Extended Substitution Cipher Chaining mode (ESCC)

Mohamed Abo El-Fotouh and Klaus Diepold
Institute for Data Processing (LDV)
Technische Universität München (TUM)
80333 München, Germany
[mohamed,kldi]@tum.de

Abstract

In this paper, we present a new tweakable narrow-block mode of operation, the Extended Substitution Cipher Chaining mode (ESCC), that can be efficiently deployed in disk encryption applications. ESCC is an extension of Substitution Cipher Chaining mode (SCC) [5]. Unlike SCC, ESCC is resistant to the attacks in [6, 7, 8].

1 Introduction

In [5], three new disk encryption modes of operations have been introduced, the Substitution Cipher Chaining mode (SCC) which is a narrow block mode of operation that provides error propagation, and two new variants of ELEPHAT (ELEPHANT^{+} and ELEPHANT$^{\times}$) that uses SCC, where ELEPHANT is a synonym for Windows Vista's disk encryption algorithm. All these modes of operations uses the Advanced Encryption Standard (AES) [2] as there block cipher.

In [6], SCC-128 (SCC which uses AES with 128-bit key) was attacked using 5 different attacks. These attacks were able to recover all the unknown keys/masks used by SCC-128 with at most 2^{40} cipher executions and 5×2^8 chosen plaintext, but the main attack in [6] was not practical on SCC-256 (as it costs about 2^{172} cipher executions).

In [8], SCC-256 (SCC which uses AES with 256-bit key) was attacked using 7 different attacks. These attacks were able to recover all the unknown keys/masks used by SCC-256, with at most 2^{44} cipher executions and 6×2^{32} chosen plaintext, using some novel ideas from [9].

In this paper, we propose a novel narrow-block disk encryption mode of operation. We named this mode Extended Substitution Cipher Chaining mode (ESCC). The ESCC mode is a tweakable block cipher mode of operation, that is base on Cipher Block Chaining mode (CBC) [16] and the Static Substitution Model (SSM) [4]. The SSM model can provide a block cipher with a secondary key. The secondary key is used to replace some bits of the cipher's expanded key. In ESCC, each sector has its unique tweak, this tweak and the previous ciphertext block will replace some bits of the expanded key (with the exception to the first block). ESCC is resistant to the attacks in [6, 7].

This paper proceeds as follows: In section 2, we will present the constraints facing the disk encryption applications. In section 3, we presented our proposed mode of operation. We conclude in section 4.

2 Disk encryption

Disk encryption is usually used to encrypt all the data on the hard disk, where all the hard disk is encrypted with a single/multiple key(s) and encryption/decryption are done on the fly, without user interference [3]. The encryption is on the sector level, that means each sector should be encrypted separately. In the following sub-section, we will define the existing constrains.

2.1 Disk Encryption Constraints

Data size: The ciphertext length should be the same as the plaintext length. In this paper, we will use the current standard (512-bytes) for the plaintext.

Performance: The used mode of operation should be fast enough, as to be transparent to the users (If using the mode of operation results in a significant and noticeable slowdown of the computer there will be great user resistance to its deployment [10]).

2.2 General Scheme and Tweak calculation

In our general scheme, the mode of operation takes four inputs to calculate the ciphertext (4096-bit). These inputs are:

1. The plaintext of size 4096-bit.
2. Encryption key of size 128 or 256-bit.
3. Tweak Key of size 128 or 256-bit.
4. Sector ID of size 64-bit.

Usually a block cipher accepts the plaintext and the encryption key to produce the ciphertext. Modes of operations have introduced other inputs. Some of these mode use initial vectors like in CBC, CFB and OFB modes [16], counters like in CTR [15] or nonces like in OCB mode [18]. The idea of using a tweak was suggested in HPC [19] and used in Mercy [1]. In [13], the formal definition of tweakable block ciphers has been introduced. In this paper, the term tweak is associated with any other inputs to the mode of operation with the exception of the encryption key and the plaintext. There are different methods to calculate the tweak from the sector ID like ESSIV [11] and encrypted sector ID [10]. We are going to use the encrypted sector ID approach, where the sector ID (after being padded with zeros) is encrypted by the tweak key to produce the tweak.

3 Proposed mode

3.1 Goals

The goals of designing the ESCC mode are:

Security: The constraints for disk encryption imply that the best achievable security is essentially what can be obtained by using ECB mode with a different key per block [17]. This is our aim.

Performance: ESCC should be at least as fast as the current solutions.

Error propagation: ESCC should propagate error to further blocks (this may be useful in some applications).

3.2 Keys

The secret key in ESCC is divided into three different keys (each of them can be either 128- or 256-bit):

1. **EKey**: which is used to generate the expanded key, used in encrypting the blocks .
2. **TK**: which is used to encrypt the sector ID to produce the tweak.
3. **BK**: which is used to generate the **BT** array, where **BT** is an array of sixty four 128-bit blocks. **BT** is constructed once at the initialization of ESCC mode, it is constructed using the AES in the counter mode [15], where the counter is initialized with zero and **BK** is the encryption key for the counter mode.

