東海大學資訊工程研究所

碩士論文

指導教授：黃宜豐

使用改良 AES 架構之分離結構加密法

Separate Structure Cryptography by Using Improved AES Architecture

研究生：陳俊穎

中華民國 一百零八年 一月

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 陳 俊 穎 所提之論文

使用改良 AES 架構之分離結構加密法

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 黃宜豐

呂芳懌

楊伏夷

指 導 教 授 黃宜豐 簽章

中華民國 108 年 1 月 7 日

摘 要

無線通訊在 2020 將是 5G 時代，其下載傳輸速率介於 1~10Gbps，約為 4G-LTE 的 40 倍速，是而目前支援 4G 加/解密的 AES，其效能將無法達到 5G 的要求，此外，AES 亦正面臨被破密的危機，如何創新構建一全新的訊息加密機制，使其能有更高的安全與效能，為當今重要的研究課題；基於此，本研究提出一全新的加密方法，稱為"使用改良 AES 架構之分離結構加密法(Separate Structure Cryptography by Using Improved AES Architecture, SSCUIA for short)"，其中分離結構加密機制為本研究提出的一創新機制，可以有效提升整體加/解密之安全與效能，本研究另一特色為使用改良 AES 架構(Improved AES Architecture, IAESA for short)來增強系統整體安全度。IAESA 使用一全新設計的金鑰擴展演算法，稱為 GDBRS (Generating Dynamic Box, Round Keys and Shifting Key)用來取代 AES 原有的金鑰擴展法，在 GDBRS 中擴展輸入密碼(*PW*) 的範圍，其長度介於 8 ~ 800000 bits，並藉由此 *PW* 產生一 16X16 動態盒(D-Box)，接著以此 D-Box 來產生回合金鑰與位移金鑰；在 IAESA 中，由於 GDBRS 的作用，能有效提升 D-Box、回合金鑰與位移金鑰的安全性，而藉由 Shifting-SubBytes (S-SubBytes, for short)運算取代 AES-SubBytes 運算，可以增益安全強度，使得將原本 AES 的 10 回合運算縮減為 3 回合，仍能不損失安全度而有效改善效能。經由理論分析比較與效能實測證明，結合 IAESA 與分離結構加密機制的 SSCUIA 將比 AES-128 更加安全與快速。

關鍵詞：5G, AES, SSCUIA, IAESA, GDBRS, D-Box

目錄

摘 要 i

第一章 研究動機與目的 1

1.1 研究背景與動機 1
1.2 研究目的 1
1.3 論文架構 1

第二章 文獻回顧與探討 2

2.1 AES 相關文獻 2
2.2 3D 運算的相關文獻 2
2.3 DASS 相關文獻 3
2.4 D-Box 相關文獻 3

第三章 使用改良AES架構之分離結構加密法 5

3.1 參數與函數 5
3.2 運算子 5
3.3 S_SubByte 6
3.4 Generate a 16 x 16 Dynamic Box from a Password 6
3.5 Improved AES Architecture 10
3.6 Separate Structure Cryptography by Using Improved AES Architecture 13

第四章 安全分析與比較 17

4.1 DASS 之安全分析 17
4.2 GDBPW 之安全分析 18
4.3 IAESA 之安全分析 18
4.4 SSCUIA 之安全分析與比較 19

第五章 效能分析與比較 23

5.1 基本運算子之效能表現 23
5.2 DASS 與 GDBPW 之效能表現 23
5.3 IAESA 效能分析 24
5.4 SSCUIA 效能分析 25

第六章 結論與展望 27

參考文獻 28

圖目錄

圖3-1 IAESA之程序圖 11

圖3-2分離結構加密法加密流程圖 14

圖5-1 DASS對於不同輸入長度之平均執行時間(1,250萬次平均時間；單位:奈秒)

.....24

圖5-2 GDBPW對於不同輸入長度之平均執行時間 (200萬次平均時間；L為Input length；單位:奈秒) 24

圖5-3 SSCUIA與AES理論評估執行速度相差倍率折線圖 26

表目錄

表4-1 IAESA與AES-128安全特性比較.....19

表4-2　SSCUIA、IAESA與AES安全特性比較.....21

表5-1 模擬測試運行環境之規格.....23

表5-2 五種運算子對於常用運算寬度之平均運行時間(單位:奈秒).....23

表5-3　AES區塊耗時表(微秒).....25

表5-4 AES與SSCUIA理論評估執行速度表(微秒).....25

表5-5 SSCUIA與AES理論評估執行速度比較表與倍率(微秒).....26

第一章 研究動機與目的

1.1 研究背景與動機

近年來，無線通訊的產品已經完全融入人們的日常生活之中，而這其中最貼近我們生活的便是行動電話，隨著智慧型手機的蓬勃發展，手機成為我們生活中不可或缺的必需品，在這種環境之下，為了滿足業界與消費者的需求，各家業者投入大量資金研究開發第五代行動通訊技術(5G) [1]。

