東 海 大 學 資 訊 工 程 研 究 所

碩士論文

指導教授：呂芳懌 博士

一個採用循序邏輯機制、三維運算和動態置換盒的加密方式

A Secure Data Encryption Method Employing a Sequential-Logic Style Mechanism, Three-Dimensional Operation and Dynamic Transition Box

研究生：戴成儒

中 華 民 國　一 百 零 二 年　七　月

東海大學碩士學位論文考試審定書

東海大學資訊工程學系 研究所

研究生 戴 成 儒 所提之論文

一個採用循序邏輯機制、三維運算和動態置換盒的加密方式

經本委員會審查，符合碩士學位論文標準。

學位考試委員會
召 集 人 羅濟舟 簽章

委 員 楊朝棟

林二郎

陳金鈴

指 導 教 授 [illegible] 簽章

中華民國 102 年 7 月 13 日

致　謝

這本論文的完成，代表著研究所的生涯已步入了尾聲。在這兩年的學習過程中，首先要感謝的是指導教授-呂芳懌教授，讓我從一個一臉茫然地踏入研究所的菜鳥， 變成一個能夠抬頭挺胸踏出社會的戰士。在這兩年來的日子裡，教授循循善誘的為我指引與督導，並且在我撰寫論文時，仔細的檢閱我的文章內容，並給予寶貴的建議與改正，使這篇論文得以更加完整。在研究所的這段日子中，我忘不了教授陪著我一起待在實驗室研究到半夜的日子，更忘不了和教授一同出席研討會與外國朋友相處的日子，其他還有更多過去不曾有過的嘗試和體驗，都要感謝教授讓我在這兩年間變得更加充實。

同時我也要感謝黃宜豐老師，在我對求學生涯陷入迷網，大學畢業正準備放棄升學之時給予開導，讓我毅然決定繼續升學就讀研究所。在我研究陷入困難時，老師也不時的與我討論，並給予我研究啟發，讓我的研究得以順利進行。

接著，要感謝實驗室的所有成員，因為有大家的熱情與幫助，為待實驗室的日子增加不少活力，今後即使離開了，我也不會忘記和大家一同奮鬥的日子，在此也祝福各位之後都能如期的完成學業。

最後，要感謝我的家人們，感謝你們的肯定與鼓勵，讓我在遇到挫折時，能夠毫不徬徨的繼續前進，讓我對未來的路途更加堅定，感謝你們。

成儒

中文摘要

近幾年來，Advanced Encryption Standard (AES)和 Data Encryption Standard (DES)已被廣泛應用於電子信息的保護。然而，由於平行計算和硬體速度的快速發展，這兩種演算法一直遭受到暴力攻擊的威脅，而要能有效抵禦這類型的威脅，基於此，我們的研究中提出了一個新的資料加密方法，稱為“安全的回授加密機制（簡稱 SeFEM)，它採用三個安全方案，包括一個循序邏輯式的加密/解密機制、三維運算和動態置換盒，其能提高密鑰的破解難度，並能有效的抵禦暴力攻擊和密碼分析攻擊，使得密文的安全等級有效的被提升。循序邏輯式的加/解密機制是一種反饋過程，其中每一回合的操作，皆會在內部生成三把動態回授金鑰提供給下一回合使用，而三維運算包括異或（⊕)、二進制加法（$+_2$）和異同（⊙）的操作，不同運算子的使用能夠進一步提高計算的複雜度。動態置換盒會非線性的重新排列金鑰中每一個 bit 的位置，使 SeFEM 的破難度難度提升。分析結果顯示，SeFEM 比 DES 和 AES 具有更高的安全等級、加密效率與更好的靈活性。

關鍵詞：SeFEM，循序邏輯式的機制，三維運算，動態置換盒，動態回授金鑰

Abstract

In recent years, the Advanced Encryption Standard (AES) and Data Encryption Standard (DES) have been commonly and widely used to protect important information carried in electronic documents. However, due to the quick development of parallel computing techniques and hardware speed, the two algorithms have so far faced the threats of Brute-Force attacks. To defend against this type of threats, in this study, we proposed a new data encryption approach, called the Secure Feedback Encryption Method (SeFEM for short), which employs three security schemes, including a sequential-logic style encryption/decryption mechanism, three-dimensional operation and dynamic transition box, to effectively enhance the security level of the delivered ciphertext, and increase the difficulty of cracking the encryption keys so as to well protect encrypted data from Brute-force and cryptanalysis attacks. The sequential-logic style encryption/decryption mechanism is a feedback process in which each of its calculation iteration/step generates three internally used dynamic feedback keys for the next iteration/step. The three-dimensional operation, including exclusive-or ($\oplus$), binary addition ($+_2$) and exclusive-and ($\odot$) operators, is utilized to further increase the computational complexity of the encryption process. The dynamic transition box nonlinearly rearranges the bits of a key for each operation, so as to increase the difficulty of cracking the SeFEM. The analytical results show that the SeFEM has a higher security level, encryption efficiency and usage flexibility than the DES and AES have.

Keywords: SeFEM, a sequential-logic style mechanism, three-dimensional operation, dynamic transition box, dynamic feedback keys

List of Contents

List of Figures

List of Tables

1. Introduction

Recently, many governments and institutes have adopted electronic documents to substitute for traditional paper documents, aiming to achieve a paperless homeland. But when a high-security-level document is transmitted through networks or the Internet, an encryption mechanism [1-3] is often required. Also, when a military office delivers a command to one of its subordinates, for example, to attack an enemy group some time later, the command must be encrypted before being sent out, particularly when the delivery goes through a wireless communication system.

On the other hand, owing to the popularity of wireless communication, wireless systems have been developed rapidly, and mobile devices are commonly used in our everyday life. However, due to their wireless transmission nature, hackers can easily eavesdrop on those messages sent through wireless channels. That is why security problems have been more serious and attracted many more researchers' attention than before. Presently Data Encryption Standard (DES) [4,5] and Advanced Encryption Standard (AES) [5,6] are two of the most widely used cryptographic techniques adopted to protect transmitted messages. However, both of them utilize only one key which is relatively short [4-6]. On the other hand, current computer processing speeds have been significantly improved. The DES encryption algorithm was successfully cracked in 1999 [4-7], implying that it is no longer a high security encryption mechanism. Although the AES has not been cracked, no one dares to say that it is always secure in protecting transmitted data. In the following, we will use documents and messages interchangeably since documents are carried in messages.

Both the AES and DES block ciphering [8] requires complicated calculation on their own parent keys so as to generate a certain number of sub-keys to encrypt plaintext. But

the combinatorial-logic style calculation is quite a problem since its outputs only rely on current inputs, without employing the outputs of its previous stage as a part of the inputs of its current stage to increase the security level of its ciphertext. Hence, their ciphertext may more easily be cracked by hackers by using cryptanalysis attacks [9], such as chosen plaintext attack [9], and attacks by statistical methods and Brute-force attacking methods [9]. Therefore, security levels of this style of encryption techniques fall short of our expectation. So how to improve their security levels has been one of the focuses of security researchers.

The principles of modern encryption mechanisms [4] are that even though the encryption process of a technique has been disclosed, as long as the hackers do not know all the encryption keys, the delivered documents are still safe since without acquiring all decryption keys, it is very hard for hackers to crack the ciphertext. On the other hand, if a ciphertext is generated by using a combinatorial-logic block encryption technique [9], the sub-keys produced by the parent key given when the system starts up are the same, no matter how complicated the encryption process is. In fact, the same plaintext block will generate the same ciphertext block. In this case, hackers may crack the system by analyzing the relationship between plaintext blocks and the corresponding ciphertext blocks [8,10,11] or by using Brute-force attacking methods. Hence, due to the high speed of current computer systems, a combinatorial-logic block encryption technique may no longer be secure.

According to our study, both the DES and AES have the following disadvantages, including

1. Employing a combinational-logic encryption principle wherein the content of a ciphertext block, i.e., the output, is totally determined by the content of the current plaintext block, thus unrelated to the content of the previous plaintext blocks.

2. Encrypting fixed-size data blocks. This will reduce the flexibility of an encryption system. If the block size of an encryption algorithm can be changed flexibly, the encryption system is then able to encrypt data of variable lengths when necessary. This can thereby resist different kinds of attacks efficiently.
3. Performing its own core computation repeatedly. For example, the DES calculates its core computation 16 times, whereas the AES computes its own 10 times. Although each repeated computation uses a new sub-key, repetitious computation by using the same equation not only weakens its security level, but also lowers its performance.
4. Adopting a fixed substitution box (S-Box) to encrypt messages. This will also reduce its security level. If messages can be encrypted by using dynamic transition boxes, the security level of the underlying system will be higher since even if the same plaintext messages appear at different places, they will be encrypted with different transition boxes, consequently generating different ciphertext messages.

Therefore, to solve these drawbacks, in this study, we propose a new encryption approach, called the Secure Feedback Encryption Method (SeFEM for short), in which plaintext blocks are encrypted by using three security mechanisms, including a sequential-logic style encryption method, a three-dimensional operation and a dynamic transition box. With this sequential-logic style encryption method, the computational result of an encryption round R as a part of (R+1)'s inputs is fed back to the encryption mechanism, thus increasing the complexity and unpredictability of the generated ciphertext. The three-dimensional operation, referring to three different operators, including a binary addition ($+_2$) [12,13], exclusive-or ($\oplus$), and exclusive-and ($\odot$), is used to encrypt a plaintext block. A dynamic transition box nonlinearly rearranges the bits of an encrypted message. The purpose is to increase the encryption complexity so

as to reduce the probability of the encryption process being cracked by hackers.

The rest of this paper is organized as follows. Chapter 2 briefly introduces the DES and AES. Chapter 3 describes the SeFEM. Security analysis and comparison are presented and discussed in Chapter 4. Performance is analyzed and evaluated in Chapter 5. Chapter 6 concludes this paper and addresses our future research.

