arbitrary length messages due to the use of cipher text stealing technique.

6 Open Problems

There still remain many open problems in the search for efficient and secure data encryption. It can therefore be hoped that many remaining open problems can be solved in the coming years. These are some of the interesting open problems: that is: There is a lack of good Boolean functions for the tweak generator which are efficient and also resist the cryptanalytic attacks, in particular algebraic and fast algebraic attacks, Extend the current work to audio, and video encryption. The given XEX ciphertext Stealing technique can be efficiently implemented by using AES having key length 256-bit. Introduce the hardware implementation of the entire work.

References

1. Schneier, B.: Applied Cryptography, Second Edn. Wiley Press
2. Stinson, D.R.: Cryptography Theory and Practice, Second Edn. CRC Press
3. Nelson, M., Gailly, J.-L.: The Data Compression Book, Second Edn. M&T Press
4. Sarkar, P.: Efficient Tweakable Enciphering Schemes from (Block-Wise) Universal Hash Functions
5. Chakraborty, D., Sarkar P.: HCH: A New Tweakable Enciphering Scheme Using the Hash-Counter-Hash
6. McGrew, D.: Counter Mode Security: Analysis and Recommendations (2002). https://citeseer.ist.psu.edu/mcgrew02counter.html
7. Rogaway, P., Bellare, M., Black, J.: OCB: A blockcipher mode of operation for efficient authenticated encryption. ACM Trans. Inf. Syst. Secur. **6**(3), 365–403 (2003)
8. Schroeppel, R.: The Hasty Pudding Cipher. The first AES conference, NIST (1998). https://www.cs.arizona.edu/~rcs/hpc
9. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 31–46. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_3

10. Goos, G., Hartmanis, J., van Leeuwen, J., Schneier, B. (eds.): FSE 2000. LNCS, vol. 1978. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44706-7
11. Fruhwirth, C.: New Methods in Hard Disk Encryption (2005). https://clemens.endorphin.org/nmihde/nmihde-A4-ds.pdf
12. Ferguson, N.: AES-CBC + Elephant diffuser: A Disk Encryption Algorithm for Windows Vista (2006). https://download.microsoft.com/download/0/2/3/0238acaf-d3bf-4a6d-b3d6-0a0be4bbb36e/BitLockerCipher200608.pdf
13. Lempel–Ziv–Welch. https://en.wikipedia.org/wiki/Lempel-Ziv-Welch
14. U.S. Code Collection. https://www4.law.cornell.edu/uscode/35/154.html
15. Blelloch, G.E.: Introduction to Data Compression. https://www.eecs.harvard.edu/~michaelm/CS222/compression.pdf
16. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. J Cryptol **4**(1), 3–72 (1991). https://doi.org/10.1007/BF00630563
17. Matsui, M.: The first experimental cryptanalysis of the data encryption standard. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 1–11. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48658-5_1
18. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48285-7_33