下一代行動網路聯盟認為，5G 應在 2020 年陸續推出，以滿足企業和消費者的需求 [2]。在傳輸速度方面，5G 能夠提供極快的傳輸速度，約為 4G 網絡的 40 倍 [1] [2]，且時延很低。現役的 4G 使用 AES 來進行資料的加/解密，而目前暫時沒有新的加密演算法能夠取代 AES，若未來 5G 仍然使用 AES 作為加密演算法的話，其面臨的挑戰即是 AES 的安全與速度是否符合未來的需求?在近幾年,針對破解 AES 的研究可說是層出不窮 [3] [4] [5] [6]，AES 的安全問題是我們即將面臨的一大考驗，此外在 5G 的傳輸速率是 4G 的 40 倍要求下[7]，其下載傳輸速率約介於 1~10Gbps[8]，而目前 AES 的加/解密速率是 425.92Mbps [9]，將明顯不足，若沒有提升的話，會面臨不及 5G 傳輸速度而形成效能瓶頸。

1.2 研究目的

基於現役 AES 所面臨的效能不足與即將被破密的密切危機，本研究提出一改良 AES 架構的方法，稱為 “使用改良 AES 架構之分離結構加密法 (Separate Structure Cryptography by Using Improved AES Architecture, SSCUIA for short.)”，期望能有效提升安全與效能而能夠符合即將到來 5G-LTE 的加/解密需求。

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，描述了使用改良 AES 架構之分離結構加密法； 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 相關文獻

高級加密標準（AES）於 2001 年 11 月 26 日由美國國家標準與技研究院（NIST）發佈為 FIPS PUB 197，自 2002 年 5 月 26 日起成為密碼學標準。它主要由四個步驟組成， SubBytes，ShiftRows，MixColumns 和 AddRoundKey，密鑰大小為 128,192 或 256 位元，取決於所需的安全級別。在 AES 加密過程中，相同的操作重複 10 次，12 次或 14 次（也稱為回合數）。賦予該算法的密鑰被擴展成長度為 4 bytes 的 44, 52 或 60 個字組。AES 的加密，首先將明文區塊與第 0 回合金鑰(RK_0)進行 XOR 運算，以產生相應的密文塊，稱為狀態矩陣，接著循序做多次的回合運算，而在最後一回合省略了 MixColumns 運算，即只執行三個步驟。

每回合加密運算均包含 4 個步驟，第一步驟 SubBytes，是透過一個固定非線性的置換盒(S-BOX)[10]，用查表的方式依序將狀態矩陣中每個元素替換成對應的密文。第二步驟的 ShiftRows，將狀態矩陣中的第 i 列做左旋 i 位元組，$0 \le i \le 3$。第三步驟是 MixColumns，這個步驟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。最後步驟 AddRoundKey，狀態矩陣中的每一元素都與該次回合金鑰（round key）相對應的位元組做 XOR 運算，而回合金鑰是經由金鑰擴展程序產生。

由於 AES 加密過程需要 10,12 或 14 回合來加密明文塊，致使其效能不足以滿足 LTE-5G 通信環境的性能要求[1]，此效能不足的缺陷是目前 AES 面臨的首要問題。

在 2010 年，Orr Dunkelman,Nathan Ke-ller 和 Adi Shamir 提出單鑰攻擊改良法來破解 AES[11],此法比窮盡搜索更具效率，他們的方法從著名的 Demirci-Sel¸cuk 攻擊中，可以對 7 回合 AES 做到更具效率的破解。由於 AES 使用固定的 S-Box，使得 SubByte 運算輸出的密文呈現線性對應的安全缺陷，因而大大降低 AES 的安全性

2.2 3D 運算的相關文獻

在本研究中，GDBRS 演算法是整合了許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，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) [12]，此方法具備三種特色，使用循序邏輯進行加解密、使用三個基本運算子進行運算以及使用動態轉換表，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[13]，由$\odot_R/\odot_{IR}$取代$\odot/\odot$來增加此第三運算子的強度，因此在本研究中，我們便使用$\oplus/\oplus$、$+_2/-_2$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。

2.3 DASS 相關文獻

在 2017 年 Generating Dynamic Box by using an Input String [14]研究中，Huang et al.提出一種演算法 Dynamically Accumulated Shifting Sub-stitution (DASS) ，依據密文所給的內容，改變其累積值(Accumulated)，之後藉由累積值所作為之位移值，並查詢在 16 x16 的映射在隨機盒位置之內容，並將其內容串聯，形成密文，其密文的敏感度高且亂度亂，用於加密法中可以提升其安全，DASS 工作方式如下:

Dynamically Accumulated Shifting Substitution (DASS) algorithm: The DASS is a one-way function to convert a plaintext input to an irreversible ciphertext output.