2. Background and Related Work

Block cipher refers to the process in which a fixed-length plaintext block is cryptographically manipulated by a series of operations to produce the corresponding secure ciphertext block, the length of which is often the same as that of the plaintext block.

2.1 Data Encryption Standard (DES)

The DES is a typical block cipher technique with 64 bits as its block size. But in practice, the keys used by this algorithm to encrypt plaintext blocks are only 56 bits in length [4,5]. The remaining 8 bits are parity bits or unused, implying that the security level of the generated ciphertext block falls short of expectation since a short key's security level is generally lower than a longer key's.

The DES encryption structure consists of the initial permutation (IP for short), 16 processing stages (called 16 rounds) and the final permutation (IP^{-1} for short), in which IP and IP^{-1} are mutually inverse arrays. Each of the 16 rounds contains a Feistel function [4,5], and an $\oplus$ operation.

Before the first round, a plaintext block (64-bit) follows the content of the given IP table to permute their bits. After that, the new 64-bit block is divided into two 32-bit sub-blocks. Let the right sub-block be $IP_{1,1}$ which is directly input to the first Feistel function, named round-1 Feistel which receives another input, called subkey1, to generate a result, denoted by $result_{1,1}$ (i.e., round1's 1^{st} result). Let the left sub-block be $IP_{1,2}$ which is exclusive-ored with $result_{1,1}$ to generate $result_{1,2}$ (i.e., round1's 2^{nd} result). Let $IP_{2,1} = result_{1,2}$ and let $IP_{2,2} = IP_{1,1}$. The rounds continue. The general rule is that round-i Feistel receives the two inputs, i.e., sub-key i and $IP_{i,1}$, with which to generate

$result_{i,1}$ which is then exclusive-ored with $IP_{i,2}$ to generate $result_{i,2}$. After that, $IP_{(i+1),2}=IP_{i,1}$ and $IP_{(i+1),1} = result_{i,2}$, for all i= 1, 2, …, 16. Lastly, $IP_{17,1}$ is the right half and $IP_{17,2}$ is the left half of the 64-bit result of round 16. The right and left halves are input to IP^{-1} to produce the 64-bit ciphertext.

The Feistel architecture [4] consists of four main functions, including expansion, key mixing, substitution, and permutation, respectively, denoted by E, $\oplus$, S (named S-Box) and P. Expansion transforms and extends a 32-bit pattern into 48 bits by using the expansion permutation [4,5]. The key mixing exclusive-ors E's output, i.e., the 48-bit output, and a 48-bit sub-key to generate a 48-bit result, which is divided into 8 6-bit patterns as the inputs of 8 S-Boxes. Each S-Box as a non-linear transformation mechanism transforms a 6-bit input to a 4-bit output, implying the output of the 8 S-Boxes is 32 bits long. After that, permutation rearranges the 32-bit output based on a fixed permutation process. The final result is also 32 bits in length.

2.2 Advanced Encryption Standard (AES)

The AES is also a kind of block cipher technique with block size 128 bits long. But its key length can be 128, 192 or 256 bits when necessary. The longer the length of the keys, the higher the security level of the system being considered. The AES uses a parent key to generate sub-keys. The AES encryption process is performed on a 4 × 4 matrix, e.g., M, in which an element is 8 bits in length. The initial M contains a plaintext block, i.e., 128 bits (=4×4×8) long. The AES encryption has 10 rounds. Each round, except the last one, comprises four stages.

In the first stage, named the SubBytes stage, an element of M, e.g., $a_{i,j}$, is substituted by its corresponding element $a'_{i,j}$ retrieved from a pre-generated table, called a Rijndael S-Box [8,14,16], the elements of which are produced beforehand by invoking

a non-linear function. In the second stage, called the ShiftRows stage, all elements of row r_i in M are left-rotated i times, $0 \leq i \leq 3$, even though the name of this stage is ShiftRows. The third stage is the MixColumns stage which linearly converts a column $(a_{0,i}, a_{1,i}, a_{2,i}, a_{3,i})^T$ of M, in which an element is one byte in length, to $(a'_{0,i}, a'_{1,i}, a'_{2,i}, a'_{3,i})^T$ by invoking the method of the Rijndael mix columns [14-16]. In fact, this stage invokes an "xtime" function [5,14], the inputs and outputs of which are all 1 byte in length. The function left shifts each input one bit with the least significant bit being filled by a 0. If the input's most significant bit before shift is 1, the shift result will be exclusive-ored with $\{1b\}_{hex}$. In the last stage, named the AddRoundKey stage, each $a_{i,j}$ in M is exclusive-ored with $k_{i,j}$ where $k_{i,j}$ is an element of a given round sub-key table used to convert $a_{i,j}$ to $a'_{i,j}$, $0 \leq i, j \leq 3$. In the AES, the parent key is employed by Rijndael's key schedule [14-16] to generate round sub-keys for each round.

2.3 Block Cipher Modes of Operation

The Cipher Block Chaining (CBC), the Propagating Cipher Block Chaining (PCBC), Cipher feedback (CFB), Output feedback (OFB) and Counter (CTR) [8] are block cipher standards recognized by National Institute of Standards and Technology (NIST). The five modes may be used in conjunction with any symmetric key block cipher algorithm approved by a Federal Information Processing Standard (FIPS) to increase the security level.

In the encryption process of the CBC [8,17], a plaintext block P_i is exclusive-ored with Initialization Vector (IV) or previous ciphertext block C_{i-1} before it is input to Block Cipher Encryption unit. The general rule of the PCBC's block encryption [8,17] is that plaintext is first exclusive-ored with IV. The exclusive-ored result and the key K of the system are then input to the Block Cipher Encryption unit to generate ciphertext

C_1. After that, the result of exclusive-oring plaintext P_{i-1} and ciphertext C_{i-1} is substituted for IV to exclusive-or with the next plaintext Pi, i=2,3,...,n where is the number of generated plaintext. The newly generated exclusive-ored result and K are then input to Block Cipher Encryption unit to generate the next ciphertext C_i.

With the CFB [8,17,18], we need an IV together with a Key K to trigger a Block Cipher Encryption unit. The output O_i of the unit is then exclusive-ored with a plaintext block P_i to produce the corresponding ciphertext block C_i. After that, CFB feeds back C_i to substitute for the IV to encrypt the next plaintext block P_{i+1}. The technical aspects of the OFB are very similar to those of the CFB. The only difference is that the OFB [8,17,18] feeds back the output of the Block Cipher Encryption unit O_i, rather than feeding back the ciphertext C_i, to the Block Cipher Encryption unit to encrypt the next plaintext block P_{i+1}. Furthermore, with the CTR [8,18], the feedback operation employed in the CFB and OFB is replaced by a counter as one of the inputs of the Block Cipher Encryption unit.

Although these types of block cipher provide the security system with data integrity and confidentiality protection, they are not safe enough to protect data, i.e., they are vulnerable to known plaintext-ciphertext cryptanalysis attacks [17,18].

In [17] and [18], some improved approaches, e.g. OPC, OPC-2, KSPC and ODC, were proposed. The general rule of the OPC-1 is that a key Key1 and previous intermediate output, e.g., G_{i-1}, are input to the Block Cipher Encryption unit to generate O_i, which is then exclusive-ored with plaintext P_i to produce G_i. Next, the G_i is binary-added with the previous output of the Block Cipher Encryption unit, e.g., O_{i-1}, to generate ciphertext C_i where i=1,2,3,…,n, G_0=IV and O_0=Key2. The general rule of the encryption process of the OPC-2 is that plaintext P_i and Key1 are input to the Block Cipher Encryption unit to generate O_i, which is then exclusive-ored with O_{i-1} to generate

C_i, where i=1,2,3,…,n, O_0=Key2.

With the KSPC, the key K is exclusive-ored with previous ciphertext, e.g., C_{i-1}, where the C_0=IV. The exclusive-ored result is then input to the Block Cipher Encryption unit to encrypt current P_i. The general rule of the encryption process of the ODC is that P_i and K are input to the Block Cipher Encryption unit to generate the output O_1, which is then binary-added with the exclusive-ored result of K and IV to generate Ci. After that, O_i is exclusive-ored with the P_{i+1}, and the exclusive-ored result and K are input to the Block Cipher Encryption unit to generate O_{i+1}, which is then binary-added with O_i to generate C_{i+1}, i=1,2,3,…,n, and C_0=IV.

In fact, the four modes proposed in [17] and [18] improved some of the shortcomings of the original Block Cipher modes of operation by using sequential logic-based feedback mechanisms. That is why we employ this mechanism to increase the security level of a protected system.

3. Feedback Encryption, Three Dimensional Operations and A Dynamic Transition Box

The parameters and functions employed in this study are defined below.

Plaintext block：p_i ,$1\leq i \leq n$, where n is the total number of blocks contained in the given plaintext

System keys：K_i ,$1\leq i \leq 8$

Dynamic keys：a_i, b_i, d_i, $1\leq i \leq n$

Dynamic feedback keys：a_{i-1}, b_{i-1}, d_{i-1}, $1\leq i \leq n$

Initial dynamic keys：$a_0 = K_9$, $b_0=K_{10}$, $d_0=K_{11}$

Encryption key：a_e

Ciphertext blocks：c_i, $1\leq i \leq n$

A document, i.e., plaintext, is divided into n blocks, each of which is *m* bits in length, i.e., $Plaintext = p_1p_2p_3...p_n$. If $|P_n| < m$, unoccupied bits are filled with zeros such that $|P_n|=m$. Thus, each P_i , $1\leq i \leq n$, is m bits in length, and a key of the system is also *m* bits long, where *m* is a multiple of 8.

3.1 Dynamic Transition Box

The transition boxes that we propose have two types, a mother transition box and a child transition box. If a data block to be encrypted by a security system is *m* bits long where *m* is a multiple of 8, then:

1). The mother transition box consists of *g* rows and *h* columns where $m=gh$, $2\leq g,h$. A sequence of numbers $1,2,3,\ldots,m-1,m$ is randomly generated and rearranged so as to produce a random number sequence, which as the initial contents of this box is then

sequentially input to the mother transition box, implying that there are $m!$ candidates of the mother transition box.