3.3 Terminologies

The following terminologies are used to describe ESCC.

IN: The input plaintext of size 4096-bit.

SID: The sector ID encoded as 64-bit unsigned integer.

GetTweak(TK,SID): Encrypts (using AES) **SID** after padding with zeros with **TK** and returns the result.

T: The tweak.

ExKey: The expanded AES key.

Expand-Key(EK): Expands the **EK** with the AES key setup routine and returns the result.

$\mathbf{X}_i$**:** The i^{th} block of text **X**, where a block is 128-bit.

$\bigoplus$**:** Bitwise xor operation.

OUT=Encrypt-AES(IN,ExKey): Encrypts **IN**, using the AES encryption routine with **ExKey** as the expanded key, and returns **OUT**.

Substitute(T,ExKey,i): Replaces the i^{th} round subkeys in **ExKey** with **T** (note that: the first round of the AES is round zero and it is the pre-whitening process).

len(X): Returns the length of the string **X** in bits.

$<<$**:** is a left rotation operation, where the rotation value is written on its right size.

3.4 Design

We decided to build the ESCC mode using the SSM model [4] to inherit from its security and high performance and use CBC like operations to gain the error propagation property. The listing of ESCC is in table 1 and it works as follows:

- The tweak **T** is calculated by encrypting the sector ID with the tweak key **TK**, due to this step the value of the tweak is neither known nor controlled by the attacker.
- The expanded key **ExKey** is calculated.
- the values of x, y and z are determined by the encryption key size.
- For the first block:
  - The secret tweak **T** replaces the subkeys of the y^{th} round.
  - The secret 128-bit $\mathbf{BT}_0 \oplus T$ replace the subkeys of the x^{th} round.

pt

Table 1. ESCC listing for disk encryption.

```
Encrypt-ESCC(IN,EK,Keylen,TK,SID)
   T=GetTweak(TK,SID)
   ExKey=Expand-Key(EK)
   KL=len(EK)
   if(KL==128)
      x=4    y=5    z=6
   else
      x=5    y=7    z=10
   end if
   Substitute(T,ExKey,y)
   Substitute(BT_0 ⊕ T,ExKey,x)
   Substitute(BT_1 ⊕ T,ExKey,z)
   AES-Encrypt(ExKey,IN_i,OUT_i)
   for i=1 to 31
      Substitute(BT_{2×i} ⊕ (OUT_{i-1} << 32),ExKey,x)
      Substitute(BT_{2×(i+1)} ⊕ (OUT_{i-1} << 64),ExKey,z)
      TT=OUT_{i-1} ⊕ T
      Substitute(TT,ExKey,y)
      AES-Encrypt(ExKey,IN_i,OUT_i)
   end for
return OUT
```

  - The secret 128-bit $\mathbf{BT}_1 \oplus T$ replace the subkeys of the z^{th} round.
  - The first block is encrypted by the new expanded key.

- A loop that runs 31 times (where i takes the values from 1 to 31):
  - The secret 128-bit $\mathbf{BT}_{2\times i}$ xored with the rotated ciphertext of the previous block replaces the subkeys of the x^{th} round.
  - The secret 128-bit $\mathbf{BT}_{(2\times i)+1}$ xored with the rotated ciphertext of the previous block replaces the subkeys of the z^{th} round.
  - A variable **TT** is calculated by xoring ciphertext of the previous block with T.
  - **TT** acts as the *active tweak* and replaces the subkeys of the y^{th} round in the expanded key.
  - The i^{th} block is encrypted by the new expanded key.

3.5 Discussion of ESCC Mode

The goal of ESCC is to encrypt each block on the hard drive in a different way. This was achieved by using the SSM model, where:

- The active tweak **TT** is placed in the middle of the expanded key, to offer full diffusion and full confusion properties in both the encryption and decryption directions (i.e any difference between two active tweaks, will be associated with full confusion and full diffusion in both the encryption and decryption directions, eliminating the bit-flipping attack of the CBC mode). Note that AES requires only four rounds to obtain full bit confusion (or mixing) and diffusion (each input bit affecting each output bit) properties [14].
- Note that the active tweak **TT** is the result of xoring:
  1. The tweak **T** (which is unique, secret and not controlled by the attacker).
  2. The ciphertext of the previous block (which is known and controlled by the attacker).
  3. From the above two notes, the attacker does not know the value of **TT**, but can flip its bits. But by changing any bits of a ciphertext block, this will result in a difference in *3 different columns*, which will destroy any attempt to lunch chosen plaintext/ciphertext attacks as in [6, 7], so any change in **TT** will be associated with full confusion and full diffusion in both the encryption and decryption directions.
- Replacing the subkeys of the x^{th} and z^{th} rounds offers full diffusion and full confusion in the encryption and decryption directions among the blocks of the same sector. Note that all the values of **BT** are unique and key dependent.