Algorithm 2: DASS

Input: an n-bit plaintext block (n is a multiple of 8), and a 16×16 random table-box (RT-Box)

Output：an n-bit ciphertext block C

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

dsc=256; //dsc :dynamic shifting counter

for i=1 to k

{ $vp[i]$=(int)p_i;

$dsc=dsc+(vp[i]+1)\times i$;}

$vp[0]=vp[1]+vp[k]\times k$;

for i=1 to k

{ $dsc=(dsc+(vp[i-1]+1)\times i) \bmod 65536$;

$vch=(vp[i]+dsc) \bmod 256$;

c_i= the content obtained by looking up the RT-Box by referring to vch, i.e, the content of the $\left\lfloor\frac{vch}{16}\right\rfloor$-th row and ($vch$ mod 16)-th column in the RT-Box;}}

2.4 D-Box 相關文獻

在 2013 年，E. M. Mahmoud et al.提出一種產生 Dynamic Key De-pendent S-box 的方法來改良 AES 的安全性[15]，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。在 2014 年，Prerana Choudhari、Vikas Kaul 與 S K Narayankhed-kar 曾發表動態 S-Box 之相關研究[16]，其方法是將 S-Box 以線性方式重排來達成動態的效果，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[17][18]，然而，這些方法都是以重排固定的 S-Box 來產生動態 S-Box，由於其皆有共同的原點，即 S-Box，單就此限制，使得這些以重新排序的方法來產生的動態 S-Box，其在盒子的亂度與輸入敏感度上都略嫌不足。

經由相關參考文獻的回顧，由於 S-Box 是固定且眾所周知的，此為 AES 在安全上可以著墨改良的弱點，其次在金鑰擴展的部分，因其前後金鑰主要藉由⊕計算產生，亦為 AES 在安全上的另一弱點；在效能方面，至少 10 回合的相同結構迴圈計算，使得效能略顯不足，如何減少迴圈數而不減低安全度，亦為可以改進的方向；因此本篇論文，藉由 D-Box 取代 S-Box 而改良的 AES 架構並結合分離結構加密法，進而實現了安全與效能的有效改良。

第三章 使用改良 AES 架構之分離結構加密法

本研究，使用藉由動態盒生成與改良 AES 架構之分離結構加密法，其工作原理如下：

3.1 參數與函數

AES 架構之分離結構加密法所用到參數與函數定義如下:

1) D-Box : 動態盒(Dynamic-Box)，為一 16x16 的盒子，根據一定方法由輸入之密碼(Password, *PW* for short)來產生。
2) ISK_s : Initial Serial Keys，由輸入之 *PW* 依據特定演算法所產生之兩組初始序列金鑰，包含 ISK_1 與 ISK_2，其長度為 1024 bits。
3) DASS : DASS (Dynamically Accumulated Shifting Substitution)，進階動態累積位移取代演算法，是將明文加密為不可逆之密文的單向函數演算法。
4) *DK*s : Dynamic Keys，由 ISK_1 與 ISK_2 透過 DASS 與三維運算 [5]所產生之三組動態金鑰，包含 DK_1，DK_2 與 DK_3，其長度為 1024-bit。
5) *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。
6) *IA*1 : 1st Insert Array,透過旗幟狀態陣列 FA 之元素值，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。
7) *IA*2 : 2nd Insert Array,透過旗幟狀態陣列 FA 之元素值，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。
8) *IAR* : Residual Insert Array,在經過 DK_1，DK_2 與 DK_3 的選用後，*FA* 中沒有被選用的所有剩餘之索引值所形成的陣列。
9) *IA*3 : 3rd Insert Array,將 *IA*2, *IAR* 與 *IA*1 串接後，再進行重排所形成的陣列。
10) *ALi* : Actual length of *IAi*，$1 \leqq i \leqq 3$，*IA*1、*IA*2 及 *IA*3 的實際長度。
11) *ALR* : Actual length of *IAR*，*IAR* 的實際長度。

3.2 運算子

令明文為 *p*，密文為 *c*，加密金鑰為 *k*，則⊕、$+_2$、$\odot_R$ 三個基本算子加/解密之定義如下:

⊕, Exclusive-OR operator :

加密：$\boldsymbol{c = p \oplus k}$;

解密：$\boldsymbol{p = c \oplus k}$;

$+_2$, Binary-Addition operator :

加密：$\boldsymbol{c = p +_2 k}$, 為 p、k 兩運算元進行二進制加法，過程中捨去其最高位元之進位。;

解密：$\boldsymbol{p = c -_2 k} \begin{cases} \boldsymbol{c - k, if\ c \geq k} \\ \boldsymbol{c + \bar{k} + 1, if\ c < k} \end{cases}$;

$\odot_R$, Rotate Equivalence operator :

加密：$\boldsymbol{c = p \odot_R k = p_R \odot k}$，其中 p_R 是由明文 p 順時針旋轉位移$\frac{|\boldsymbol{p}|}{4}$個位元所得到;