2). The child transition box is obtained by rotating the mother transition box clockwise (may also be counterclockwise) t times, where the value of the count variable t is determined by the feedback keys. We will show this later. Two examples of the 16-bit mother transition box and the child transition boxes generated by exchanging the elements of the mother transition are illustrated Figure 1.

3). The encryption process of a child transition box as shown in the upper half of Figure 2 moves the j^{th} bit of the plaintext block (or of a dynamic key) to the position specified by the content of the j^{th} position of the child transition box, e.g., k, i.e., to the k^{th} position of the ciphertext block, where $1 \leq k \leq m$, e.g., the 1^{st} bit b_0 (indexed by 00-00) and the 2^{nd} bit b_1 (indexed by 00-01) of the plaintext block are moved to the position specified by the content of the 1^{st} position (i.e., 1) and 2^{nd} position (i.e., 8) of the child transition box. The process terminates when all the bits of the plaintext block (or the dynamic key) are correctly moved to their positions.

4). The decryption process of a child transition box as illustrated in the lower half of Figure 2 moves the k^{th} bit of a ciphertext block to the j^{th} position of the plaintext block where k is the content of the j^{th} position of the child transition box, e.g., b'_0 and b'_1 are moved to the 8^{th} (indexed by 10-00) and 1^{st} (indexed by 00-00) position of the plaintext block, respectively. The process terminates upon the completion of the transition of all the bits in the ciphertext block.

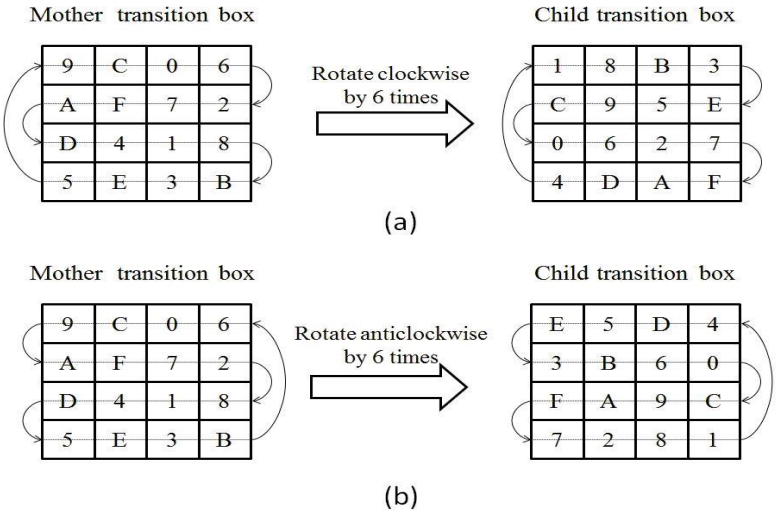

Figure 1. An embodiment of a 16-bit mother transition box and the generated child transition box

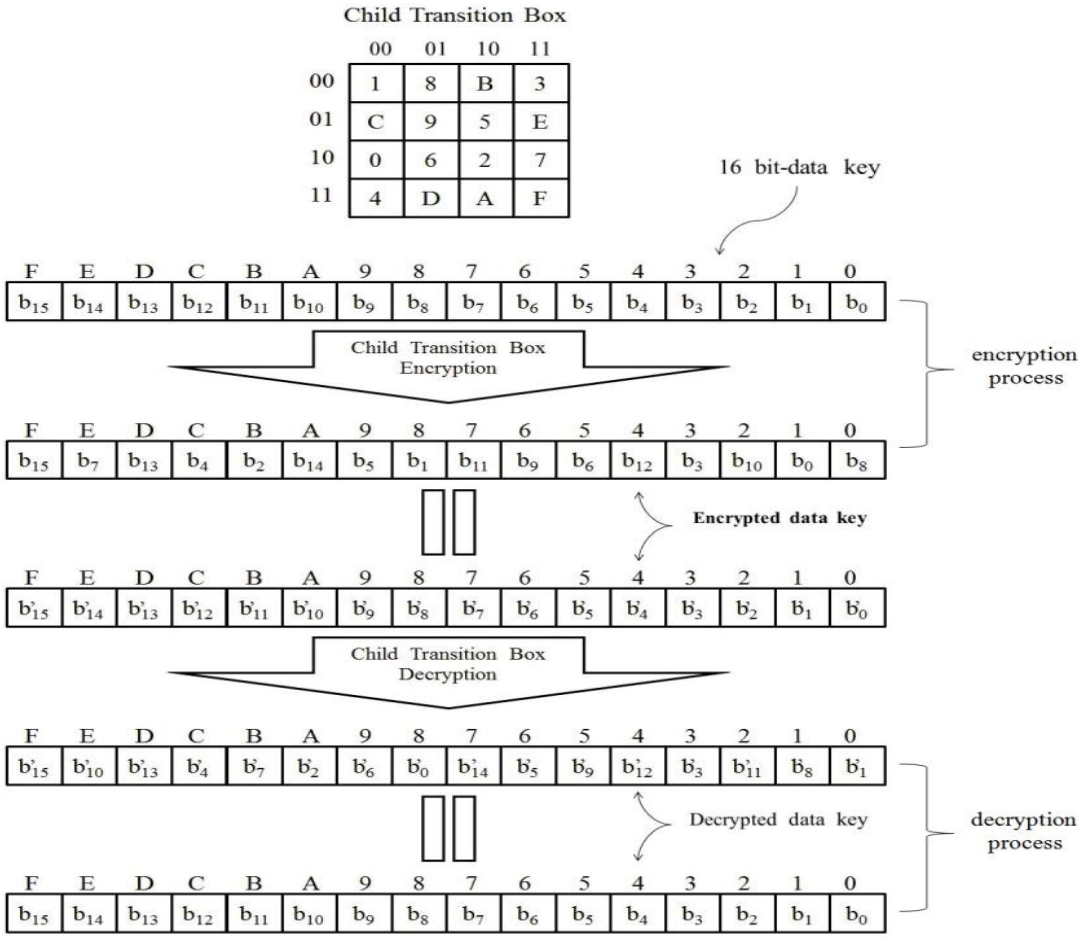

Figure 2. An embodiment of the encryption/decryption of a block by using the 16-bit child transition box

3.2 Encryption

The encryption process of the SeFEM is shown in Figure 3.

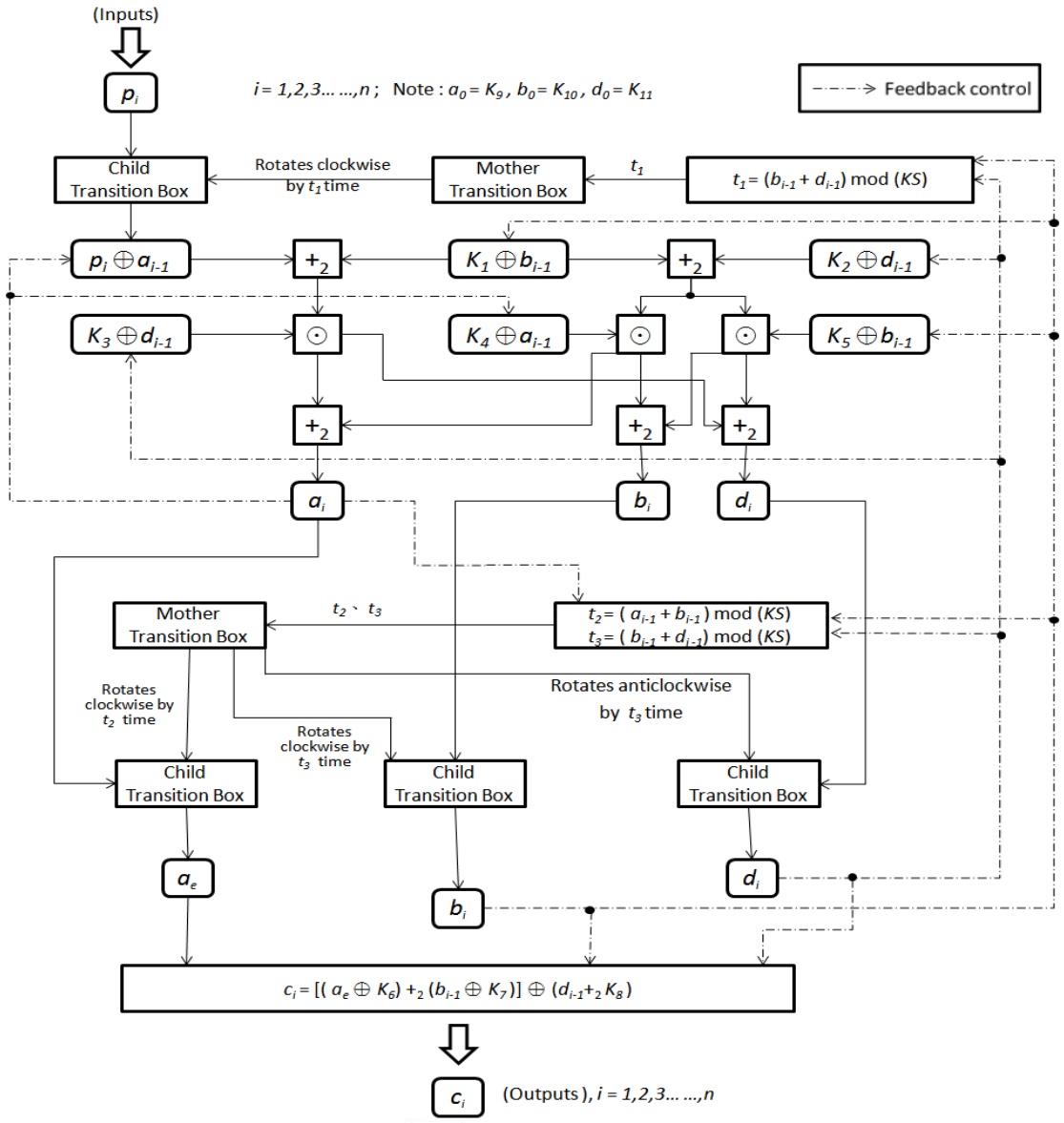

Figure 3. The encryption flow chart of the SeFEM where p_i is the plaintext block i, i =*1,2,… …n*; $a_0=K_9$, $b_0=K_{10}$; $d_0=K_{11}$; *KS*:key size

Step 1: Deriving an encrypted p_i-key from the plaintext block p_i, $1 \le i \le n$.

