東海大學資訊工程研究所

碩士論文

指導教授: 黃宜豐

使用輸入字串產生 16x16 動態盒

Generating a 16x16 Dynamic Box by Employing Input String

研究生：潘星佑

中華民國 一零七 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 潘 星 佑 所提之論文

使用輸入字串產生 16x16 動態盒

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 陳金鈴 簽章

委 員 江輔功

時文中

指 導 教 授 黃宜華 簽章

中華民國 107 年 6 月 30 日

摘要

鑑於透過使用查表方式進行加密和解密資料非常快速且富變化，但是目前使用的表是靜態 S-Box，其內容是固定且為大眾周知，秘密性大為降低，然而若能夠根據使用者輸入的密碼字串來重建表的內容，則此表將因不同的密碼而有不同的內容，呈現動態改變，攻擊者亦因不知道密碼而不知表的內容，安全度將大大提升。在本研究中，我們開發出一種演算法，即，使用輸入字串產生 16x16 動態盒(Generate a 16 x 16 Dynamic Box by using an Input String，簡稱 GDBIS)，字串長度介於 8 位元到 800000 位元之間，可有效抵禦暴力法攻擊。GDBIS 有三個程序，(1)產生初始循序密鑰 ISK_1 和 ISK_2；(2)產生動態密鑰，DK_1，DK_2 和 DK_3；（3）透過使用一種稱為 Dynamic Box Generator（DBG for short）演算法產生動態盒(D-Box)。本研究同時開發了一種新的密鑰擴展演算法，稱為 Generation of D-Box and Round Keys（GDBRK for short）。GDBRK 由輸入的 Cipher key 透過 GDBIS 演算法產生一個 16x16 D-Box，再使用此 D-Box 產生 Round Keys。此外，AES 中使用的 S-Box 將被此 D-Box 取代如此將提高安全度而能保有效能。根據我們的安全分析，經由 GDBIS 產生的 D-Box 是不可逆轉的，具有高度的混亂與極高的輸入敏感度。而由 GDBRK 產生的 Round Keys 其相互間的關聯度極低。效能分析顯示，GDBIS 用於產生 D-Box 的時間介於 15 到 19μs 之間，由於 Cipher key 長度的不同 GDBRK 耗時介於 20 和 25μs 之間，表明這兩種機制具有出色的安全度與效能，能夠滿足實際應用的需要。

關鍵字: S-Box, GDBIS, D-Box, AES, GDBRK, E-DASS

Abstract

Today, encrypting and decrypting data by employing a lookup table is ever changeful and very quickly. But currently, the table to be looked up is a static S-Box, the contents of which is fixed, well-known and greatly reduced the confidentiality. However, if the contents of the table can be reconstructed according to the password input by the user, the table will have various contents due to different passwords. The attacker also does not know the contents of the table because he does not know the password, and thus, the security will be greatly improved. Based on this, in this study, we propose an algorithm, namely Generate a 16 x 16 Dynamic Box by using an Input String (GDBIS for short). The length of the password string is between 8 bits and 800,000 bits, which can effectively defend against all kinds of brute-force attack. The GDBIS has three procedures, including the one that generates Initial Serial Keys, ISK_1 and ISK_2; the one that produces Dynamic Keys, DK_1, DK_2 and DK_3; the procedure that yields a D-Box by using an algorithm, named Dynamic Box Generator (DBG for short). We also develop a new key expansion algorithm, called Generation of D-Box and Round Keys (GDBRK for short). Given a Cipher key, the GDBRK generates a D-Box, with which to produce Round Keys sequentially. The S-Box used in AES is also substituted by the D-Box for further increasing its security level without the loss of its encryption and decryption performance. According to our security analysis, the D-Box generated by GDBIS is irreversible with high degree of chaos and has excellent sensitivity on inputs. The Round Keys generated by the GDBRK have low relation among themselves. Our performance analysis also shows that the time that the GDBIS spends to generate a D-Box are between 15 and 19 μs. Due to the difference in password length, GDBRK takes between 20 and 25μs showing that the two mechanisms have excellent performance and are able to meet the needs of practical applications.

Keywords: S-Box, GDBIS, D-Box, AES, GDBRK, E-DASS

目錄

圖目錄

表目錄

第一章　　研究動機與研究目的

1.1 研究背景與動機

Advanced encryption standard(AES)由美國國家標準與技術研究院（NIST）發布於 2001 年 11 月 26 日 [1]，並在 2002 年 5 月 26 日成為有效的標準，AES 廣泛用於當今軟體和硬體應用傳輸的數據之保護，我們網上購物交易敏感訊息是一個典型的例子。

由於電腦計算能力與傳輸速率的提升，導致 AES 的安全性和效率問題逐漸明朗，在 2010 年，由 A. Biryukov et al.所提出的研究中[2]，對 AES 256 位元的密鑰版本已經被破解到 10 層迴圈，除了上述的相關研究之外，對於破解 AES 的研究可說是不勝枚舉[3,4,5,6]，如何改良加強 AES 的安全度為現今研究的重要課題。

1.2 研究目的

有許多研究討論如何提高 AES 的安全性，例如 block cipher mode of operation [7]或提高 S-Box 的複雜性[8,9,10]。在 AES 四個加密步驟中，SubBytes 這個動作，即，使用查表來進行資料串流的加/解密可說是相當快速且富有變化，然而， AES 中的 SubBytes 是依據固定的 S-Box 進行查表，使得安全度大為降低，倘若能由 Cipher Key 來產生相對應的動態盒(Dynamic Box, D-Box for short)以取代原本固定的 S-Box，則在不知道 Cipher Key 就不知道 D-Box 內容的情況下，將可大大提升 AES 破密的困難度。因此，在本研究中，我們提出一種能夠依據輸入之金鑰字串產生相對應 D-Box 的方法，即，“由一個字串產生一個 16x16 動態盒(Generate a 16 x 16 Dynamic Box from a String)”演算法，簡稱 GDBIS。 GDBIS 由 3 個程序構成，即，(1)產生初始序列金鑰(Initial Serial Keys), ISK_1 與 ISK_2、(2) 產生動態金鑰(Dynamic Keys), DK_1, DK_2 與 DK_3、(3)藉由動態盒產生器(DynamicBox Genetor)產出 D-Box。GDBIS 具有下列優異特性，即，(1) 擁有極高的輸入敏感度 (2) 產出的表格具有擬動態隨機變化之

重排特性 (3) 由不同字串產生完全相同表格之碰撞機率極低 (4) 為一個極佳之單向函數演算法。基於以上優異特性，由 GDBIS 產出之 D-Box 將擁有極高之安全性。另外，在本研究中，我們開發一套全新的金鑰擴充方法來取代 AES 原本金鑰擴充之方法，其不但可以產出必要的回合金鑰，更重要的是，能同時由輸入之 Cipher Key 來產出其對應之動態盒(D-Box)，用來取代 S-Box，做為 AES 資料串流加/解密之用，如此能在幾乎不損失效能的情況下，大大提升 AES 的安全性。

1.3 論文架構

本文的其餘章節安排如下：第 2 章介紹本文的相關研究；第 3 章中，描述了 GDBIS； 第 4 章為安全性分析與比較；第 5 章效能分析；最後，第 6 章為結論與展望。

第二章　　文獻回顧與探討

2.1 AES 之相關文獻

AES 是目前使用最廣泛的區塊密碼技術，通常使用 128 位元的資料區塊對資料進行加密和解密。 這個演算法使用 10 回合，12 回合，14 回合的加密計算。賦予該算法的密鑰被擴展成每個包含 4 bytes 的 44 個字組。 AES 算法中使用的密鑰大小可以是 128,192 與 256 位元，其分別搭配 10,12,與 14 回合的運算，以增加安全度。

AES 各回合均包含 4 個加密階段，在第一階段的 SubBytes 運算(如圖 2.1)，透過一個非線性的置換盒(S-Box)，用查表的方式把狀態矩陣中每個位元組替換成對應的位元組；

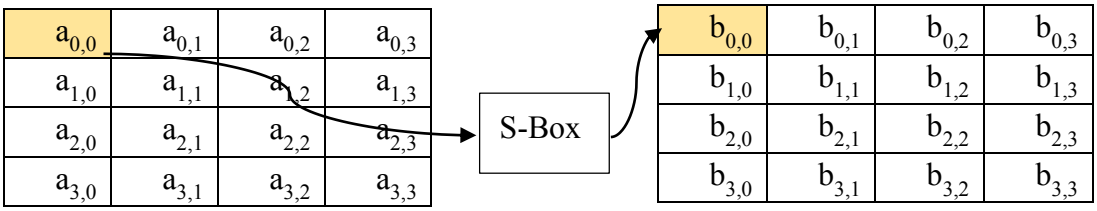

圖 2.1:AES-SubBytes

第二階段的 ShiftRows 運算(如圖 2.2)，是將狀態矩陣中的第 i 列做左旋 i 位元組，$0 \le i \le 3$；

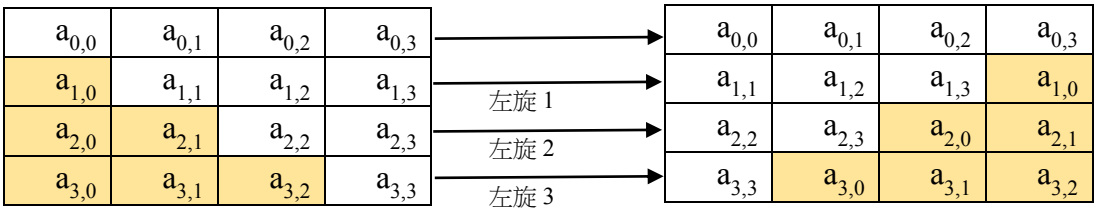

圖 2.2:AES-ShiftRows

第三階段的 MixColumns 運算(如圖 2.3)，是將狀態矩陣中的每個直行在 modulo $x^8+x^4+x^3+x+1$ 之下，和一個固定多項式 c(x)作乘法；