解密：$\boldsymbol{p = c \odot_{IR} k}$ =將$(\mathbf{c} \odot \boldsymbol{k})$逆時針旋轉位移$\frac{|\boldsymbol{p}|}{4}$個位元；

3.3 S_SubByte

本程序全名為 Shifting_SubByte(S_SubByte, for short), 先將狀態矩陣元素與相對應之位移金鑰(SK, for short)位元組做$+_2$運算後，再對 D-Box 進行 SubByte 操作。/* format is S_SubByte(*p*, D-Box, *SK*) */

其反運算為 InvS_SubByte。

Input : *p* , a D-Box and *SK*

Output : *c*

1) *c* = *p*$+_2$*SK;*
2) *c* =SubByte(*c* , D-Box);
3) return *c* ;

3.4 Generate a 16 x 16 Dynamic Box from a Password

本法，Generate a 16 x 16 Dynamic Box from a Password (GDBPW, for short)由三個程序構成，即，(1) Generation of Initial Serial Keys、(2) Generation of Dynamic Keys、(3) Dynamic Box Generator 演算法(簡稱 DBG 演算法)，茲分述如下：。

3.4.1Generation of Initial Serial Keys

本程序藉由輸入的密碼(*PW*)來產生兩個初始序列金鑰 ISK_1 與 ISK_2，其產生過程如下:

Algorithm 2:

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

Input : *PW* and a R-Box

Output : ISK_1, ISK_2

1) If $\boldsymbol{|PW| < 1024}$, then{

$\boldsymbol{ISK_1}$ =DASS_expansion(*PW*, R-Box);

$\boldsymbol{ISK_2}$ =Binaryadder_expansion(*PW*);}

2) If $|\boldsymbol{PW}| = \mathbf{1024}$, then{

$\boldsymbol{ISK_1} = \boldsymbol{PW}$;

$\boldsymbol{ISK_2} =$ DASS(*PW*, R-box);}

3) If $|\boldsymbol{PW}| > \mathbf{1024}$, then{

(a) $\boldsymbol{PW} = \boldsymbol{p_1 p_2 p_3} \ldots \boldsymbol{p_m}$,where $|\boldsymbol{p_i}| = \mathbf{1024}, \mathbf{1} \leq \boldsymbol{i} < \boldsymbol{m}, \boldsymbol{m} \geq \mathbf{2}$;

(b) If $|\boldsymbol{p_m}| < \mathbf{1024}$, then $\boldsymbol{p_m} =$ Binaryadder _expansion($\boldsymbol{p_m}$);

(c) $\boldsymbol{ISK_1} = \boldsymbol{p_1} \oplus \boldsymbol{p_2} \oplus \ldots \oplus \boldsymbol{p_m}$;

(d) $\boldsymbol{ISK_2}$ =DASS($\boldsymbol{ISK_1}$,R-box);}

前述中，若 *PW* 的長度小於 1024 bits，將會使用下列兩個演算法來擴充其長度，使得$|\boldsymbol{PW}| = \mathbf{1024}$，此兩個演算法如下:

Algorithm 3:

The DASS_expansion algorithm, which expands input data to length 1024 bits by employing the DASS algorithm. /* format is DASS_expansion(*P*, R-Box) */

Input: a string *P* with $|\boldsymbol{P}| < \mathbf{1024}$ and a R-Box.

Output: an expanded string *P* with $|\boldsymbol{P}| = \mathbf{1024}$.

1) While $|\boldsymbol{P}| < \mathbf{1024}$ do
2) *P'* = DASS(*P*, R-box);
3) $\boldsymbol{P} = \boldsymbol{P'}||\boldsymbol{P}||\boldsymbol{P'}$;
4) End while
5) *P* = DASS(*P*, R-box);
6) return Right(*P*, 1024); /* return the 1024 right most bits of *P* */

Algorithm 4:

The Binaryadder_expansion algorithm, which expands input data to length 1024 bits by employing the Binary addition. /* format is Binaryadder_expansion(*P*) */

Input: a string *P* with $|\boldsymbol{P}| < \mathbf{1024}$.

Output: an expanded string *P* with $|\boldsymbol{P}| = \mathbf{1024}$.