1). Input the plaintext block p_i, $1 \le i \le n$;

2). Calculate parameter $t_1 = (b_{i-1}+d_{i-1}) \bmod KS$, $1 \le i \le n$, (1)

where *KS* stands for key size;

3). Rotate the mother transition box clockwise by t_1 times to obtain the first child transition box;

4). Perform the encryption process by applying the first child transition box to p_i to generate the encrypted p_i-key, also denoted by p_i, for later use.

Step 2: Generating the dynamic keys a_i, b_i and d_i, $1 \le i \le n$

To simplify the following description, several notations are created, including:

$A=p_i \oplus a_{i-1}$, $B=K_1 \oplus b_{i-1}$, $C=K_2 \oplus d_{i-1}$, $D=K_3 \oplus d_{i-1}$, $E=K_4 \oplus a_{i-1}$, $F=K_5 \oplus b_{i-1}$

1). Calculate: $a_i=[(A +_2 B) \odot D] +_2 [(B +_2 C) \odot E]$, (2)

$b_i=[(B +_2 C) \odot E] +_2 [(B +_2 C) \odot F]$, (3)

$d_i=[(B +_2 C) \odot F] +_2 [(A +_2 B) \odot D]$ (4)

2). Calculate parameters $t_2= (a_{i-1} + b_{i-1}) \bmod KS$, (5)

and $t_3= (a_{i-1} + d_{i-1}) \bmod KS$; (6)

3). Rotate the mother transition box clockwise t_2 times to generate the second child transition box, and then perform the encryption process by applying this child transition box to the dynamic key a_i, obtained by invoking Eq.(2), to generate the encryption key a_e;

4). Rotate the mother transition box clockwise t_3 times to generate the third child transition box, and then perform the encryption process by applying this child transition box to the parameter b_i, obtained by invoking Eq.(3), to generate the dynamic key, also denoted by b_i;

5). Rotate the original mother transition box anticlockwise t_3 times to generate the fourth child transition box, and then perform the encryption process by applying this child transition box to the parameter di, obtained by invoking Eq.(4), to generate the dynamic key, still denoted by d_i;

Step 3: Outputting the ciphertext block c_i, $1 \le i \le n$

1). Calculate $c_i = [(a_e \oplus K_6) +_2 (b_{i-1} \oplus K_7)] \oplus (d_{i-1} +_2 K_8)$, $1 \le i \le n$, (7)

2). Output the ciphertext block c_i, $1 \le i \le n$

In the encryption process of the SeFEM, the parameters b_{i-1}, d_{i-1} and a_e, rather than b_i, d_i and a_i, are invoked to generate c_i, implying that b_i, d_i and a_i are internally used in the encryption/decryption processes. Hence, hackers are unable to infer the dynamic feedback keys a_{i-1}, b_{i-1} and $d_{i-1}$from the dynamic keys a_i, b_i and d_i. Therefore, a_{i-1}, b_{i-1} and d_{i-1} are very secure, and they are also changed continuously and dynamically to raise the security level of c_i. That is why we dare to say that the SeFEM's encryption process is more secure than those of the conventional feedback control mechanisms [8,9,11].

3.3 Decryption

In fact, the SeFEM can be installed in a single machine to encrypt a stored file F, and decrypt the file when users wish to retrieve F. It can also be employed to encrypt a file H which needs to be delivered to and be decrypted at the receiving site. In both cases, the encryption process and decryption process have to both keep K_1~K_{11}.

The decryption process of the SeFEM is as follows.

Step 1: Restoring the dynamic key a_i, $1 \le i \le n$.

1). Input the ciphertext block c_i, $1 \le i \le n$;

2). Restore the encryption key a_e where

$$a_e = \begin{cases} [[c_i \oplus (d_{i-1} +_2 K_8)] - (b_{i-1} \oplus K_7)] \oplus K_6, & if \ \ c_i \oplus (d_{i-1} +_2 K_8) \ge (b_{i-1} \oplus K_7) \\ [[c_i \oplus (d_{i-1} +_2 K_8)] + \overline{(b_{i-1} \oplus K_7)} + 1] \oplus K_6, & if \ \ c_i \oplus (d_{i-1} +_2 K_8) < (b_{i-1} \oplus K_7) \end{cases} \quad (8)$$

3). Calculate parameter t_2 where $t_2 = (a_{i-1} + b_{i-1}) \bmod KS$;

4). Rotate the mother transition box clockwise t_2 times to generate the second child transition box, and then perform the decryption process by applying the child

transition box to the encryption key a_e to restore the dynamic key a_i.

Step 2: Restoring the dynamic key b_i and d_i, $1 \le i \le n$.

To simplify the following description, several notations are created, including:

$G = (B +_2 C) \odot E$, $H = (a_i - G) \odot D$, $L = (a_i + \overline{G} + 1) \odot D$, then

1). Restore the encrypted p_i-key

$$p_i = \begin{cases} [[(a_i - G) \odot D] - B] \oplus a_{i-1}, & if \ \ a_i \ge G \ \text{and} \ \ H \ge B \\ [[(a_i - G) \odot D] + (\overline{B} + 1)] \oplus a_{i-1}, & if \ \ a_i \ge G \ \text{and} \ \ H < B \\ [[(a_i + \overline{G} + 1) \odot D] - B] \oplus a_{i-1}, & if \ \ a_i < G \ \text{and} \ \ L \ge B \\ [[(a_i + \overline{G} + 1) \odot D] + (\overline{B} + 1)] \oplus a_{i-1}, & if \ \ \mathrm{a_i} < G \ \text{and} \ \ L < B \end{cases}; \quad (9)$$

2). Calculate parameters:

$b_i = [(B +_2 C) \odot E\ ] +_2 [\ (B +_2 C) \odot F\ ]$ (10)

$d_i = [(B +_2 C) \odot F\ ] +_2 [\ (A +_2 B) \odot D\ ]$ (11)

3). Calculate parameter $t_3 = (a_{i-1} + d_{i-1}) \bmod KS$; (12)

4). Rotate the mother transition box clockwise t_3 times to generate the third child transition box, and then perform the decryption process by applying this child transition box to parameter b_i, obtained by applying Eq.(10), to generate the dynamic key b_i for the next round.

5). Rotate the mother transition box anticlockwise t_3 times to generate the fourth child transition box, and then perform the decryption process by applying this child transition box to parameter d_i, obtained by invoking Eq.(11), to generate the dynamic key d_i for the next round.

Step 3: Restoring the plaintext block p_i, $(1 \le i \le n)$

1). Calculate parameter $t_1 = (b_{i-1} + d_{i-1}) \bmod KS$

2). Rotate the mother transition box clockwise t_1 times to generate the first child transition box, and then perform the decryption process by applying this child

transition box to the encrypted p_i-key to restore plaintext block p_i, $(1 \le i \le n)$.

3.4 Binary Adder

In the SeFEM's encryption process, we employ a binary adder, which as a binary operator with two parameters, e.g., A and B, i.e., $A+_2B$, is different from XOR in that a binary adder does not numerically restore B to its original value when A is added twice, e.g., $A+_2B+_2A \neq B$, but $A \oplus B \oplus A = B$, i.e., A disappears from the exclusive-ored result.

Given a plaintext block *p*, a ciphertext block *c* and a dynamic key *K*, the binary adder $+_2$ is defined as follows.

Encryption: $c = p+_2K$, where *p* and *K* undergo binary addition, and ignore the carry generated by the addition of the most significant bits;

Decryption: $p = c -_2 K = \begin{cases} c - K, & if \ \ c \ge K \\ c + \overline{K} + 1, & if \ \ c < K \end{cases}$, (13)

where $-_2$ as a binary subtraction is the inverse operation of $+_2$, and $\overline{K}$ is the one's complement of key *K*.

The drawback of the binary adder is that its operational speed is a little lower than that of the XOR. The encryption algorithm of the binary adder with the two streams, A and B, i.e., $C = A +_2 B$, of n (=128) bits long is shown in Figure 4. The decryption algorithm of $C = A -_2 B$ is illustrated in Figure 5.