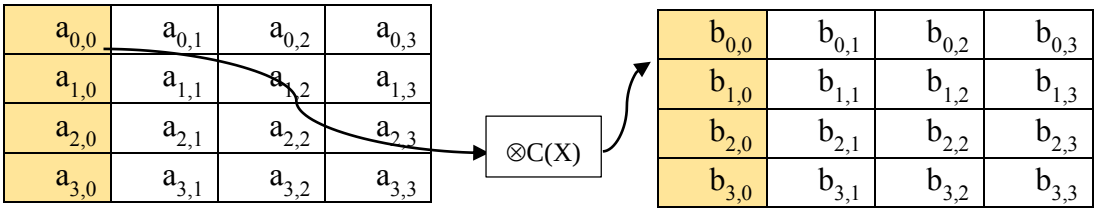

圖 2.3:AES-MixColumns

在最後階段的 AddRoundKey 運算(如圖 2.4)，其將狀態矩陣中的每一個位元組都與該次回合金鑰（round key）做 XOR 運算，而回合金鑰由金鑰擴展程序產生。

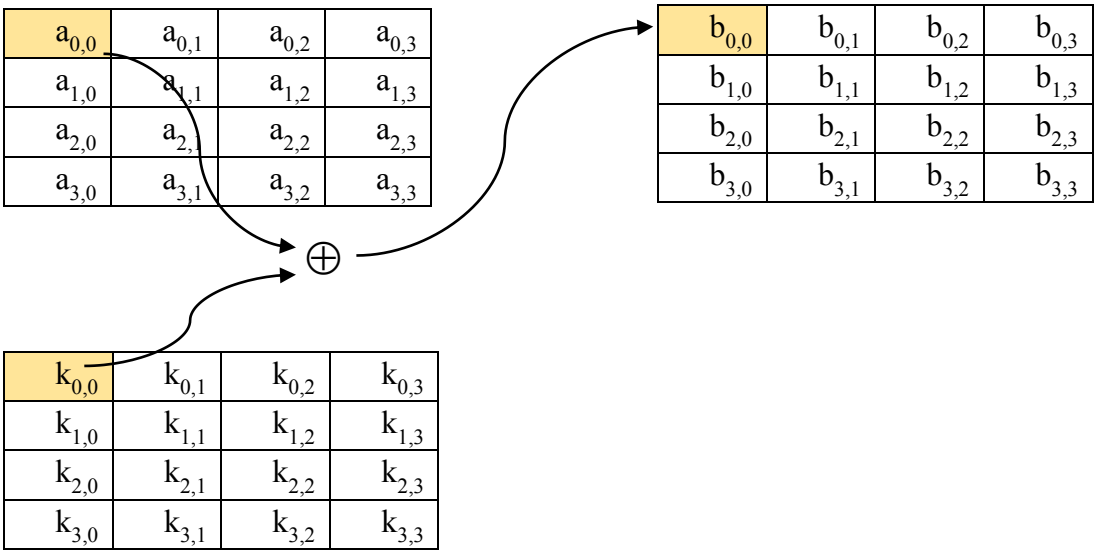

圖 2.4:AES-AddRoundKey

2.2 Dynamic S-Box 之相關文獻

在 AES 中，S-Box 能夠充分展現混淆(confusion)的特性[11,12]，即增加明文與金鑰之間計算的複雜度，使金鑰之內容均勻混入加密的訊息中，讓密文與加密金鑰間的關係變得複雜而難以用統計的方式進行計算，然而在 AES 中，查表所使用的是固定的 S-Box，故駭客可藉由密文直接推敲出明文，在安全上可說是有所不足。

在 2012 年，Razi Hosseinkhani 與 H. Haj Seyyed Javadi 曾發表動態 S-Box 之相關研究[13]，其方法是將 cipher key 作為輸入，利用 cipher key 產生相關數值，

進而對原本的 S-Box 進行 4 次重排以產生相對應的 S-Box，除此之外，E. M. Mahmoud et al.在 2013 年亦提出一種產生 Dynamic Key-Dependent S-box 的方法來改良 AES[14]，其做法便是藉由 secret key 來產生 initial state of a pseudo random (PN) sequence generator，而 PN generator 的輸出便用來作為重排原本 S-Box 之參數，除了上述兩項研究之外，還有許多關於改進 S-Box 或是產生動態 S-Box 的相關研究[15,16,17]。然而，這些方法均是以重排固定的 S-Box 來產生相對應的動態 S-Box，但單以這些重排的方法而產生動態 S-Box，其本質上有一嚴重的安全缺失，即，它們都是由已知的 S-Box 重排產生，故在敏感度與亂度上都稍嫌不足，在本研究中，我們提出 GDBIS 演算法，即，藉由任一輸入字串來產生 D-Box，在我們的方法中，完全排除使用 S-Box，故沒有先天 S-Box 內容的影響與限制，以 GDBIS 產生之 D-Box 取代 AES 的 S-Box，更能提高整體安全度。

2.3 3D 運算之相關文獻

在本研究中，GDBIS 演算法是藉由整合了多種不同的演算法來進行計算，藉此降低各項參數之關聯性並提高其輸入敏感度與複雜度，其中核心部分之一便是使用三維運算。在 2015 年，Huang et al.提出一個循序加密的方法，即，Secure Feedback Encryption Method (SeFEM) [18]，此方法具備三種特色，包括使用循序邏輯進行加解密、使用三個基本運算子進行運算及使用動態轉換表，明文塊經由動態轉換表置換後再與多重金鑰(multiple keys)進行 3D 運算，藉此產生密文塊，在 SeFEM 中使用的基本運算子為⊕、$+_2$、⊙，其中⊙與⊕具有互補關係，有高度的關聯性，因此降低其複雜度，為 SeFEM 的一個缺點，因此 Huang et al.在 2017 年提出的改良研究中，即，A 3D Encryption with Shifting Mapping Substitution Mechanism[19]，由$\odot_R/\odot_{IR}$取代⊙/⊙來增加此第三運算子的強度，因此在本研究中，我們便使用⊕/⊕、$+_2/-_2$與$\odot_R/\odot_{IR}$這三個基本運算子所建構的 3D 運算來產生動態金鑰，藉此增加 D-Box 之亂度而仍能保有高度的效能。

第三章　　動態盒生成原理

3.1 名詞與解釋

D-Box　　Dynamic Box，由任意輸入字串所產生的 16x16 動態盒，其 256 個元素值是由 00~FF 隨機排列而成。

ISK$_s$　　Initial Serial Keys，包括 *ISK*$_1$ 和 *ISK*$_2$，它們由輸入字串透過特定演算法產生，長度為 1024 bits。

DK$_s$　　Dynamic Keys，由 *ISK*$_1$ 與 *ISK*$_2$ 透過 E-DASS 與三維運算所產生之三組動態密鑰，其長度為 1024 bits。

FA　　Flag Array，為一個旗幟狀態陣列，其索引值被用來作為 D-Box 的元素值，用來確保 D-Box 內元素的唯一性，當 Flag Array 的元素值為" F "時，表示該元素之索引值尚未被選用，而" T "則表示該元素之索引值已被選用。

*IA*1　　1st Insert Array, 透過旗幟狀態陣列 *FA* 元素值，即，" T " or " F "，的辨識，將 *DK*$_1$ 陣列中重複出現之字元除去後的陣列，其字元之 ASCII 值將被用來作為 D-Box 的元素值。

*IA*2　　2nd Insert Array, 透過旗幟狀態陣列 *FA* 元素值，即，" T " or " F "，的辨識，從 *DK*$_2$ 與 *DK*$_3$ 中萃取出沒有出現於 *IA*$_1$ 的所有字元所形成的陣列，其字元之 ASCII 值將被用來作為 D-Box 的元素值。

IAR　　Residual Insert Array,在經過 *DK*$_{1\sim3}$ 選用後，*FA* 未被取用之所有剩餘元素值為" F "的索引值所形成的陣列。

*IA*3　　3rd Insert Array, 將 *IAR* 進行重排後所形成的陣列，其元素值將被用來作為 D-Box 的元素值。

RLi　　*IAi* 的實際長度，$1 \le i \le 3$。

RLR　　*IAR* 的實際長度。

3.2 基本運算子

令明文為 p，密文為 c，加密金鑰為 k，則⊕、$+_2$、$\odot_R$ 三個基本運算子[19]加/解密之定義如下:

⊕ Exclusive-OR operator :

加密：$c=p\oplus k$;

解密：$p=c\oplus k$;

$+_2$ Binary-Addition operator :

加密：$c_i = p_i +_2 k$, 為 p、k 兩運算元進行二進制加法，並在過程中捨去其最高位之進位。;

解密：$p_i = c_i +_2 k = \begin{cases} c_i - k, & if \ \ c_i \geq k \\ c_i + \bar{k} + 1, & if \ \ c_i < k \end{cases}$ /* 為$+_2$之反運算 */

$\odot_R$ Rotate Equivalence operator :

加密 ： $c = p\odot_R k = p_R \odot \ k$ ，其中 p_R 是由明文 p 順時針旋轉 h 個位元所得到，其中 $h = |k| / 4$;

解密 ： $p = c\odot_{IR} k$ = 逆時針旋轉$(c\odot k)$ $|k| / 4$ 個位元所得；/*$\odot_{IR}$ 為$\odot_R$ 之反運算*/

3.3 Enhanced DASS 演算法

在[20]中定義的 DASS 演算法有一個缺點，因為移位計數器僅具線性變化且每次查找 S-Box 時只有 0 到 8 之間的範圍，因此它不能很好的防禦暴力攻擊。以下是其增強版本，Enhanced DASS（簡稱 E-DASS），其作為單向功能採用動態積累的移位替代機制將明文加密成不可逆的密文。

Algorithm 3.1: E-DASS algorithm /* format is E-DASS(P,R-Box) */

Input: a plaintext P of n bits in length, and a 16 x16 random-box (R-Box for short) where n is a multiple of 8.