1) While $|\boldsymbol{P}| < \mathbf{1024}$ do
2) $\boldsymbol{P} = (\boldsymbol{P} \parallel \boldsymbol{P}) +_{\mathbf{2}} (\boldsymbol{P} \parallel \boldsymbol{P})$;
3) end while
4) return Right(*P*, 1024); /* return the 1024 right most bits of *P* */

3.4.2Generation of Dynamic Keys

本程序整合使用 DASS 演算法與 3D 運算，將 ISK_1 與 ISK_2 依序混亂擴展成三個動態金鑰，即，DK_1, DK_2 與 DK_3，其產生方式如下：

Algorithm 5:

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

Input: ISK_1, ISK_2 and a R-Box

Output: DK_1, DK_2 and DK_3

1) $DK_1 =$ DASS($ISK_1 \oplus ISK_2$, S-Box);

2) $DK_2 =$ DASS($ISK_1 +_2 ISK_2$, S-Box);

3) $DK_3 =$ DASS($DK_1 \odot_R DK_2$, S-Box);

3.4.3Dynamic Box Generator Algorithm

本演算法是藉由三個長度為 1024 位元的動態金鑰，即，DK_1, DK_2 與 DK_3，來產生一個 16X16 動態盒，其內容如下：

Algorithm 6:

The Dynamic Box Generator algorithm (DBG, for short) /*format: DGB(DK_1, DK_2, DK_3) */

Input: DK_1, DK_2 and DK_3

Output: D-Box

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

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

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

*IA*1[128]; *IA*2[256]; *IAR*[256]; *IA*3[256]; /* Candidate arrays*/

*AL*1=0; *AL*2=0; *ALR*=0; *AL*3=0; /* Actual length of candidate arrays */

FA[256]; /* Taken flag array */

D-Box[256]; /* Element array of dynamic box */

For *j* = 0 to 255

    FA[*j*] = “F”;

Next *j*

For *j* = 0 to 127

    KA[*j*] = Int(KA_j);

    KB[*j*] = Int(KB_j);

    KC[*j*] = Int(KC_j);

Next *j*

For *i* = 0 to 127 /* Filling *IA*1 from DK_1 */

    If (*FA*[*KA*[*i*]]=”F”) then

        { *IA*1[*AL*1] = *KA*[*i*]; *FA*[*KA*[*i*]] = "T";

          *AL*1 = *AL*1+1;}

Next *i*

For *i* = 0 to 127 /* Filling *IA*2 from DK_2 */
If (*FA*[*KB*[*i*]]=”F”) then
{ *IA*2[*AL*2] = *KB*[*i*]; *FA*[*KB*[*i*]] = "T";
*AL*2 = *AL*2+1;}
Next *i*

For *i* = 0 to 127 /* Filling *IA*2 from DK_3 */
If (*FA*[*KC*[*i*]]=”F”) then
{ *IA*2[*AL*2] = *KC*[*i*]; *FA*[*KC*[*i*]] = "T";
*AL*2 = *AL*2+1;}
Next *i*

For *i* = 0 to 255 /* Remained values */
If (*FA*[*i*]=”F”) then
{ *IAR*[*ALR*] = *i*; *ALR* = *ALR*+1;}
Next *i*;

For *i* = 0 to *ALR*–1 /* Expanding *IA*2 by concatenating *IAR* */
*IA*2[*AL*2+*i*] = *IAR* [*i*] ;
Next *i*;

For *i* = 0 to *AL*1–1 /* Expanding *IA*2 by concatenating *IA*1 */
*IA*2[*AL*2+ *ALR* +*i*] = *IA*1 [*i*] ;
Next *i*;

h_1=0; h_2=0; *jup*=0;
For *j* = 0 to 41
h_1= h_1 + *KC*[*j*];
h_2= h_2 + *KC*[42+*j*];;
jup= *jup* + *KC*[84+*j*];
Next *j*
h_1= h_1 *mod* 256; h_2= h_2 *mod* 256;
jup= *jup mod* 128;
If (*jup* is even) then *jup*= *jup* +129;

For *i* = 0 to 255 /* Rearrange *IA*2 to obtain *IA*3 */

$IA3[i] = IA2[(h_1+i\times 29) \ mod \ 256]$;

Next *i*;

For $i = 0$ to 255 /* Filling D-Box */

D-Box[i] = $IA3[(h_2+i\times jup) \ mod \ 256]$;

Next *i*;

3.4.4 GDBPW

Algorithm 7:

The Generate a 16 x 16 Dynamic Box from a Password (GDBPW, for short) algorithm /* format is GDBPW(*PW*, R-Box) */

Input: *PW*, R-Box

Output: D-Box

1) Call GISK(*PW*, R-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) return D-Box

3.5 Improved AES Architecture

本架構的 IAESA，由兩個部分所組成，金鑰擴充演算 GDBRS 和改良 AES 架構兩部分，改良 AES 架構將回合數由 10 回合減至 3 回合，並由 GDBRS 作為金鑰擴充，下列兩小節將做更詳細的分述:

3.5.1 Generation of D-Box Round Keys and Shifting Key

本研究認為要在不損失 AES 效能下而可以有效加強 AES 的安全度，最直接有效的作法是重新設計 AES 回合金鑰擴充演算法，目前的 AES 回合金鑰擴充演算法雖然快速，但因前後回合金鑰的產生，主要為⊕的可逆運算，使得前後回合金鑰有著極強的關聯度而大大降低了安全強度，因此，若能改良金鑰擴充演算法，首先由 *PW* 取代 Cipher Key，如此可以不受 128 位元長度的限制，接著由 *PW* 來產生 D-Box，並用此 D-Box 來產生回合金鑰與取代目前的 S-box，則前後回合金鑰的關聯度將大為降低而 SubBytes 的安全度將會明顯提升。以下是本研究提出的回合金鑰擴充改良方法:

Algorithm 8:

The Generation of D-Box, Round Keys and Shifting Key (GDBRS, for short) algorithm /* format is GDBRS(*PW*) */

Input: *PW*

Output: D-Box, Round Keys (RK_0~RK_3) and

Shifting Key (SK_1~SK_3)

1) Call GDBPW(*PW*,S-box) to obtain D-Box;
2) TDK_1 = DASS($DK_1 \oplus DK_2$, D-Box);
3) TDK_2 = DASS($TDK_1 \oplus DK_3$, D-Box);
4) *RKT* = DASS($TDK_1 \oplus TDK_2$, D-Box)
5) *SKT* = DASS($TDK_1 +_2 TDK_2$, D-Box);
6) RK_j = Mid(*RKT*⊕*SKT*, 128×(*j*+2), 128), $0 \leqq j \leqq 3$; /* |*RKT*|=|*SKT*|=1024, $|RK_j|$=128,$0 \leqq j \leqq 3$ */
7) SK_j = Mid($RKT +_2 SKT$, 128×(*j*+3), 128), $1 \leqq j \leqq 3$; /* |*RKT*|=|*SKT*|=1024, $|SK_j|$=128,$1 \leqq j \leqq 3$ */

3.5.2 改良 AES 架構

本研究改良 AES 架構(Improved AES Architecture, IAESA for short.)，分成兩部分，第一部分為 key expension 的部分，改良成 GDBRS，從中生成 D-Box 取代原本的 S-Box，並生成回合金鑰和位移金鑰，第二部分，AES 的回合加密，由 10 回合縮減至 3 回合，其中兩回合的 SubBytes，改良成 S-SubBytes，兩部分的整合稱為 IAESA，其加密流程圖如圖 3-1 所示。

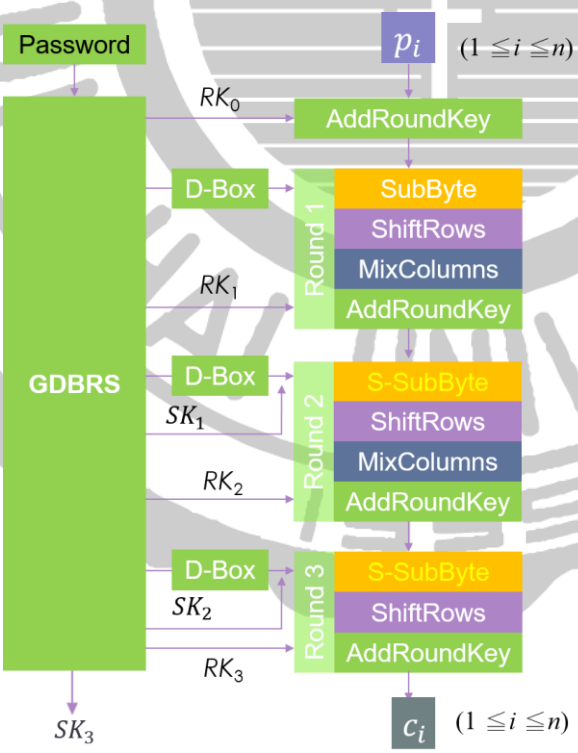

圖3-1 IAESA之流程圖

Algorithm 9 為 IAESA 加密流程詳情，如下:

Algorithm 9:

Improved AES Architecture_Encryption algorithm (IAESA, for short) /* format is IAESA(*P,PW*) */

Input: Plaintext *P*, *PW*

Output: Ciphertext *C*

1) Let $P=p_1p_2\ldots p_n$, $|p_j|=128, 1 \leqq j \leqq n$;
2) Let $C=c_1c_2\ldots c_n$, $|c_j|=128, 1 \leqq j \leqq n$;
3) CALL GDBRS(*PW*) to obtain D-Box, $RK_0 \sim RK_3$ and $SK_1 \sim SK_3$
4) For i=1 to n

   p_i =AddRoundKey(p_i, RK_0);

   /*Round 1 */

   p_i =SubByte(p_i, D-Box);

   p_i =ShiftRows(p_i);

   p_i =MixColumns(p_i);

   p_i =AddRoundKey(p_i, RK_1);

   /*Round 2 */

   p_i =S_SubByte(p_i, D-Box, SK_1);

   p_i =ShiftRows(p_i);

   p_i =MixColumns(p_i);

   p_i =AddRoundKey(p_i, RK_2);

   /*Round 3*/

   p_i =S_SubByte(p_i, D-Box,SK_2);

   p_i =ShiftRows(p_i);

   c_i=AddRoundKey(p_i, RK_3);