Algorithm 1: Encryption process of the binary adder

Input: Streams A and B

Output: C (=A $+_2$B)

{Let A=A[128]A[127] … A[2]A[1], B=B[128]B[127] … B[2]B[1], C=C[128]C[127] …
C[2]C[1] and
carry= carry[128]carry[127] … carry[1]carry[0] where each of A[i], B[i], C[i] and carry[i] is a binary digit, $1 \le i \le 128$; carry[0]=0;

For i = 1 to 128 {

If carry[i-1] = 0 then

If A[i] = B[i] then

If A[i] = 0 then {C[i] = 0; carry[i] = 0;} /*A[i]+B[i]+carry[i-1] = $0+_20+_20 = 00_2$*/

else {C[i] = 0; carry[i] = 1;} /*A[i]+B[i]+carry[i-1] = $1+_21+_20 = 10_2$*/

else /*A[i]≠B[i] */ {C[i] = 1; carry[i] = 0;}

/*A[i]+B[i]+carry[i-1] = $(1+_20+_20)$ or $(0+_21+_20) = 01_2$*/

else /*carry[i-1]=1*/

If A[i] = B[i] then

If A[i] = 0 then {C[i] = 1; carry[i] = 0;}

/*A[i]+B[i]+carry[i-1] = $0+_20+_21 = 01_2$*/

Else /*A[i]=B[i]=1*/ {C[i] = 1; carry[i] = 1;}

/*A[i]+B[i]+carry[i-1] = $1+_21+_21 = 11_2$*/

else /*A[i]≠B[i]*/

{C[i] = 0; carry[i] = 1;} /*A[i]+B[i]+carry[i-1] = $(1+_20+_21)$ or $(0+_21+_21) = 10_2$*/}}

Figure 4 The encryption algorithm of the SeFEM

Algorithm 2: Decryption process of the binary adder

Input: Streams A and B

Output: C (=A $-_2$B)

{Let A=A[128]A[127] … A[2]A[1], B=B[128]B[127] … B[2]B[1], C=C[128]C[127] …
C[2]C[1] and
carry= carry[128]carry[127] … carry[1]carry[0] where each of A[i], B[i], C[i] and carry[i] is a binary digit, $1 \le i \le 128$; carry[0]=0;

For i = 1 to 128 {

If carry[i-1] = 0 then

If A[i] = B[i] then

{C[i] = 0; carry[i] = 0;}

/*A[i] $-_2$ B[i] $-_2$ carry[i-1] = (1 $-_2$ 1 $-_2$ 0) or (0 $-_2$ 0 $-_2$ 0) = 00_2*/

else If A[i] = 1 then {C[i] = 1; carry[i] = 0;}

/*A[i] $-_2$ B[i] $-_2$ carry[i-1] = 1 $-_2$ 0 $-_2$ 0 = 01_2*/

else {C[i] = 1; carry[i] = 1;} /*A[i] $-_2$ B[i] $-_2$ carry[i-1] = 0 $-_2$ 1 $-_2$ 0 = 11_2*/

else /*carry[i-1]=1*/

If A[i] = B[i] then

{C[i] = 1; carry[i] = 1;}

/* A[i] $-_2$ B[i] $-_2$ carry[i-1] = (1 $-_2$ 1 $-_2$ 1) or (0 $-_2$ 0 $-_2$ 1) = 11_2*/

else If A[i] = 1 then {C[i] = 0; carry[i] = 0;}

/*A[i] $-_2$ B[i] $-_2$ carry[i-1] = 1 $-_2$ 0 $-_2$ 1 = 00_2*/

else {C[i] = 0; carry[i] = 1;}/*A[i] $-_2$ B[i] $-_2$ carry[i-1] = 0 $-_2$ 1 $-_2$ 1 = 10_2*/}}

Figure 5 The decryption algorithm of the SeFEM

4. Security Analysis and Comparison

A well-designed encryption mechanism must be one with a high security level so as to effectively protect a system from being attacked by hackers, and with high performance and a low cost in order to efficiently encrypt and decrypt data [19]. In the following, we will analyze the security of the SeFEM and compare it with the AES and DES cryptographic methods.

4.1 Encryption Complexity of the Dynamic Transition Box

The child transition box as a dynamic transition box stated above is used to nonlinearly rearrange the bits of the encrypted message. What is the probability *p* of recovering the original message from the corresponding encrypted message by using the child transition box? Lemma 1 shows the answer.

Lemma 1:

Assume that key A is *m* bits in length, and there are *n* 1's and (*m*-*n*) 0's in this key, $n \le m$. Then the probability *p* of recovering the original key A from the encrypted key A by applying the child transition box is $p = \frac{1}{\binom{m}{n}}$.

Proof: Since the encrypted key A is nonlinearly rearranged, the number of all possible arrangements of this key is $C_n^m = \frac{m!}{n!(m-n)!}$.

If the child transition box is unknown to hackers, since the encrypted key A will be one of the possible arrangements, the probability *p* of recovering the original key A from the encrypted key A on one trial is $p = \frac{1}{\frac{m!}{n!(m-n)!}} = \frac{1}{\binom{m}{n}}$. Q.E.D.

In this study, the key a_e is unknown to hackers, i.e., the numbers of 1's and 0's in key a_e are unknown to hackers. Then the number of all possible nonlinear arrangements of the encrypted key a_e is $\binom{m}{0}+\binom{m}{1}+\binom{m}{2}+...+\binom{m}{m-1}+\binom{m}{m}=2^m$.

Hence, the probability p of recovering dynamic key a_i from the encrypted key a_e on one trial is $p=\frac{1}{2^m}$.

4.2 Complexity of the Three Dimensional Operations

Let X and Y be two keys, each of which is m bits in length. The probability p of recovering the value of an (X, Y) pair from an illegally intercepted $X \oplus Y$ is $p=\frac{1}{2^m}$ [12]. But what is the probability of recovering of (X, Y) from $X \odot Y$ (or from $X +_2 Y$)? Lemma 2 (Lemma 3) will show the results.

Lemma 2:

Since the keys X and Y are both m bits in length, the probability p with which we can obtain a correct X and Y on one trial from an illegally intercepted $X \odot Y$ is $p=\frac{1}{2^m}$.

Proof: Let $X = x_m...x_2x_1$, $Y = y_m...y_2y_1$ and let $Z = X \odot Y = z_m...z_2z_1$ where each of x_i, y_i and z_i is a binary digit, and $z_i = x_i \odot y_i$, $1 \le i \le m$. If $z_i = 0$, the possible value of an (x_i,y_i) pair is (0,1) or (1,0). Otherwise, the possible value is (0,0) or (1,1). Hence, when z_i is known, no matter whether it is 0 or 1, for each i, $1 \le i \le m$, the probability of obtaining the correct (x_i,y_i)pair on one trial is $\frac{1}{2}$, and then the probability to correctly recover the original value of (X, Y) on one trial is $\left(\frac{1}{2}\right)^m=\frac{1}{2^m}$. Q.E.D.

Lemma 3:

Let X and Y be two keys, each of which is m bits in length. The probability p of recovering the value of an (X, Y) pair on one trial from an illegally intercepted $X+_2Y$ is $p = \frac{1}{2^m}$.

Example 1: Let X,Y and Z be three keys, each of which is 4 bits in length and $Z=X+_2Y$. The possible pairs of (X, Y) such that Z = 1001 are listed in Table 1.

Table 1. The possible pairs of (X, Y) where $Z = 1001 = X +_2 Y$

Without carry		With carry	
X	Y	X	Y
0000	1001	1111	1010
0001	1000	1110	1011
0010	0111	1101	1100
0011	0110	1100	1101
0100	0101	1011	1110
0101	0100	1010	1111
0110	0011		
0111	0010		
1000	0001		
1001	0000		

Proof: Let $Z = X+_2Y$. Then there are two cases for the binary addition of the most significant bits, i.e., with carry and without carry.

Case 1: If the situation of without carry occurs, then $Z= X+_2Y$ is equivalent to $Z= X+Y$, and the possible values of (X, Y) are $(0, Z)$, $(1, Z-1)$, $(2, Z-2)$, …, $(Z-1, 1)$ and $(Z, 0)$, i.e., a total of $Z+1$ possible values.

Case 2: If the situation of with carry occurs, since we ignore this carry, $Z= X+_2Y$ is equivalent to $Z=X+Y-2^m$, and the possible values of (X, Y) are $(2^m-1, Z+1)$, $(2^m-2, Z+2)$, $(2^m-3, Z+3)$,..., $(Z+2, 2^m-2)$ and $(Z+1, 2^m-1)$, i.e., a total of 2^m-Z-1

possible values after ignoring the carry. Hence, for each Z, there is a total of $(Z+1) + (2^m-Z-1) = 2^m$ possible values of (X, Y) such that $Z=X+_2Y$. The probability p of recovering the original value of (X, Y) on one trial is $p = \frac{1}{2^m}$.

Q.E.D.

Further, before we discuss the relationship between p_i and c_i for each i, $1 \le i \le n$, the relationship between a_e and c_i for each i, $1 \le i \le n$, should be addressed first. Lemma 4 will illustrate the probability of recovering the value of a_e from c_i.

Lemma 4:

Assume the encrypted key a_e and the ciphertext block c_i for each i, $1 \le i \le n$, are m bits in length. Then the probability p for recovering the original value of a_e from c_i on one trial for each i, $1 \le i \le n$, is $p = \frac{1}{2^m}$.

Proof: According to Eq.(7),

$$\begin{aligned} c_i &= [(a_e \oplus K_6) +_2 (b_{i-1} \oplus K_7)] \oplus (d_{i-1} +_2 K_8), \ 1 \le i \le n \\ &= [(a_e \oplus K_6) +_2 b'_{i-1}] \oplus d'_{i-1}, \end{aligned} \quad (14)$$

where $b'_{i-1}=b_{i-1} \oplus K_7$ and $d'_{i-1} = d_{i-1} +_2 K_8$. In Eq.(14), two inverse operations of $\oplus$ and one inverse operation of $+_2$ are required to obtain $(a_e, K_6, b'_{i-1}, d'_{i-1})$ from c_i. By lemma 3 and [12], due to applying the two operators a total of three times, the probability p of recovering the original values of $(a_e, K_6, b'_{i-1}, d'_{i-1})$ from c_i on one trial is $p = \left(\frac{1}{2^m}\right)^3 = \frac{1}{8^m}$.

However, $\frac{1}{8^m} << \frac{1}{2^m}$, in which $\frac{1}{2^m}$ is the probability of recovering the original value of a_e from c_i for each i, $1 \le i \le n$, on one trial, and which is also the least probability of recovering a_e from c_i with a blind guess. If we want to obtain a_e by analyzing c_i, the values of K_6, b'_{i-1}, d'_{i-1} are required. However, to recover $(a_e, K_6, b'_{i-1}, d'_{i-1})$ from c_i, based on Eq.(7), the recovering probability is $\frac{1}{8^m}$ which is far smaller than $\frac{1}{2^m}$, where

$\frac{1}{2^m}$ is the probability of recovering a_e from c_i by using a blind guess. Hence, the probability p of recovering the original value of a_e from c_i on one trial for each i, $1 \le i \le n$, is $p = \frac{1}{2^m}$. Q.E.D.