Output: ciphertext C.

1. Let $P = p_1 p_2 \ldots p_k$ and $C = c_1 c_2 \ldots c_k$, where $k = n/8$;

2. *dsc* = 256; /**dsc*: dynamic shifting count*/

3. For *i*= 1 to *k* {

4. $vp[i]$ = Int(p_i); /* Int(p_i) is the ASCII code of character p_i*/

5. *dsc* = *dsc* + (*vp*[*i*] +1)**i*;}

6. *vp*[0] = *vp*[1] + *vp*[*k*];

7. For *i*= 1 to *k* {

8. *dsc* = (*dsc* + (*vp*[*i*-1])**i*) mod 65536;

9. *ch* = str((*vp*[*i*]+ *dsc*) mod 256);

10. c_i = the corresponding content in the R-Box after *ch* is substituted by lookingup R-Box;}

3.4 Generating a 16 x 16 Dynamic Box from an Input String

GDBIS 演算法由三個程序構成，即，(1) 產生初始循序密鑰、(2) 產生動態密鑰、(3) 藉由 Dynamic Box Generator (DBG)產生 D-Box，從輸入字符串產生一個 16×16 的動態盒，其中 *P* 為輸入之明文字串，其長度介於 8 至 800000 bits，i.e.，8≤|*P*|≤800000。

3.4.1 Generation of Initial Serial Key

當輸入字串的長度小於 1024 bits，則 GDBIS 使用 E-DASS_expansion（）和 Binaryadder_expansion（）擴展它，使得字串的長度為 1024 bits。這兩種密鑰擴展演算法如下所述。

Algorithm 3.2: E-DASS_expansion (*P*)

Input: a string *P* with length less than 1024 bits

Output: an extended string *P* with length 1024 bits exactly

1. Do {

2. *P*'= E-DASS (*P*, S-Box);

3. *P* = *P*||*P*';

4. while (|*P*|<1024);}

5. *P* = E-DASS (*P*, S-Box);

6. return Right(*P*, 1024);

Algorithm 3.3: Binaryadder_expansion (*P*)

Input: a string *P* with length less than 1024 bits

Output: an extended string *P* with length 1024 bits exactly

1. Do {

2. $P = (P||P) +_2 (P||P)$;

3. While (|*P*|<1024);}

4. return Right(*P*,1024);

The algorithm for of ISK_1 and ISK_2 is as follows.

Algorithm 3.4:

The Generation of Initial Serial Keys algorithm (圖3.1)/* format is GISK(*PW*, S-Box) */

Input : *PW* and a S-Box

Output : ISK_1, ISK_2

1. If |*PW*| < 1024 then{

ISK_1 = E-DASS_expansion(*PW*);

ISK_2 = Binaryadder_expansion(*PW*);}

2. If |*PW*| = 1024 then{

ISK_1 = *PW*;

ISK_2 = E-DASS(*PW*, S-Box);}

3. If |*PW*| > 1024 then{

(1) $PW = p_1p_2p_3 \ldots p_m$,where $|p_i| = 1024$,

$1 \leq i < m, m \geq 2$;

(2) If $|p_m| < 1024$ then p_m = Binaryadder

_expansion（p_m）;

(3) $ISK_1 = p_1p_2 \oplus \ldots \oplus p_m$;

(4) ISK_2 = E-DASS(ISK_1, S-Box);}

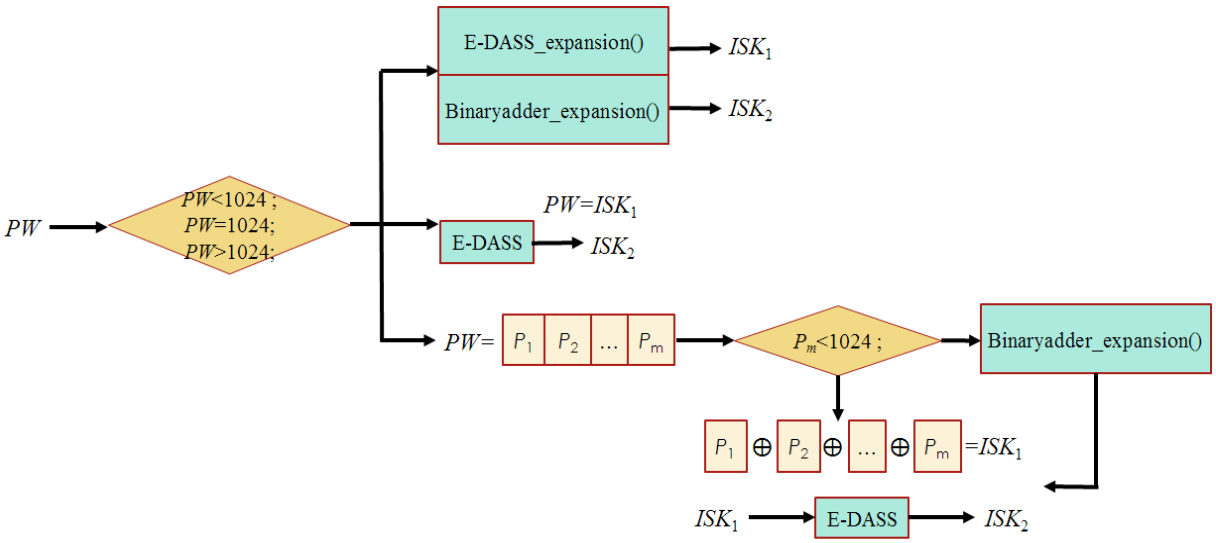

圖 3.1:GISK 架構圖

3.4.2 Generation of Dynamic Keys

In the following, we introduce the process of Dynamic keys generation as following.

Algorithm 3.5:

The Generation of Dynamic Keys algorithm (圖 3.2) /* format is GDK(ISK_1, ISK_2, S-Box) */

Input: ISK_1, ISK_2 and S-Box

Output: DK_1, DK_2 and DK_3

1. DK_1 = E-DASS($ISK_1 \oplus ISK_2$, S-box);
2. DK_2 = E-DASS($DK_1 +_2 ISK_2$, S-box);

3. DK_3 = E-DASS($DK_1 \odot_R DK_2$, S-box);

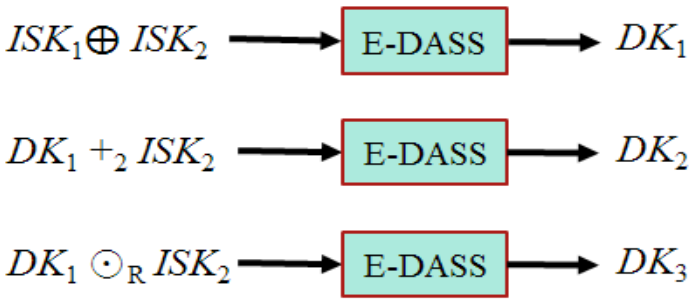

圖 3.2:GDK 架構圖

3.4.3 Dynamic Box Generator (DBG for short) algorithm

The algorithm of Dynamic Box Generator is as follows.

Algorithm 3.6:

The Dynamic Box Generator algorithm (DBG, for short) (圖 3.3) /*format:

DBG(DK_1, DK_2, DK_3) */

Input: DK_1, DK_2 and DK_3

Output: D-Box

{ Let $DK_1 = KA_0||KA_1||\ldots||KA_{127}$, where KA_j is 8 bits in length for all j, $0 \leqq j \leqq 127$;

Let $DK_2 = KB_0||KB_1||\ldots||KB_{127}$, where KB_j is 8 bits in length for all j, $0 \leqq j \leqq 127$;

Let $DK_3 = KC_0||KC_1||\ldots||KC_{127}$, where KC_j is 8 bits in length for all j, $0 \leqq j \leqq 127$;

For j = 0 to 127{

$KA[j]$ = Int(KA_j); $KB[j]$ = Int(KB_j); $KC[j]$ = Int(KC_j); $IA1[j]$=0; }

For j = 0 to 255{

FA[j]=”F”; D-Box[j]=0; $IA2[j]$=0; $IAR[j]$=0; $IA3[j]$=0; }

$RL1$=0; $RL2$=0; RLR=0;

For i = 0 to 127{

If ($FA[KA[i]]$ = "F"){

$IA1[RL1] = KA[i]$; $FA[KA[i]]$ = "T"; $RL1 = RL1+1$;}}

```
For i = 0 to 127{
    If (FA[KB[i]] = "F"){
        IA2[RL2] = KB[i]; FA[KB[i]] ="T"; RL2 = RL2+1;}
    If (FA[KC[i]] = "F"){
        IA2[RL2] = KC[i]; FA[KC[i]] = "T"; RL2=RL2+1;}}
For i = 0 to 255{
    If (FA[i] = "F"){
        IAR[RLR] = i; RLR = RLR+1;}}
HL = RLR/2;
For i =0 to HL-1{
    IA3[i] = IAR[HL+i]; IA3[HL+i] = IAR[i];}
If (RLR is odd){
    IA3[RLR-1] = IAR[RLR-1];}
t1=0; t2=0; t3=0; j=0;
while (j≦255) {
    If (t1 < RL1){
        D-Box[j] = IA1[t1];j = j+1; t1 = t1+1;}
    If (t2 < RL2){
        D-Box[j] = IA2[t2]; j = j+1; t2 = t2+1;}
    If (t3 < RLR){
        D-Box[j] = IA3[t3]; j = j+1; t3 = t3+1;}}}
```