Notes:

1. By introducing the tweak, the attacker can not perform the mix-and-match attack [17] among blocks of different sectors, as each sector has a unique secret tweak. The tweak replaces the subkeys of the middle round of the AES to assure that any difference between two tweaks, will be associated with full confusion and full diffusion in both the encryption and decryption directions. Thus, encrypting two equal blocks in different sectors will produce two different ciphertexts and decrypting two equal blocks in different sectors will produce different plaintexts.
2. By introducing the **BT** array values (that replaces certain words in the expanded key) the attacker can not perform the mix-and-match attack among the blocks within the same sector. As each sector has two distinct 128-bit in the expanded key. This requirement is achieved in both the encryption and decryption directions. As equal plaintext blocks (within the same sector), will have the same state until the x^{th} encryption round then the state will change. And equal ciphertext blocks (within the same sector), will have the

Table 2. Number of clock cycles reported by different mode of operation.

	Key length 128-bits	Key length 256-bits
CBC	12630	16898
CFB	12585	16935
LRW	19778	24015
XTS	24420	28846
ESCC	12660	16867

same state until the z^{th} decryption round then the state will change.

3.6 Performance

The Speed presented in table 2, are obtained form the optimized Gladman's C implementation [12], Running on a PIV 3 GHz (Note that the values reported are in processor clock cycles). Note that the reported values are the minimum of 1000 measurements, to eliminate any initial overheads or cache misses factors. It is clear that ESCC possesses high throughput.

3.7 Pros of ESCC

Security: Each sector is encrypted in a different way, so replacing ciphertext between different sectors will not help the attacker, as they are encrypted with a different expanded keys and each block within the sector is encrypted in a different way, due to the use of **BT** (so the attacker will not benefit from changing the positions of the blocks).

Performance: ESCC possesses high performance as it uses only simple and fast operations.

Error propagation: As each sector depends on its previous sector, error propagation is met.

ESCC meets all its design goals.

4 Conclusions

In this paper, we proposed a novel mode of operation for disk encryption applications. Our proposed mode possesses a high throughput. Although, it was designed based on the CBC mode, it does not suffer from the bit-flipping attack.

References

[1] P. Crowley. Mercy: a fast large block cipher for disk sector encryption. In *Bruce Schneier, editor, Fast Software Encryption: 7th International Workshop, FSE 2000*, 2001.

[2] J. Daemen and V. Rijmen. *The Design of Rijndael*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[3] M. El-Fotouh and K. Diepold. Statistical Testing for Disk Encryption Modes of Operations. Cryptology ePrint Archive, Report 2007/362, 2007.

[4] M. El-Fotouh and K. Diepold. Dynamic Substitution Model. In *The Fourth International Conference on Information Assurance and Security (IAS'08)*, Naples, Italy, September 2008.

[5] M. El-Fotouh and K. Diepold. The Substitution Cipher Chaining mode. In *SECRYPT 2008*, Porto, Portugal, July 2008.

[6] M. El-Fotouh and K. Diepold. Cryptanalysis of Substitution Cipher Chaining mode (SCC). In *to appear, in 2009 IEEE International Conference on Communications (ICC 2009)*, Dresden, Germany, June 2009.

[7] M. El-Fotouh and K. Diepold. Enhanced Cryptanalysis of Substitution Cipher Chaining mode (SCC-128). In *Submitted*, 2009.

[8] M. El-Fotouh and K. Diepold. How to break the Substitution Cipher Chaining mode with 256-bit Keys (SCC-256). In *Submitted*, 2009.

[9] M. El-Fotouh and K. Diepold. The Pushdown attack on AES. In *SECUREWARE 2009*, Athens, Greece, June 2009.

[10] N. Ferguson. AES-CBC + Elephant diffuser : A Disk Encryption Algorithm for Windows Vista. `http://download.microsoft.com/download/0/2/3/0238acaf-d3bf-4a6d-b3d6-0a0be4bbb36e/BitLockerCipher200608.pdf`, 2006.

[11] C. Fruhwirth. New Methods in Hard Disk Encryption. `http://clemens.endorphin.org/nmihde/nmihde-A4-ds.pdf`, 2005.

[12] B. Gladman. `http://fp.gladman.plus.com/AES/index.htm`, August 2008.

[13] M. Liskov, R. Rivest, and D. Wagner. Tweakable Block Ciphers. In *CRYPTO '02: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, 2002.

[14] L. May, M. Henricksen, W. Millan, G. Carter, and E. Dawson. Strengthening the Key Schedule of the AES. In *ACISP '02: Proceedings of the 7th Australian Conference on Information Security and Privacy*, pages 226–240, London, UK, 2002. Springer-Verlag.

[15] D. McGrew. Counter Mode Security: Analysis and Recommendations. http://citeseer.ist.psu.edu/mcgrew02counter.html, 2002.

[16] A. Menezes, P. V. Oorschot., and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[17] I. P1619. Ieee standard for cryptographic protection of data on block-oriented storage devices. *IEEE Std 1619-2007*, April 2008.

[18] P. Rogaway, M. Bellare, and J. Black. OCB: A block-cipher mode of operation for efficient authenticated encryption. *ACM Trans. Inf. Syst. Secur.*, 6(3):365–403, 2003.

[19] R. Schroeppel. The Hasty Pudding cipher. The first AES conference, NIST, 1998.