Let $x_i \xrightarrow{p} y_i$ be the notation of the probability p of recovering y_i from x, for each i. For example, the probability of recovering a_e from c_i for each i, $1 \le i \le n$, is $\frac{1}{2^m}$, which can be denoted as $c_i \xrightarrow{1/2^m} a_e$. Similarly, according the flow chart of the SeFEM shown in Figure 3, there exist $a_e \xrightarrow{1/2^m} a_i$, $a_i \xrightarrow{1/2^m} p_i$, and $p_i \xrightarrow{1/2^m} p_{i,original}$, implying that $c_i \xrightarrow{1/2^m} p_{i,original}$ is the probability of recovering $p_{i,original}$ from c_i for each i, $1 \le i \le n$, is $\frac{1}{2^m}$.

4.3 Cryptanalysis of Attacks

4.3.1. Cryptanalysis on known plaintext and the corresponding ciphertext attacks

To analyze the relationship between a plaintext block and its ciphertext block of a fixed-length-data-block system, hackers may first collect the ciphertext blocks and the corresponding plaintext blocks of the system. When receiving a ciphertext block, they can look up the corresponding plaintext block from the collected (plaintext block, ciphertext block) pairs by employing parallel computing techniques.

However, as fixed-length-data-block encryption/decryption systems, the DES and AES at maximum have 2^{64} and 2^{128} possible plaintext blocks, respectively, indicating that currently it is hard for hackers to crack the AES with this method. But it may be cracked in the near future. The SeFEM is a feedback control mechanism. This type of attack can only be applied to the provisions of the first ciphertext block, but is not

applicable to crack the consequent ciphertext blocks, since these ciphertext blocks are produced by using a sequential-logic style encryption method which generates different keys to encrypt different plaintext blocks, and cannot be decrypted by a parallel manner. So the SeFEM can more effectively protect a system from the known plaintext/ciphertext attack than the DES and AES can.

4.3.2. Differential and linear attacks

Both differential attack [20,21] and linear attack [20,21] have the following characteristics:

1). Hackers need to collect a very large number of (plaintext block, ciphertext block) pairs of a cryptographic system before attacking this system; 2). As the targets of these attacks, the DES's and AES's sub-keys are derived from a single parent key; 3). The DES and AES use static internal S-Boxes to encrypt all the delivered messages.

For each of the two systems, the ciphertext blocks collected in the (plaintext block, ciphertext block) pairs are all derived from the same parent key and S-Box. So it is relatively more easy for hackers to crack the parent key and S-Box by analyzing the relationship between these plaintext blocks and ciphertext blocks collected in the (plaintext block, ciphertext block) pairs. However, if a large number of (plaintext block, ciphertext block) pairs is generated by invoking system keys and different transition boxes, rather than by using a single parent key and a fixed S-Box, then it is much harder for hackers to solve the system keys and transition boxes.

That is why the SeFEM employs three dynamic keys (i.e., a_i, b_i, and d_i), three dynamic feedback keys (i.e., $a_{i\text{-}1}$, $b_{i\text{-}1}$, and $d_{i\text{-}1}$), eight system keys ($K_1 \sim K_8$) and three initial keys (a_0, b_0, d_0), to protect an information system. In fact, a total of 11 independent keys, including $K_1 \sim K_8$, a_0, b_0, and d_0, and four dynamic child transition

boxes are invoked by the SeFEM.

The differential attack and linear attack on the SeFEM are analyzed as follows:

1. The DES and AES are a combinatorial-logic style encryption mechanism so that they suffer the differential attack and linear attack when hackers utilize parallel decryption processing techniques. However, the SeFEM is a sequential-logic style encryption mechanism. Hackers cannot exploit parallel decryption processing techniques for trying to crack the SeFEM.
2. While encrypting plaintext blocks, due to using the sequential-logic style mechanism, the three dynamic feedback keys of the SeFEM change continuously and dynamically. That is why it can defend against these two attacks. Moreover, if a plaintext block is 64 (128) bits in length, the hackers need to analyze 2^{64x11} (2^{128x11}) bits before they can crack the 11 independent keys. This is almost an impossible mission when the sequential-logic style encryption approach is used to generate encryption keys.
3. The same plaintext blocks appearing at different positions of the plaintext will be encrypted by different dynamic transition boxes to make the SeFEM more secure than the DES and AES since the latter two employ static S-Boxes. Although there are only 128 (256) different transition boxes that the SeFEM may produce, when the boxes are 64 (128) bits long, it is difficult for hackers to identify which dynamic transition box is currently being used to encrypt a plaintext block. Hence, the deployment of dynamic transition boxes can effectively resist differential attack and linear attack.

4.4 Flexibility

Both the DES's and AES's data blocks are fixed in length, which dramatically reduces the flexibility of an encryption system. If the encryption system can encrypt data of

different lengths, i.e., the size of a data block can be flexibly varied, then the system can more effectively resist possible attacks than the DES and AES can.

To overcome the disadvantages of the DES and AES, i.e., those inherent to the encryption of fixed-size data blocks, the SeFEM enables the encryption of flexible-size data blocks in the situation where the plaintext blocks to be encrypted and those encryption parameters and mechanisms, including system keys, initial keys, dynamic keys, dynamic feedback keys, encryption key, dynamic transition boxes, and the resultant ciphertext blocks, need to be the same size. In fact, the SeFEM can flexibly be adopted by different encryption systems, particularly those most suitable for mobile systems. If the encryption system needs to extend the size of a data block to increase its security level, the SeFEM is still applicable. But the DES and AES are not, implying that the SeFEM is more flexible for use than the DES and AES.

4.5 Comparison

The DES was cracked due to its short key length (only 56 bits long), rather than owing to its algorithmic flaws. Although the key length of the AES is 128 bits, parallel computing technologies have been developed rapidly, and the AES may someday be cracked. Table 2 summarizes the features of the DES, AES and SeFEM.

Table 2. Summary of the features of the DES, AES and SeFEM

Method / Characteristics	DES	AES	SeFEM	Remark
Algorithm	Excellent	Excellent	Excellent	Up to the present, the algorithm has not been cracked by hackers.
Operation structure	Good	Good	Excellent	The DES and AES are combinatorial-style encryption mechanisms, whereas SeFEM is a sequential-logic style encryption mechanism.
Flexibility	Low	Low	High	—
Parallel decryption	Yes	Yes	No	—
Security of key(s)	Low	Middle	High	SeFEM has 11 system keys
Security of transition box	Low	Middle	High	The transition boxes of the DES and AES are fixed in length, whereas the SeFEM's is dynamic.
Known plaintext/ ciphertext attack	Low	Middle	High	Security level for defending the attack
Differential attack	Low	Middle	High	Security level for defending the attack
Linear attack	Middle	High	High	Security level for defending the attack
Security	Low	Middle	High	—

5. Performance Analysis

In this chapter, we simulate and analyze the en/decryption processes of the SeFEM and evaluate their performance. Table 3 lists the specifications of the simulation environment.

Table 3. Specifications of the experimental platform

Component	*Description*
CPU	Intel(R) Core(TM) Q9400 2.66GHz
RAM	3 GB
OS	Windows 7
Programming tools	Java 1.7.0_13/ Eclipse

5.1 Binary Adder Simulation

From the algorithms shown in Figures 4 and 5, we can see that their operation processes are almost the same. So the costs of performing the encryption and decryption of the binary adder are the same. The costs of performing XOR, XAND and en/decryption of the binary adder are listed in Table 4, in which the costs of the binary adder and the inverse-binary adder are each only about two times that of XOR, implying that the binary adder is practically feasible.

Table 4. Costs of performing XOR, XAND, the binary adder and the Inverse-binary adder when the operand length = 128 bits

Operator	XOR ($\oplus$)	XAND ($\odot$)	Binary adder ($+_2$)	Inverse- Binary adder ($-_2$)
Execution time(*us*)	5.376	5.377	10.369	10.369

5.2 System Simulation Results

The performance of an encryption method heavily depends on the amount of computations that the method has. All computations in terms of different numbers of operations for the DES, AES and SeFEM in detail are listed in Table 5, which also shows that the total load and cost of the AES are much higher than those of the SeFEM, implying that the SeFEM is qualitatively more efficient than the AES. Hence the ranking of the performance of the three security mechanisms is SeFEM > AES > DES. The quantitative analysis on the performance of the AES and SeFEM will be a part of our future studies.

The implementation of the SeFEM is shown in the Appendix of this thesis.

Table 5. All computations in terms of different numbers of operations of the encryption/decryption processes of the DES, AES and SeFEM in detail

Scheme	Encryption	Decryption
DES (64-bit block) [4,6]	16 ⊕s (32 bits) + 16 ⊕s (48 bits) + 1 IP (64 bits) + 1 IP-1 (64 bits) + 128 S-Box (6 bits) + 16 Expansions (48 bits) + 16 Permutations (32 bits)	The number of operations is the same as that of the encryption process.
AES (128-bit block, 128-bit key) [16]	(AddRoundKey) 176 ⊕s (8 bits)	The number of operations is the same as the sum of the numbers of those operations employed by the encryption process for the three stages, including AddRoundKey, SubBytes, and ShiftRows
	(SubBytes) 160 Substitutions (8 bit) [22]	
	(ShiftRows) 30 ShiftRows (128 bit)	
	(MixColumns) 36 Rijndael columns mixing [15] (128 bits)	(MixColumns) 36 Rijndael columns mixing [15] (128 bits). (Generally, the operations of a decryption process are often more complex than those of the corresponding encryption

		process.)
SeFEM (128-bit block)	9 $\oplus$s (128 bits) + 3 $\odot$s (128 bits)+ 7 $+_2$s (128 bits) + 3 ts' generation, i.e., t_1, t_2 and t_3 (8 bits) + 4 child boxes' generation (128 bits) + 4 permutations (128 bits)	6 $\oplus$s (128 bits)+ 3 $-_2$s (128 bits) + 3 $\odot$s (128 bits)+ 7 $+_2$s (128 bits) + 3 ts' generation, i.e., t_1, t_2 and t_3 (8 bits) + 4 child boxes' generation (128 bits) + 4 permutations (128 bits)