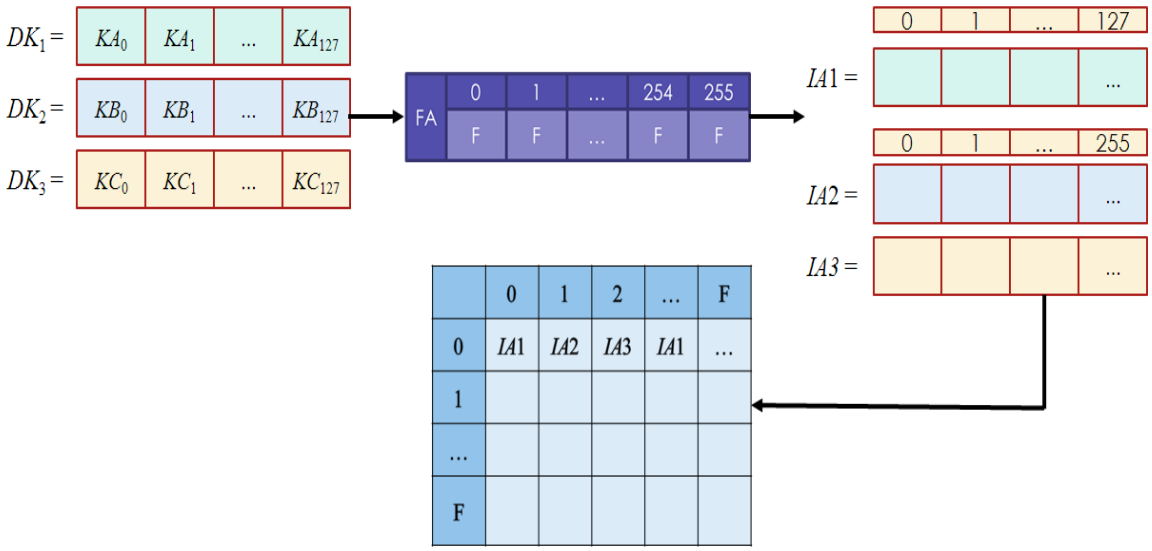
DK1 =
KA0
KA1
...
KA127
DK2 =
KB0
KB1
...
KB127
DK3 =
KC0
KC1
...
KC127
FA
0
1
...
254
255
F
F
...
F
F
IA1 =
0
1
...
127
...
IA2 =
0
1
...
255
...
IA3 =
...
0
1
2
...
F
0
IA1
IA2
IA3
IA1
...
1
...
F

圖 3.3:DBG 架構圖

3.5 Improved AES by Employing D-Box

為了提高 AES 的安全性而不失其效能，本研究中產生了一全新的密鑰擴展演算法，即，Generating Dynamic Box and Round Keys，簡稱 GDBRK，用來取代 AES 原有的密鑰擴展演算法，並使用 GDBRK 產生的 D-Box 取代 S-Box 來改良 AES，稱為 AESEDB (Improved AES by Employing D-Box)，茲分述如下：

3.5.1 Generating Dynamic Box and Round Keys

GDBRK 由輸入的密碼金鑰(*PW*)，其長度介於 8～800000 bits，透過 GDBIS 演算法產生動態盒(D-BOX)，再經由此動態盒結合 E-DASS 與 3D 運算產生所需的回合金鑰群(Round Keys)，由於產生的過程是不可逆的，故 Round Keys 之間的關聯性大為降低，相較於 AES Round Keys 的產生，是使用可逆操作的"⊕"運算子，有相當高的關聯性，GDBRK 產生的 *RKs* 有更高的安全度，Algorithm 3.7 即為 GDBRK 演算法:

Algorithm 3.7: GDBRK algorithm

Input: Password (*PW*, for short)

Output: D-Box and Round Keys (RK_0~RK_5)

1. Call GISK(*PW*, S-Box) to obtain ISK_1 and ISK_2;
2. Call GDK(ISK_1, ISK_2, S-Box) to obtain DK_1, DK_2 and DK_3;
3. Call DBG(DK_1, DK_2, DK_3) to obtain D-Box;
4. TDK_1 = E-DASS($DK_1 \oplus DK_2$, D-Box);
5. TDK_2 = E-DASS($TDK_1 +_2 DK_3$, D-Box);
6. RK_j = Mid($TDK_1 \odot_R TDK_2$, 16×(j+1)+1, 16), $0 \leqq j \leqq 5$; /* length of RK_j is 16 bytes and TDK_1 , TDK_2 is 128 bytes */
7. return D-Box, RK_0~RK_5;

3.5.2 AESEDB 之工作原理

AESEDB 的架構如圖 3.1 所示，其中，AESEDB 使用密碼金鑰(*PW*)取代 *Cipher Key* 輸入至 GDBRK 以產生加密需要的 D-Box 與 *Round Keys* (RK_0~RK_5)，由於 *PW* 的長度範圍為 8~800000 bits，而不是固定的 128 bits，此變動長度的特性，不僅增加了使用的彈性，同時也因其長度的不固定，成為破密的重要變動因素；基於不同的 *PW*，GDBRK 將產生不同的 D-Box，攻擊者在不知道 *PW* 的內容下，當然也不知道 D-Box 的內容，使用 D-Box 來取代 S-Box 進行 SubBytes 運算，將大大增加密文的安全性，破密的困難度得以顯著提升；透過 GDBRK 演算法整合使用 3D 運算、E-DASS 運算與 D-Box 而產生的 *Round keys*，將繼承了 3D 運算及 E-DASS 的安全特性，如此前後的 *Round keys* 關聯性極低，複雜度極高，所以由本法產生的 Round Keys 比 AES 的 Round Keys 擁有更高的安全度，基於此，本法將 AES 10 回合的運算縮減為 5 回合的運算，其安全度仍將高於原 AES 的安全度（詳見 4.4），更提升了效能(詳見 5.4)。

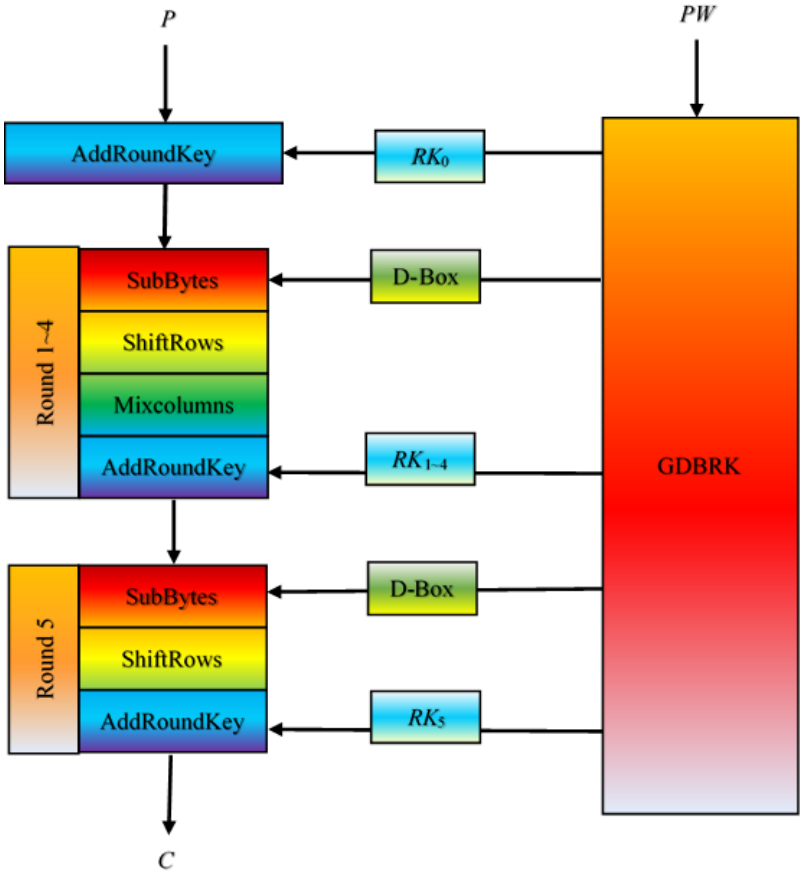

圖 3.4：AESEDB 架構圖

第四章　　安全分析與比較

在本研究中，動態盒(D-BOX)的產生主要是由三個動態金鑰，即，DK_1, DK_2 與 DK_3 之元素值，透過取用重設 *FA* 元素，藉以去除 DK_1, DK_2 和 DK_3 的重複字元之後，用於產生 D-BOX，故 DK_1, DK_2 與 DK_3 之亂碼品質至為重要。此三個動態金鑰主要是由輸入字串經由多次 E-DASS（）運算進行字串置換而得。其次，本研究重新改寫了 AES 的回合金鑰擴充演算法，此新法的安全特性及對 AES 安全度的影響亦需要進一步討論。

4.1　E-DASS（）之安全分析

E-DASS（）演算法基本上是透過查表方法將字串進行置換，首先將長度為 *n* 位元之輸入字串分割成 *k* 個字元(位元組)，每個輸入字元的相應密文是將其 ASCII 碼與當前動態移位累積值（簡稱 *dsc*，見本演算法的第 5 步）運算後再透過查表產生。故，在整個查表過程中，*dsc* 值的變化為 E-DASS（）安全的所在。

dsc 的初始設定值為 256，經由步驟 3 至步驟 5 的 For 迴圈作用，其中步驟 5 將每一輸入字元之 ASCII 值與其對應位置之乘積增量貢獻累加至 *dsc* 中，因此，相同字元出現在不同位置會有不同的 *dsc* 增量貢獻。

在步驟 7 至步驟 10 的 For 迴圈中，依序地產生密文 c_i，而每一個 c_i 是透過查找 R-Box 代替字串 *ch*，並且 *ch* 的 ASCII 碼是（$dsc + vp[i]$）mod 256。在步驟 8 中，*dsc* 的非線性累積增量是由 *vp* [*i*-1]和 *i* 的乘積所引起的，因此 *dsc* 在每一步都是動態非線性變化的。這裡所提的動態，主要是指每次輸入的明文字串是動態改變，故藉由明文字串進行運算的 *dsc* 是動態改變的。

由前述的分析，我們可以得到以下結論:

1. 經由 E-DASS（）作用產出的每一個密文字元 c_j，$1 \leqq j \leqq k$，皆是由兩個動態參數 *vp*[*i*]與 *dsc* 一起決定，故即使 R-Box 為眾所皆知，駭客仍是無法用 c_j 找出正確的 p_j，$1 \leqq j \leqq k$。

2. 由 E-DASS（）產出的密文擁有極高的輸入敏感度，因為在步驟 3 和步驟 5 之間，明文上的任何微小變化都會被放大並反映在 *dsc* 的值上。
3. 經由步驟 8 *dsc* 的循序非線性累積增量作用，進而影響整個查表對應位置，使得密文的輸出具有高度的不可預測性。

4.2 GDBIS 之安全分析

GDBIS 將輸入之字串經由多次 E-DASS( )的作用而產生動態金鑰 DK_1, DK_2 與 DK_3，從而產生 D-Box，因 GDBIS 繼承了 E-DASS（）的安全性，擁有極高的輸入敏感度與無法由動態盒反向推演求出字串的單向特性。再者，輸入字串的長度介於 8~800000 位元間，使其具有極佳的輸入彈性，在使用上更方便，更因其長度變化上限為 800000 位元，使得輸入之密碼字串可以是一句話甚至是一篇文章，如此將可有效抵禦暴力法攻擊，大大增加其產出之 D-BOX 的安全性。

4.3 AESEDB 之安全分析與比較

本節將從密碼金鑰安全性、置換盒的強度、回合金鑰安全性以及整體系統複雜度等方向對 AESEDB 的安全度進行分析。

首先，在密碼金鑰上，原先 AES 採用固定長度，i.e.，128 位元長度，的密碼金鑰作為保護，然而，固定長度意的密碼鑰較易被猜測，相較之下，因 AESEDB 的 *PW* 長度範圍為 8~800000 bits，在可變動長度的密碼金鑰保護下，其長度變化也將成為破密將面臨的重要變動因素。

在置換盒應用強度的表現中，如圖 4.1，AES 是眾所周知的固定置換盒 S-Box，幾乎沒有安全性可言，其作用僅是將密文狀態混淆，而 AESEDB 的動態置換盒(D-Box)，會因 *PW* 不同而不同，攻擊者將無從知曉其內容，使每次的 SubBytes 都是私密的加密，大大增加了 SubBytes 破密的困難度，安全度得以有效提升。

AESEDB 經由 GDBRK 產生的回合金鑰其前後回合金鑰間無固定關係，故關聯度極低，無法由前後的回合金鑰互相關聯求出，固擁有較高的安全度。

從系統複雜度來看時，AES 僅透過固定長度金鑰(Cipher Key），i.e.，128 位元長度，與關聯性極高的回合金鑰保護加密系統，而 AESEDB 採用長度變化大的密碼金鑰(*PW*），i.e.，8~800000 位元長度，與關聯性極低的回合金鑰保護加

密系統，複雜且敏感易變的 D-Box 由密碼金鑰來產生，在不知密碼金鑰的情況下就不知 D-Box 內容，使得 AESEDB 系統安全性大增。表 4.1 將 AESEDB 與 AES-128 就不同的安全特性進行比較，其中，在系統變數長度一項，由於 D-Box 共有 256!個且外界無從得知，基於 $2^{1683} < 256! < 2^{1684}$，故，我們視 D-Box 為一具有 1683 位元長度的變數。

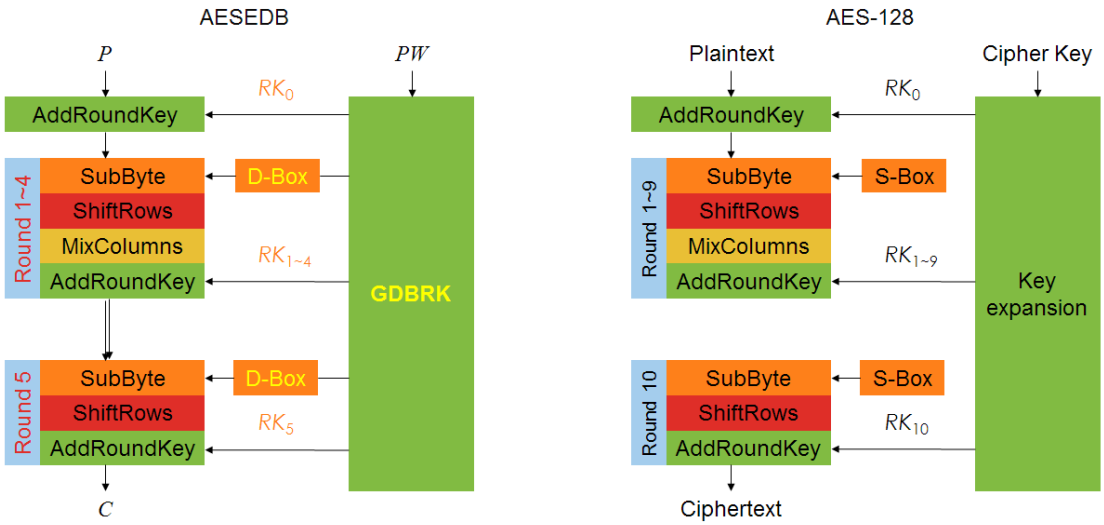

圖 4.1: AESEDB 與 AES-128 架構比較圖

表 4.1：AESEDB 與 AES-128 安全性比較

特性/版本	AESEDB	AES-128
密鑰長度	8~800000-bit/優	128-bit/劣
置換盒	D-Box/優	S-Box/劣
回合金鑰	關聯性低，安全度較高/優	關聯性高，安全度較低/劣
系統變數	2451-bits/優 (128*6+1683)	1408-bits/劣 (128*11)

表 4.1 清楚呈現，AESEDB 在密鑰長度、置換盒強度、回合金鑰強度與系統變數總長度等安全特性接由於 AES 使得 AESEDB 的安全度遠高於 AES。

第五章　　效能分析

演算法的效率非常重要，特別是在密碼學方面，因此，本章節將分析 E-DASS, GDBIS, GDBRK 與 AESEDB 的效率。 輸入字串的可變長度將用於模擬實際使用，模擬環境如表 5.1 所示。

表 5.1：模擬環境規格

CPU	AMD Phenom II X4 955 @ 3.20GHz
RAM	12GB
O.S.	Windows 10
Programming tools	C / Microsoft (R) C/C++ Optimizing Compiler Version 19.00.24215.1

5.1 基本運算子的效能

首先測試基本運算子 $\oplus$, $+_2$, $-_2$, $\odot_R$, 和 $\odot_{IR}$，每個基本運算子的耗時是執行 1250 萬次後的平均值。表 5.2 列出了每個運算子花費的平均時間。

表 5.2：本研究中使用的基本運算子 $\oplus/\oplus$, $+_2/-_2$ and $\odot_R/\odot_{IR}$ 在不同長度密鑰下的執行時間（為 1250 萬次執行時間的平均值，單位為 ns）

Key size / Operator	128-bit	256-bit	512-bit	1024-bit
$\oplus$	10.77	20.59	38.37	76.09
$+_2$	11.08	23.94	68.22	155.35
$-_2$	11.12	23.30	67.21	146.05
$\odot_R$	22.98	42.73	84.22	169.29
$\odot_{IR}$	23.83	43.01	83.67	177.04

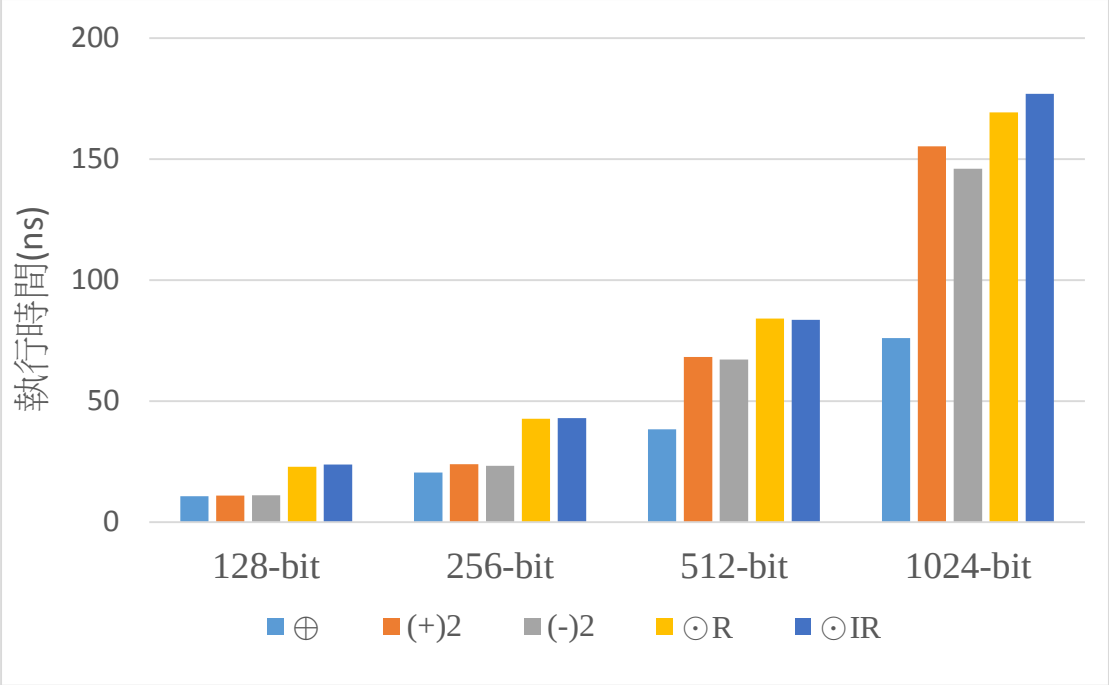

圖 5.1: 本研究中使用的基本運算子 $\oplus/\oplus$, $+_2/-_2$ and $\odot_R/\odot_{IR}$ 在不同長度密鑰下的執行時間圖

除了 XOR 運算子，其他的運算子都是由軟體實現。但是，執行時間不會和 XOR 差太多。 換句話說，如果都用硬體實現，時間可以進一步縮短，但改進不會很大。

5.2 E-DASS 和 GDBIS 的效能

基本上，E-DASS 和 GDBIS 具有極佳的輸入彈性，因兩者都能夠處理不同長度的輸入。但是，GDBIS 的過程步驟較冗長繁瑣會因輸入的不同長度而有較大的差異。 此外，當輸入密鑰字串大小較長時，E-DASS 的成本也會增加。表 5.3 列出了不同長度密鑰大小的 E-DASS 的執行時間。

表 5.3：E-DASS 在密鑰大小分別為 128,256,512 和 1024 位元下的執行時間（為 1250 萬次的執行時間的平均值，單位為 ns）

Input size			
128-bit	256-bit	512-bit	1024-bit
302.69	509.99	918.72	1744.11

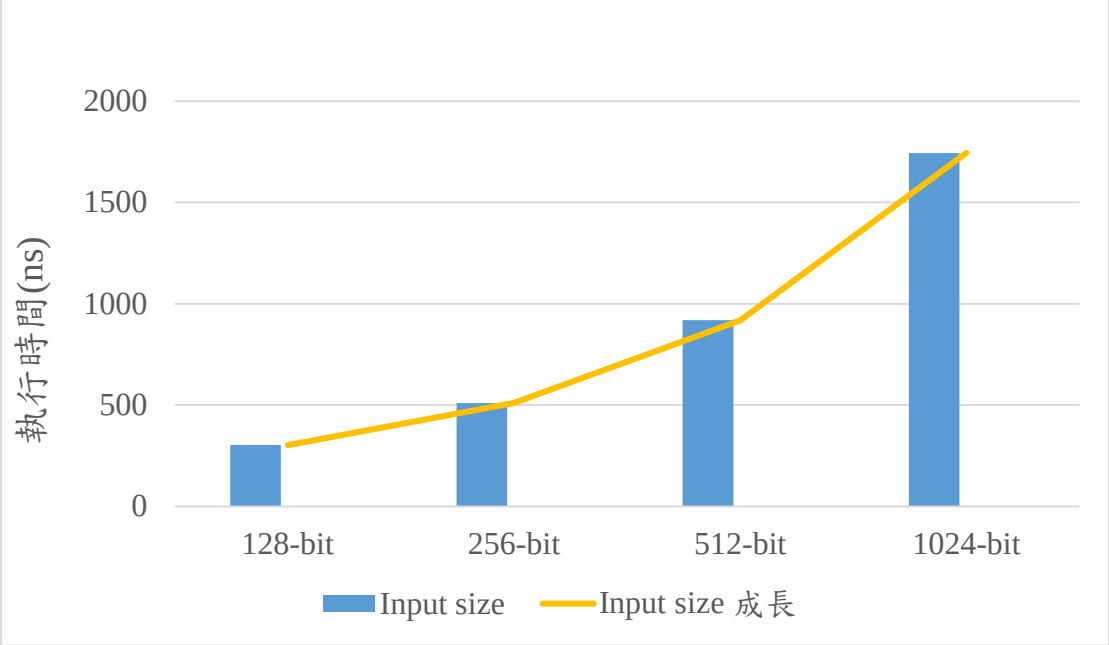

圖 5.2: E-DASS 在密鑰大小分別為 128,256,512 和 1024 位元下的執行時間圖

表 5.3 說明了 E-DASS 的執行時間，線性增長。 但增幅並不顯著。

表 5.4：GDBIS 在不同長度下的平均執行時間（為 200 萬次執行時間的平均值，單位為 μs）

Input size (l)				
128-bit	l <1024bits	l =1024bits	1024bits< l <10Kb	10Kb< l ≤100Kb
16.61	17.98	15.73	15.67	18.75

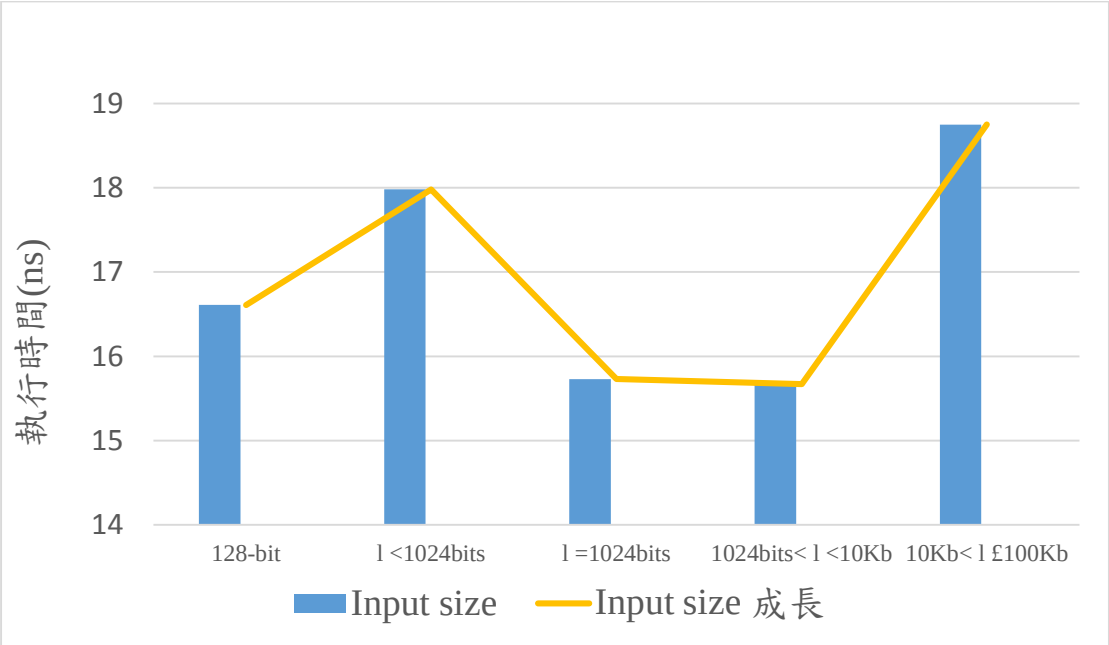

圖 5.3: GDBIS 在不同長度下的平均執行時間圖

表 5.4 呈現 GDBIS 的耗時介於 16.61～18.75μs，變化相當小，幾乎可以忽略輸入字串的長度，提衝使用者相當的便利性。

5.3 GDBRK 的效能

表 5.5：GDBRK 在 AES-128,192 與 256 的平均執行時間（為 20 萬次執行時間的平均值，單位為 μs）

AES-128 (128-bit, 10 round keys)	AES-192 (192-bit, 12 round keys)	AES-256 (256-bit, 14 round keys)
20.79	24.14	24.47

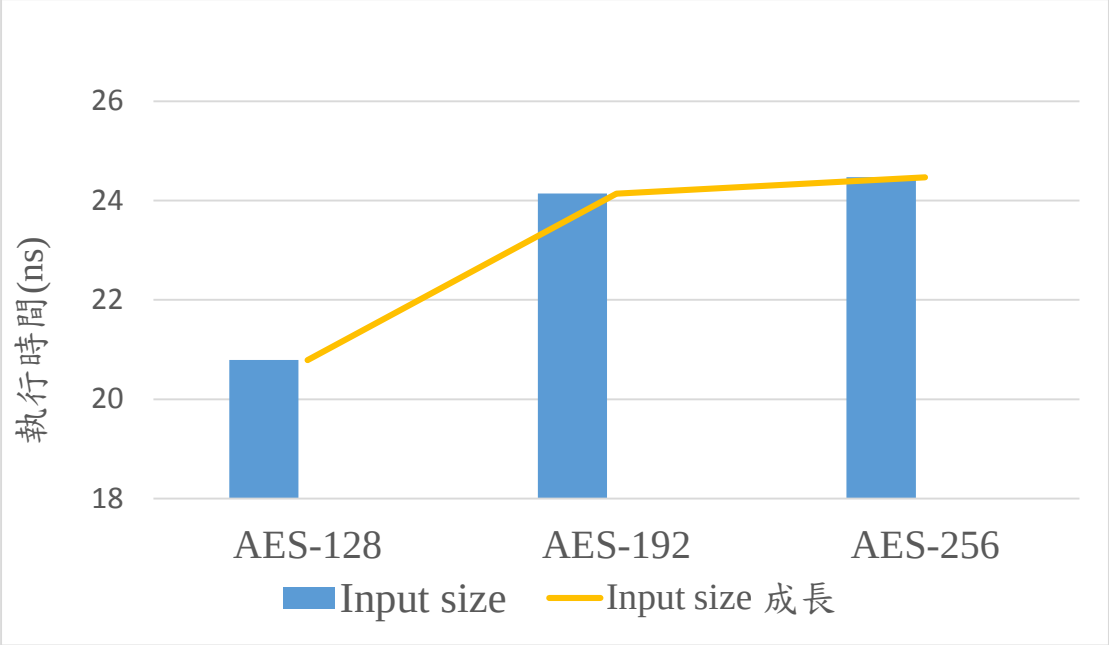

圖 5.4: GDBRK 在 AES-128,192 與 256 的平均執行時間圖

GDBRK 耗時介於 20.79～24.47μs 之間，此耗時極小且只於加密系統初始化時使用一次，不會影響後續之加/解密耗時。

5.4 AESEDB 效能分析與比較

在效能方面，圖 5.5 所示，AESEDB 將運算回合數由 AES 原先的 10 回合減至 5 回合，故效率增加了 2 倍。表 5.6 所示，AESEDB 之平均加密速度為 AES 的 2.1 倍，而 AESEDB 之平均解密速度為 AES 的 2.5 倍，AESEDB 在加密與解密的效能表現上皆優於 AES。