6. Conclusions and Future Work

The encryption architectures of the DES and AES are similar, i.e., a combinational-logic encryption mechanism. So they may be decrypted by using parallel computing techniques. But the SeFEM's is a sequential-logic encryption approach. It is almost impossible for hackers to solve it with parallel techniques. Both the DES and AES utilize a single parent key to generate sub-keys, with which to encrypt messages. So they can more easily be cracked. The SeFEM employs 11 independent system keys to encrypt messages, making it very difficult to be cracked. Also, the transition boxes invoked by the DES and AES are static, while those employed by the SeFEM are dynamic. Of course, the latter is more secure than the former two. Further, both the DES and AES encrypt fixed-size data blocks, while the SeFEM's is flexible, thus more suitable for use by mobile systems than the DES and AES. Each of the DES and AES has only one encryption operator, i.e., exclusive-or. But the SeFEM employs three. It means that the DES and AES adopt a one-dimensional key-exchange operation and the SeFEM utilizes a three-dimensional one so that the SeFEM is more difficult to be cracked than the DES and AES are. Also, the DES and AES encryption techniques repeatedly perform their own core operations. But the SeFEM has no duplicated encryption computation, making its own encryption process more efficient than those of the DES and AES. Table 5 shows that the qualitative performance ranking of the DES, AES and SeFEM is SeFEM > AES > DES [23].

Moreover, if the SeFEM requires a parent key before the encryption process begins, we can substitute for the initial feedback key d_0 with the parent key. Then the dynamic key d_i and the dynamic feedback key d_{i-1} will change as d_0 is replaced by the input parent key. Furthermore, the keys d_i and d_{i-1}, due to the effect of the feedback

mechanism, change their values continuously, consequently greatly increasing their security levels.

Compared to existing data encryption methods, the SeFEM is a novel one, not only because it uses the sequential-logic style encryption mechanism, three-dimensional operations and dynamic transition boxes, but also more importantly it hides its dynamic keys b_i, d_i, and a_i which are internally used to generate those encryption parameters employed in both the encryption and decryption processes of the security system, and thus are invisible to hackers. That is why its sequential-logic style encryption mechanism is more secure than conventional ones.

In the SeFEM, ciphertext generated after the first round will be protected by the sequential-logic style encryption mechanism. However, because it is unprotected by this encryption mechanism, hackers have the opportunity to break the ciphertext generated in the first round by collecting and analyzing a large number of the first ciphertext block. However, in an actual breaking process, it is almost impossible for hackers to crack the security system by only collecting and analyzing the first (plaintext block, ciphertext block) pair. In fact, the opportunity of cracking the first ciphertext block does exist in the SeFEM. Let the first-stage ciphertext be the message directly output from the SeFEM, when the corresponding plaintext is input. Thus a mechanism which can effectively prevent hackers from collecting (plaintext block, first-stage ciphertext block) pairs, especially the first (plaintext block, first-stage ciphertext block) pair, is required. Also, we would like to analyze the quantitative performance of the AES, DES and SeFEM so that users can know the time that the three systems require to encrypt different types of input data. These constitute our future studies.

References

[1] Y.L. Huang and F.Y. Leu, "Constructing a Secure Point-to-Point Wireless Environment by Integrating Diffie-Hellman PKDS RSA and Stream Ciphering for Users Known to Each Other," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 2, no. 3, September 2011, pp. 96-107.

[2] S.M. Lee, D.S. Kim and J.S. Park, "A Survey and Taxonomy of Lightweight Intrusion Detection Systems," *Journal of Internet Services and Information Security*, vol. 2, issue 1/2, February 2012, pp. 119-131.

[3] S.K. Pandey and R. Barua, "Efficient Construction of Identity Based Signcryption Schemes from Identity Based Encryption and Signature Schemes," *Journal of Internet Services and Information Security*, Vol. 1, No.2/3, August 2011, pp. 161-180.

[4] J. Katz and Y. Lindell, Introduction to Modern Cryptography, Chapman & Hall/CRC Press, 2008.

[5] C.H. Yang, Network Security: Theory and Practice, Xbook Marketing Co. Ltd., September, 2008.

[6] M. Bellare and P. Rogaway, *Introduction to Modern Cryptography*, chapter 3, May 11, 2005.
http://digidownload.libero.it/persiahp/crittografia/2005_Introduction_to_Modern_Cryptography.pdf

[7] J. Hunker and C.W. Probst, "Insiders and Insider Threats—an Overview of Definitions and Mitigation Techniques," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, Vol. 2, No. 1, March 2011, pp. 4–27.

[8] M. Dworkin, “Recommendation for Block Cipher Modes of Operation Methods and Techniques,” National Institute of Standards and Technology Special Publcation. 800-38A 2001 Edition, December 2001.

[9] Types of Cryptographic Attacks. In Domain 5: Cryptography. Retrieved February 27, 2011, fromhttp://www.giac.org/resources/whitepaper/cryptography/57.php

[10] S. Klaus, *Cryptography and Public Key Infrastructure on the Internet*, John Wiley and Sons, 1st edition, June 2, 2003.

[11] A.J. Menezes, P.C. van Oorschot and S.A. Vanstone, "Chapter 7: Block Ciphers". *Handbook of Applied Cryptography*. CRC Press. ISBN 0-8493-8523-7. October 1997.

[12] Y.F. Huang, F.Y. Leu, C.H. Chiu and I.L. Lin, "Improving Security Levels of IEEE802.16e Authentication by Invoking Diffie-Hellman PKDS," *Journal of Universal Computer Science*, Vol. 17, No.6, March 2011, pp. 891-911.

[13] A.P. Moore, D.M. Cappelli, T.C. Carony, E. Shaw, D. Spooner and R.F. Trzeciak, “A Preliminary Model of Insider Theft of Intellectual Property,” *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, Vol. 2, No. 1, March 2011, pp. 28–49.

[14] E. Barkan and E. Biham, "In How Many Ways Can You Write Rijndael?" in Proceedings of ASIACRYPT 2002, (Lecture Notes in Computer Science), edited by Y. Zheng, vol. 2501. Berlin, Germany: Springer-Verlag, Dec. 2002, pp. 160-175.

[15] J. Daemen and V. Rijmen, “AES Proposal: Rijndael,” *The First Advanced Encryption Standard Candidate Conference*, NIST, September 1999.

[16] Federal Information Processing Standards Publication 197, “Announcing the Advanced Encryption Standard (AES)” November 26, 2001.

[17] Y.L. Huang, F.Y. Leu, J.C. Liu, J.H. Yang, C.W. Yu, C.C. Chu, C.T. Yang, "Building a block cipher mode of operation with feedback keys," Industrial Electronics (ISIE), 2013 IEEE International Symposium on, May 2013, pp. 1-4.

[18] Y.L. Huang, F.Y. Leu, J.H. Yang, "Building a block cipher mode of operation with two keys," The Asian Conference on Availability, Reliability and Security (AsiaARES 2013), Mar. 2013, pp. 392-398.

[19]T. Eisenbarth, S. Kumar, L. Uhsadel, C. Paar and A. Poschmann, "A Survey of Lightweight-Cryptography Implementations," *IEEE Design & Test of Computers*, Dec. 2007, pp. 522-533.

[20] H.M. Heys, "A tutorial on linear and differential cryptanalysis," *Technical Report CORR 2001-17*, Centre for Applied Cryptographic Research, Department of Combinatorics and Optimization, University of Waterloo, March 2001.

[21] Y.S. Yeh, C.Y. Lee, T.Y. Huang and C.H. Lin "A Transpositional Advanced Encryption Standard (AES) Resists 3-Round Square Attack," *International Journal of Innovative Computing, Information and Control*, vol. 5, no. 5, May 2009, pp. 1253-1264.

[22] L. Cui and Y. Cao "A New S-Box Structure Named Affine-Power-Affine," *International Journal of Innovative Computing, Information and Control*, Vol. 3, No. 3, June 2007, pp. 751-759.

[23] H.Z. Yao, "The Comparison of Efficiency and Security between AES and DES," *Journal of Zhongkai University of Agriculture and Technology*, February 2006, pp. 44–48.

Appendix: System Implementation

In this appendix, we show the implementation of the SeFEM, which is written in JAVA language on a JAVA working environment. The operation procedure is as follows.

Step 1:

Start up the system by clicking the SeFEM icon shown on the screen (see the right portion of Figure 6). When the program is successfully started up, as illustrated in the left portion of Figure 6, the "Message board" window shows "Initialized successfully" to indicate that the SeFEM has been successfully initialized.