表 5.6：AES 與 AESEDB 實測執行速度比較表（單位為 ms）

Type / Input size	AES		AESEDB		AES/AESEDB	
	加密	解密	加密	解密	加密	解密
4KB	1.607	5.889	0.813	2.405	1.98	2.45
64KB	25.673	94.406	12.353	37.765	2.08	2.50
1MB	414.286	1508.101	198.125	604.196	2.09	2.50

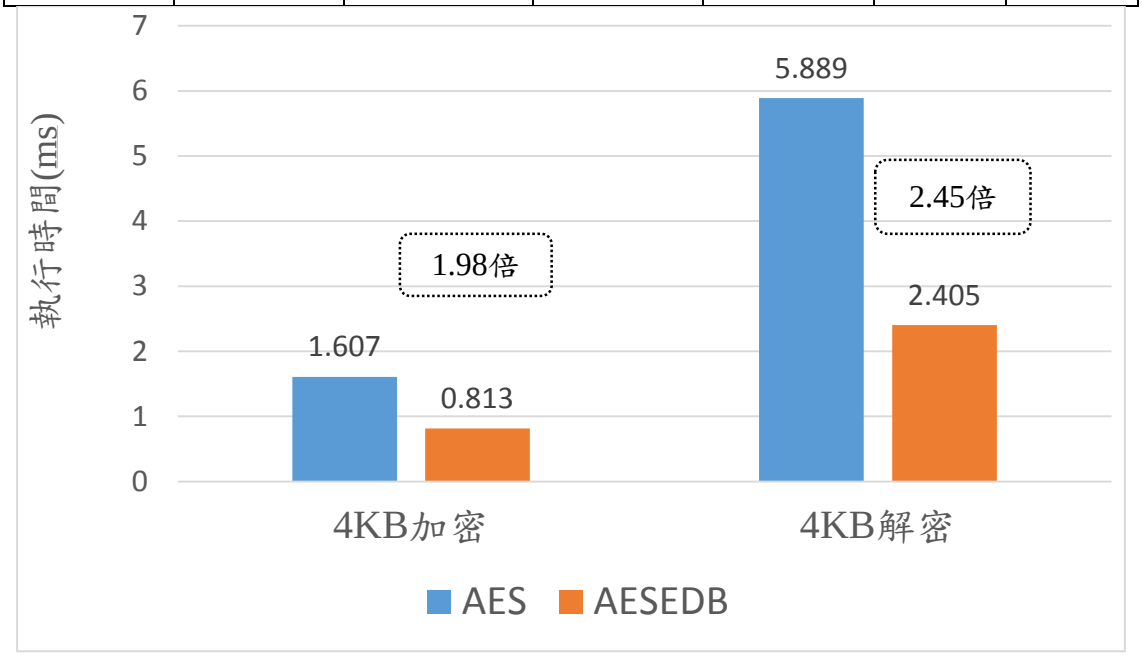

圖 5.5: AES 與 AESEDB 4KB 加解密執行速度比較圖

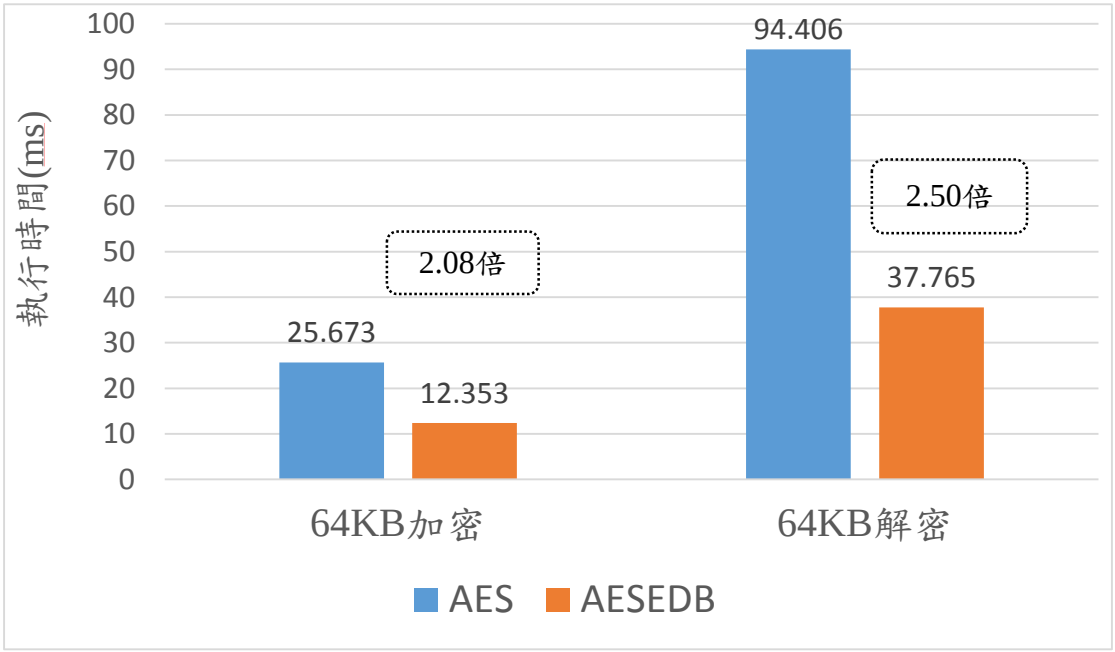

圖 5.6: AES 與 AESEDB 64KB 加解密執行速度比較圖

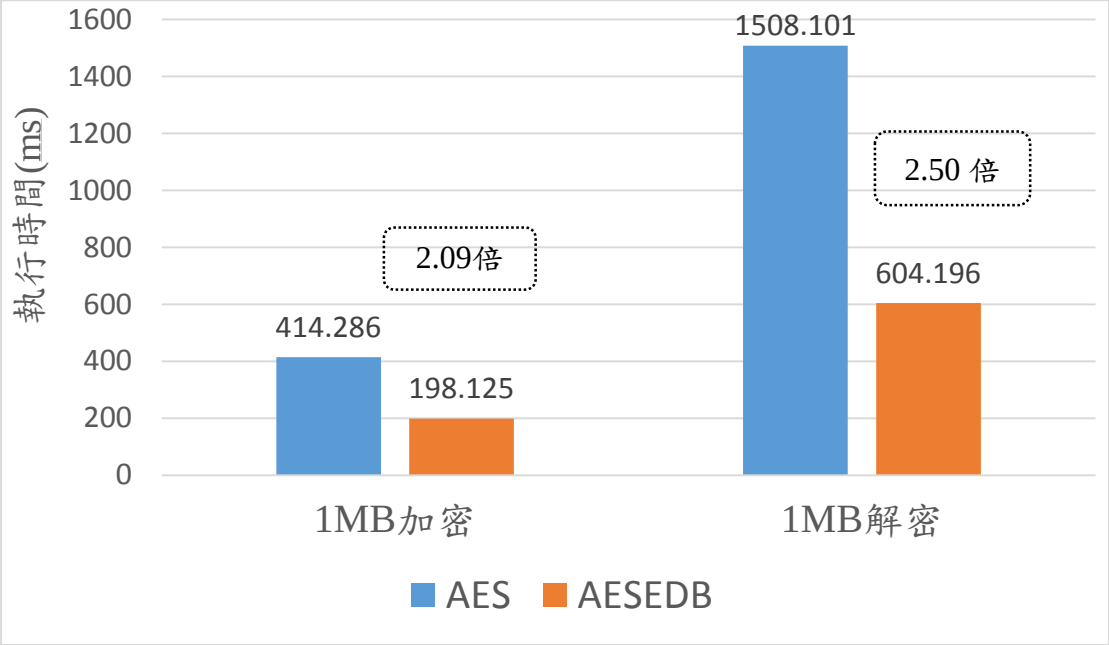

圖 5.7: AES 與 AESEDB 1MB 加解密執行速度比較圖

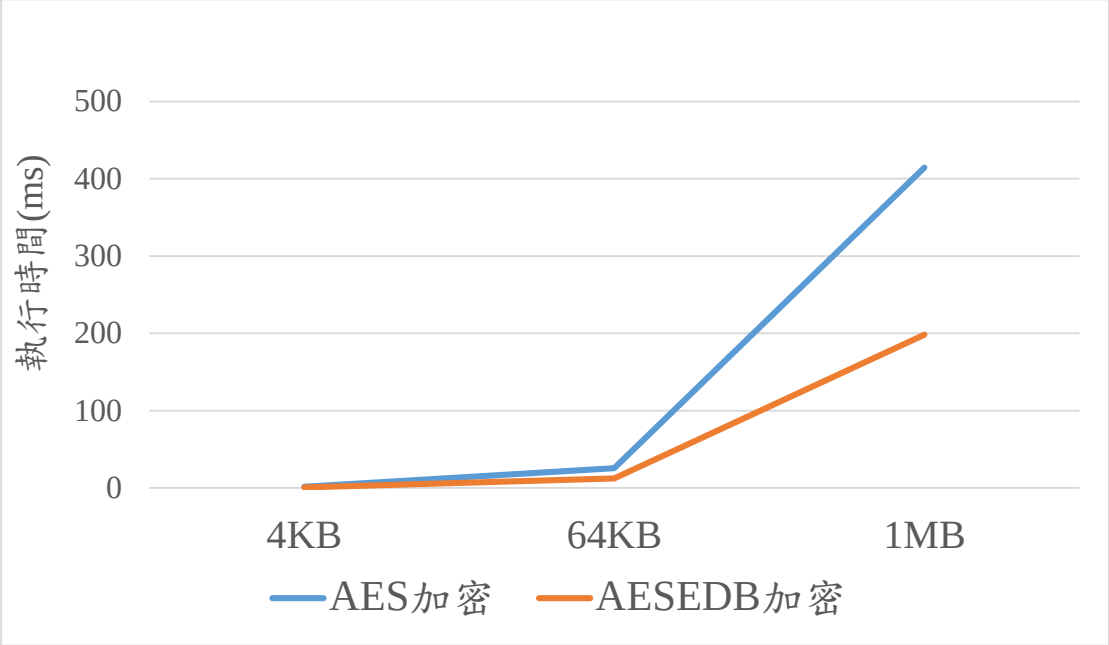

圖 5.8: AES 與 AESEDB 4KB~1MB 加密執行速度比較圖

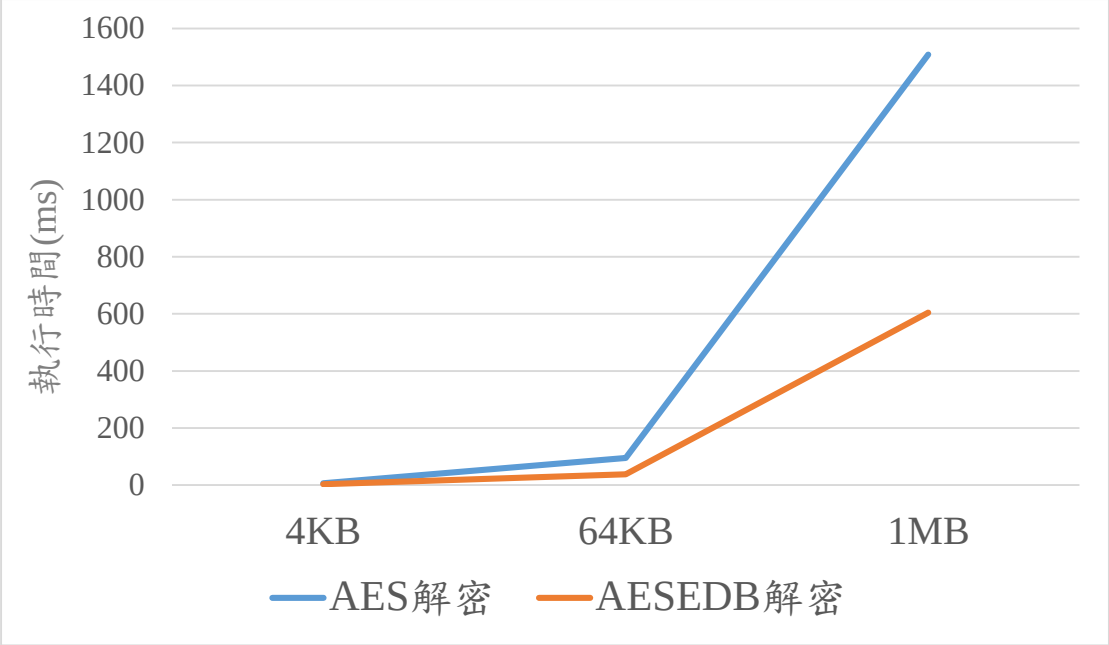

圖 5.9: AES 與 AESEDB 4KB~1MB 解密執行速度比較圖

第六章　　結論與展望

本研究開發出 GDBIS, GDBRK 與 AESEDB 等方法來改良 AES 的安全與效能，其特性、效能、貢獻分述如下：

(1) GDBIS 演算法，因密鑰字串長度介於 8 ~ 800000 bits，使得本法擁有即廣泛的輸入範圍，舉凡一句話或一段文章都可以成為密鑰，故，本法有著極佳的應用彈性與極高的安全性可以有效抵禦暴力法攻擊，更因本法不是透過修改 S-Box 而得到 D-Box，同時因繼承了 E-DASS 演算法的特性，使本法所產出的 D-Box 擁有極高的亂度、極佳的輸入敏感度與不可逆的特性。

(2) 新的金鑰擴充演算法，即，GDBRK，使用 *PW* 透過 GDBIS 來產生 D-Box，再用產生的 D-Box 作為查表的依據，整合使用三維運算與 E-DASS 演算法產生 Round keys，經由此法產生的 Round keys 不但擁有極高的亂度且 Round keys 之間的相互關聯度極低，大大增加了破密的困難度。

(3) 表 4.1 清楚呈現 AESEDB 在四項安全特性指標，即，密鑰長度、置換盒強度、回合金鑰強度與系統變數總長度等，皆比 AES 擁有更高的安全性，亦即，AESEDB 的安全度將遠高於 AES-128。

(4) 根據表 5.6 實際測試結果顯示，使用本研究提出之方法，即，AESEDB 其平均加密速度為 AES 的 2.1 倍，而平均解密速度為 AES 的 2.5 倍。

基於上述結論，本研究所提出之 AESEDB 具有快速、安全的特性，在未來，我們期望能在不損失安全度下，針對 AESEDB 之效能做進一步的提升，使 AESEDB 之速度能比 AES 快 10~20 倍。除了可以應用在改良 AES 外，本研究還可以應用於 SHA(Secure Hash Algorithms)與 HMAC(Hash-based message authentication code)的改良開發，使其同時能有更高安全與更佳效能，這些將是我們要努力突破的目標工作。

參考文獻

[1] *Announcing the Advanced Encryption Standard (AES)*, Federal Information Processing Standards Publication 197,*United States National Institute of Standards and Technology (NIST)*,November 26, 2001.

[2] A.Biryukov, O.Dunkelman, N.Keller, D. Khovratovich, and A.Shamir, "*Key Recovery Attacks of Practical Complexity on AES-256 Variants with up to 10 Rounds*," In: H. Gilbert, H. ed., EUROCRYPT, LNCS, 2010, vol. 6110, pp. 299–319.

[3] R.Li and C.Jin, "*Meet-in-the-middle attacks on 10-round AES-256*," Designs, Codes, and Cryptography, vol. 80, no. 3, pp. 459-471, Sep. 2016.

[4] A. Bogdanov, D. Khovratovich, C. Rechberger, "*Biclique Cryptanalysis of the Full AES*," 17th International Conference on the Theory and Application of Cryptology and Information Security, 2011, pp. 344-371.

[5] A. Bogdanov and D. Khovratovich, "*Related-key cryptanalysis of the full AES-192 and AES-256,*" 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. pp. 1-18.

[6] J. Kim, S. Hong and B. Preneel, "*Related-key rectangle attacks on reduced AES-192 and AES-256*," 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, pp. 225-241.

[7] P. Rogaway, M. Bellare, J. Black and T. Krovetz, "*OCB: A block-cipher mode of operation for efficient authenticated encryption*," eighth ACM Conference on Computer and Communications Security(CCS-8), Philadelphia, PA, USA, November 5, 2001, ACM Press, New York, pp. 196-205.

[8] G. Manjula and H. S. Mohan, "*Constructing key dependent dynamic S-Box for AES block cipher system*," 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), Bangalore, India, 21-23 July 2016, pp. 613-617.