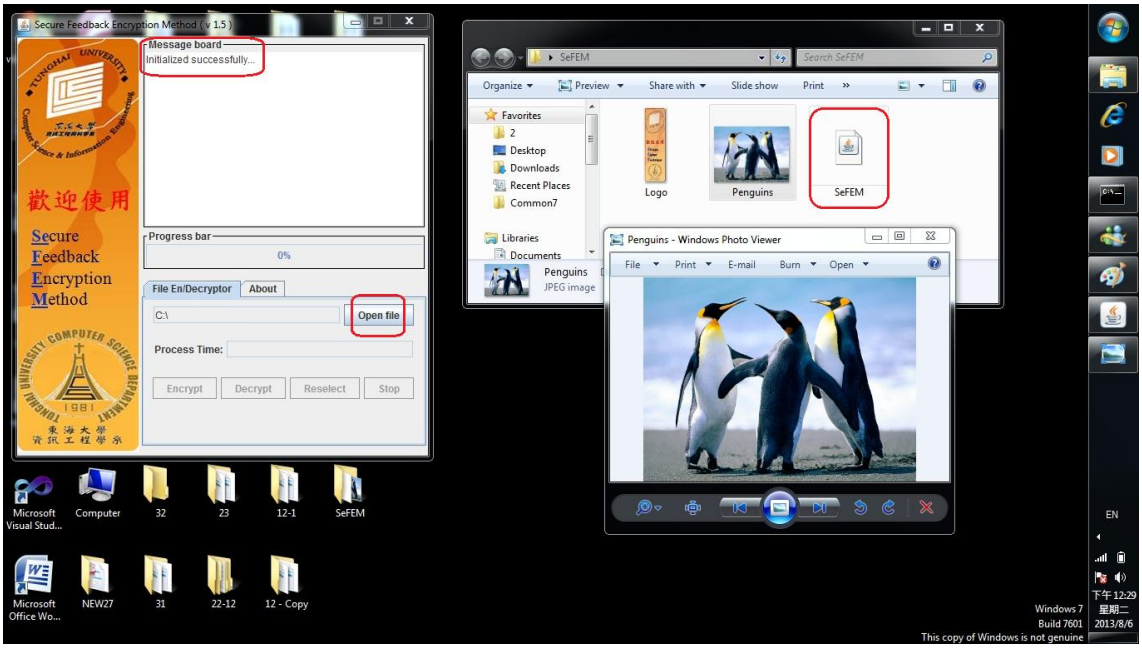

Figure 6. Program initialization

Step 2:

Click "Open file" button, as shown in the right portion of Figure 7, user can then select the file to be en/decrypted, e.g., F. Now the "Message board" window displays the file path (see Figure 8a). If the filename extension of F is not "SeFEM", meaning that F is an un-encrypted file, then the system activates the two buttons, denoted by

“Encrypt” and “Reselect”. Otherwise the system considers that the user would like to decrypt F, it then activates the two buttons, named “Decrypt” and “Reselect” (see Figure 8b). Here “Reselect” button is activated so that when the user does not want to en/decrypt the selected file, he/she can press this button to choose another one.

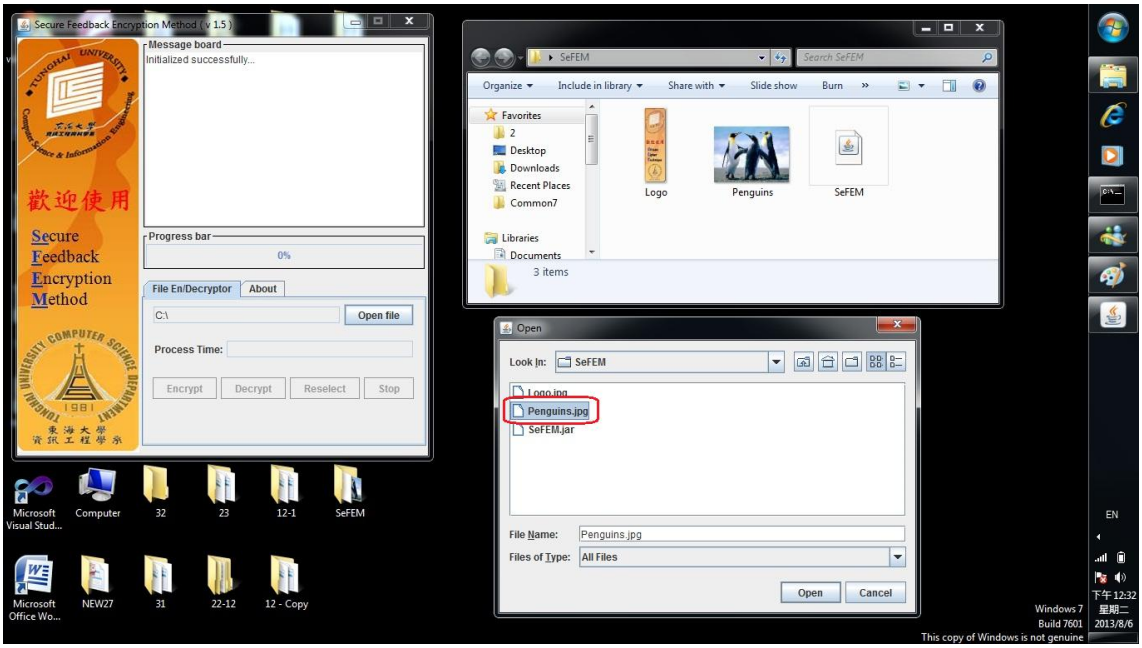

Figure 7. Document category and file selection

(a)The selected file is to be encrypted (b)The selected file is to be decrypted

Figure 8. Select the file to be en/decrypted

Step 3:

Press the “Encrypt” (or “Decrypt”) button is to run the program. The “Progress bar” window shown in the middle of the left portion of Figures 9 and 10 indicates the percentage which has been finished by the program. After the completion of the encryption (decryption), “Message board” window shows a message to indicate the completion of the operation, and displays the consumed time. In the “Message board” window shown in Figure 10, we select penguins.SeFEM and press the “Reselect” button. But the reselected file is still penguins.SeFEM. After the selected file is completely encrypted/decrypted, if we click “Open file” button again, we can select the next file for en/decryption.

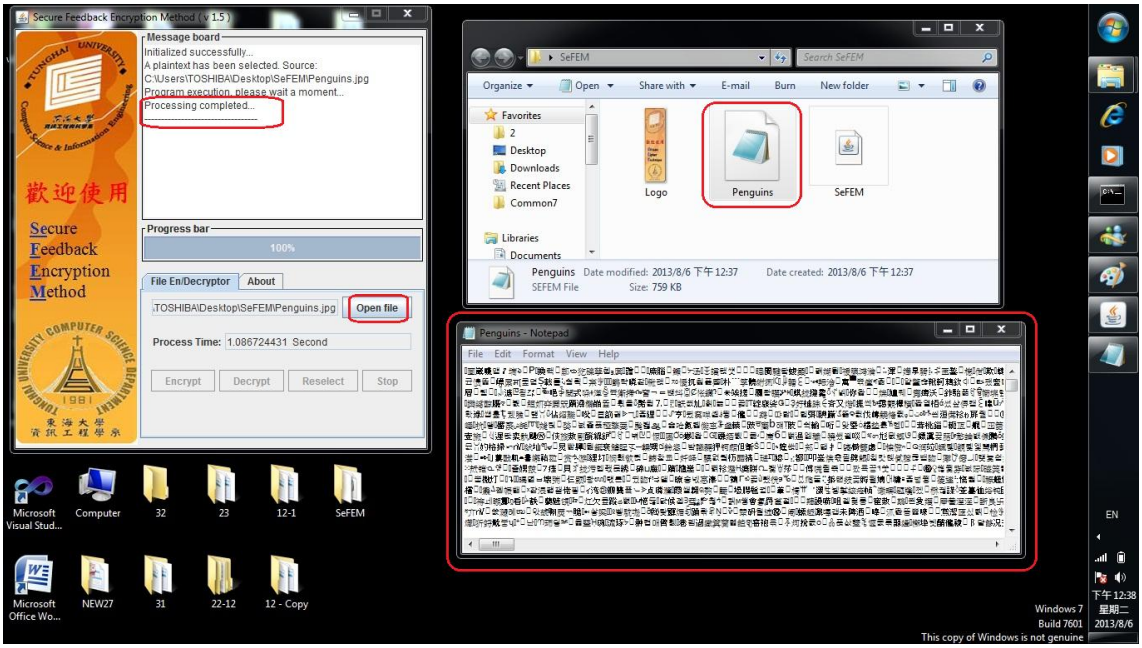

Figure 9. The completion of the encryption process

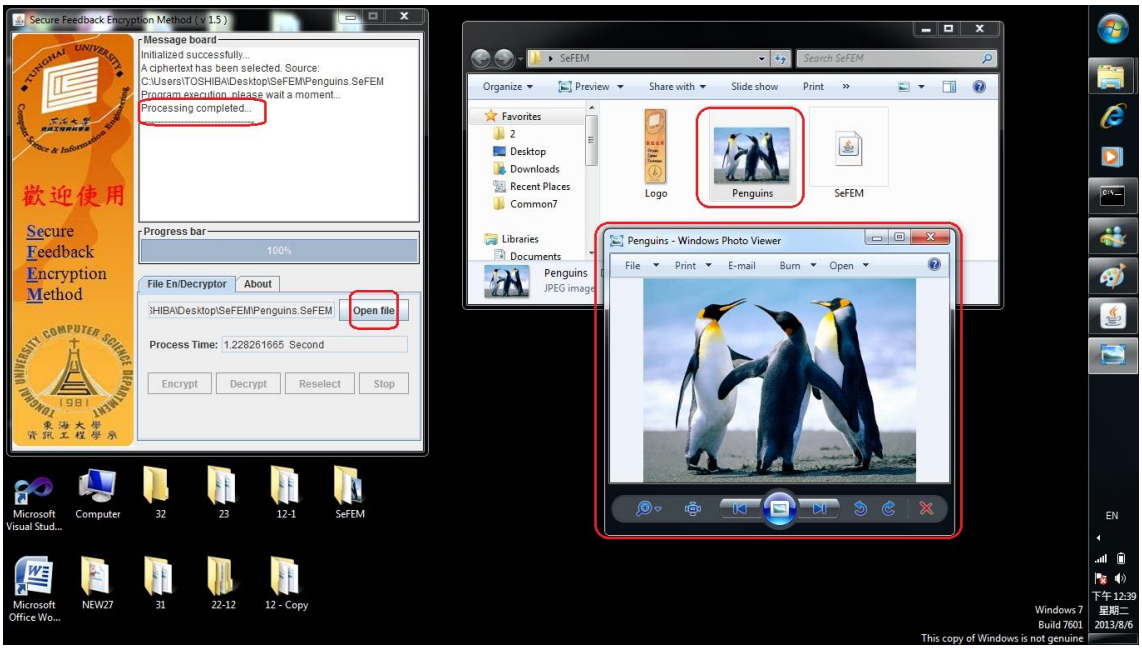

Figure 10. The completion of the decryption process