[9] A. Alabaichi and A. I. Salih, "*Enhance security of advance encryption standard algorithm based on key-dependent S-box*," 2015 Fifth International Conference on Digital Information Processing and Communications (ICDIPC), Sierre, Switzerland, 7-9 Oct. 2015, pp. 44-53.

[10] S. Arrag, A. Hamdoun, A. Tragha and E. Khamlich Salah, "*Implementation of stronger AES by using dynamic S-box dependent of master key*," Journal of Theoretical and Applied Information Technology, vol. 53, no. 2, pp. 196-204, Jul. 2013.

[11] I. Hussain, T. Shah, M. A. Gondal, W. A. Khan and H. Mahmood, "*A group theoretic approach to construct cryptographically strong substitution boxes*," Neural Computing and Applications, vol. 23, no. 1, pp. 97-104, Jul. 2013.

[12] T. Shah, I. Hussain, M. A. Gondal and H. Mahmood, "*Statistical analysis of S-box in image encryption applications based on majority logic criterion*," International Journal of Physical Sciences, vol. 6, no. 16, pp. 4110-4127, Aug. 2011.

[13] R. Hosseinkhani and H. H. S. Javadi, "*Using cipher key to generate dynamic S-box in AES cipher system*," International Journal of Computer Science and Security, vol. 6, no. 1, pp. 19-28, 2012.

[14] E. M. Mahmoud, A. A. El Hafez, T. A. Elgarf and Z. Abdelhalim, "*Dynamic AES-128 with Key-Dependent S-box*," International Journal of Engineering Research and Applications, vol. 3, no. 1, pp. 1662-1670, Feb. 2013.

[15] V. Kaul, V. A. Bharadi, P. Choudhari, D. Shah and S. K. Narayankhedkar, "*Security Enhancement for Data Transmission in 3G/4G Networks*," 2015 International Conference on Computing Communication Control and Automation, Pune, India, 26-27 Feb. 2015, pp. 95-102.

[16] F. H. Nejad, S. Sabah and A. J. Jam, "*Analysis of avalanche effect on advance encryption standard by using dynamic S-Box depends on rounds keys*," 2014 International Conference on Computational Science and Technology (ICCST), Kota Kinabalu, Malaysia, 27-28 Aug. 2014, no. 7045184.

[17] X. Wang and Q. Wang, "*A novel image encryption algorithm based on dynamic S-boxes constructed by chaos*," Nonlinear Dynamics, vol. 75, no. 3, pp. 567-576, Feb. 2014.

[18] Y. L. Huang, C. R. Dai, F. Y. Leu and I. You, "*A secure data encryption method employing a sequential-logic style mechanism for a cloud system*," International Journal of Web and Grid Services, vol. 11, no. 1, pp. 102-124, Jan. 2015.

[19] Y. L. Huang, F. Y. Leu, I. You, R. Y. Su, P. H. Su and H. C. Chen, "*A 3D Encryption with Shifting Mapping Substitution Mechanism*," The 5th IEEE CCNC International Workshop on Security and Cognitive Informatics for Homeland Defense (SeCIHD 2017) , Las Vegas, USA, 8-11 Jan. 2017.

[20] Y. L. Huang, F. Y. Leu, P. H. Su, T. H. Sung and S. C. Liu, "*A Secure and High Performance Wireless Sensor Network Based on Symmetric Key Matrix*," Tenth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS-2016) , Fukuoka Institute of Technology (FIT), Fukuoka, Japan, 6-8 July 2016.