5) return *C*;

Algorithm 9:

Improved AES Architecture_Decryption algorithm (IAESA, for short) /* format is I-IAESA(*P,PW*) */

Input: Ciphertext *C*, *PW*

Output: Plaintext *P*

1) Let $C=c_1c_2\ldots c_n$, $|c_j|=128, 1 \leqq j \leqq n$;
2) Let $P=p_1p_2\ldots p_n$, $|p_j|=128, 1 \leqq j \leqq n$;
3) CALL GDBRS(*PW*) to obtain D-Box, $RK_0 \sim RK_3$ 、$SK_1 \sim SK_3$;
4) For i=1 to n

   c_i =AddRoundKey(c_i, RK_3);

   /* Round 1 */

c_i =InvShiftRows(c_i);

c_i =InvSubByte(c_i, Inverse D-Box,SK_2);

c_i =AddRoundKey(c_i, RK_2);

c_i =InvMixColumns(c_i);

/* Round 2 */

c_i =InvShiftRows(c_i);

c_i =InvS_SubByte(c_i, Inverse D-Box, SK_1);

c_i =AddRoundKey(c_i, RK_1);

c_i =InvMixColumns(c_i);

/*Round 3*/

c_i =InvShiftRows(c_i);

c_i =InvSubByte(c_i, Inverse D-Box);

p_i=AddRoundKey(c_i, RK_0);

5) return P

3.6 Separate Structure Cryptography by Using Improved AES Architecture

本研究，使用改良 AES 架構之分離結構加密法(Separate Structure Cryptography by Using Improved AES Architecture, SSCUIA for short)，通過採用具有動態盒的改良 AES 架構對明文的第一區塊加密並使用其產生的密文區塊(C_1)與動態加密金鑰群提供給後續的明文區塊加密之用，其中，分離結構加密機制是指將構成明文檔案的 n 個區塊分離為第一個區塊(P_1)與 2-to-n 區塊($P_2 \sim P_n$)兩部分；首先，使用初始加密程序將第一個明文區塊(P_1)加密，並將其產生的密文區塊(C_1)與關聯產出的三個動態加密金鑰 i.e. (SK_3, TK_1 and TK_2)提供後續 2-to-n 區塊加密之用，其次，使用循序邏輯加密方式對 2-to-n 區塊進行加密，其中所產出的金鑰長度、明文/密文區塊長度，除 PW 外，均為 1024 位元，而 PW 的長度介於 8~800000 位元，SSCUIA 加密流程圖如圖 3-2 所示。

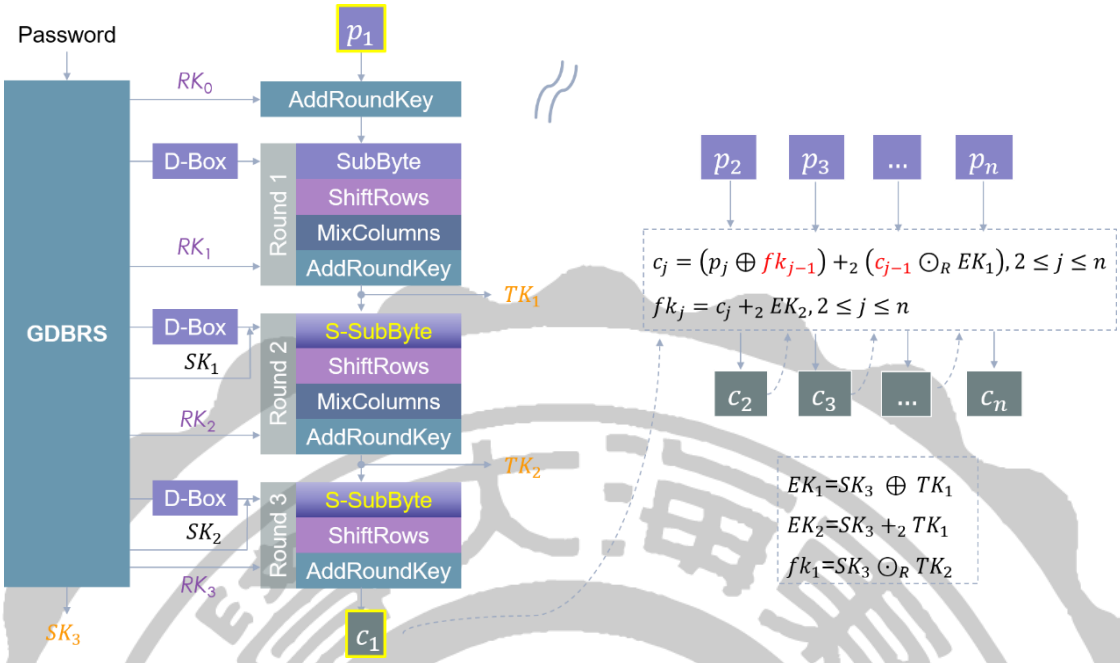

圖3-2分離結構加密法加密流程圖

SSCUIA 的加密由兩個程序構成，即，(1)初始加密程序與(2) 2-to-*n* 區塊加密程序，茲分述如下:

(1) 初始加密程序:

(a) 透過 GDBRS，藉由輸入的 *PW* 產生 D-Box, RK_0 ~ RK_3 與 SK_1~ SK_3，以提供後續加密之用

(b) 透過 IAESA，將第一個明文區塊 P_1 加密產出第一個密文區塊 C_1 與動態加密金鑰 TK_1 與 TK_2，提供 2-to-*n* 加密之用

(2) 2-to-*n* 區塊加密程序:

(a) 藉由初始加密程序產生的 SK_3,TK_1 與 TK_2 進行三維運算分別產出 EK_1,EK_2 與 fk_1，以提供後續加密之用

(b) 藉由 EK_1,EK_2 與 fk_1 與 C_1 對 P_j, $2 \le j \le n$ 進行三維加密運算產生密文區塊 C_j 與回授加密金鑰 fk_j, $2 \le j \le n$

3.6.1 GDBRS for SSCUIA

Algorithm 10:

The Generation of D-Box, Round Keys and Shifting Key for SSCUIA (GDBRS, for short) algorithm /* format is GDBRS(*PW*) */

Input: *PW*

Output: D-Box, Round Keys (RK_0~RK_3) and
Shifting Key (SK_1~SK_3)

1) Call GDBPW(*PW*,S-box) to obtain D-Box, DK_1, DK_2 and DK_3;
2) TDK_1 = DASS($DK_1 \oplus DK_2$, D-Box);

3) TDK_2 = DASS($TDK_1 \oplus DK_3$, D-Box);
4) TDK_3 = DASS($TDK_2 +_2 DK_1$, D-Box);
5) TDK_4 = DASS($TDK_3 +_2 DK_2$, D-Box);
6) $RK_0 = TDK_1 \odot_R TDK_4$
7) For j = 1 to 3
8) $RK_j = (RK_{j-1} \oplus TDK_j) +_2 TDK_{j+1}$
9) $SK_j = (RK_{j-1} +_2 TDK_j) \odot_R TDK_{j+1}$

3.6.2 Encryption for SSCUIA

SSCUIA 的工作原理/加密流程，詳述如下：

Algorithm 10: SSCUIA_Encryption algorithm
Input: Plaintext P, PW
Output: Ciphertext C

1) Let $P=p_1p_2\ldots p_n$, $|p_j|=1024, 1 \leqq j \leqq n$;
2) Let $C=c_1c_2\ldots c_n$, $|c_j|=1024, 1 \leqq j \leqq n$;
3) CALL GDBRS(PW) to obtain D-Box、$RK_0 \sim RK_3$ and $SK_1 \sim SK_3$
4) p_1 =AddRoundKey(p_1, RK_0);
   /*Round 1 */
5) p_1=SubByte(p_1, D-Box);
6) p_1 =ShiftRows(p_1);
7) p_1 =MixColumns(p_1);
8) p_1 =AddRoundKey(p_1, RK_1);
9) $TK_1 = p_1$;
   /*Round 2 */
10) p_1 =S_SubByte(p_1, D-Box, SK_1);
11) p_1 =ShiftRows(p_1);
12) p_1=MixColumns(p_1);
13) p_1=AddRoundKey(p_1, RK_2);
14) $TK_2 = p_1$;
   /*Round 3*/
15) p_1 =S_SubByte(p_1, D-Box,SK_2);
16) p_1 =ShiftRows(p_1);
17) c_1=AddRoundKey(p_1, RK_3);
18) $EK_1=SK_3 \oplus TK_1$;
19) $EK_2=SK_3 +_2 TK_1$;
20) $fk_1=SK_3 \odot_R TK_2$
21) For j = 2 to n
22) $c_j=(p_j \oplus fk_{j-1}) +_2 (c_{j-1} \odot_R EK_1)$

23) $fk_j = c_j \oplus EK_2$

24) Next j;

25) return *C*;

3.6.3 Decryption

SSCUIA 解密原理如下：

Algorithm 10:

The SSUCIA_Decryption algorithm

Input: Ciphertext *C*, *PW*

Output: Plaintext *P*

1) Let $C=c_1c_2\ldots c_n$, $|c_j|=1024, 1 \leqq j \leqq n$;
2) Let $P=p_1p_2\ldots p_n$, $|p_j|=1024, 1 \leqq j \leqq n$;
3) CALL GDBRS(*PW*) to obtain D-Box、$RK_0 \sim RK_3$ and $SK_0 \sim SK_3$;
4) c_1 =AddRoundKey(c_1, RK_3);
   /* Round 1 */
5) c_1 =InvShiftRows(c_1);
6) c_1 =InvSubByte(c_1, Inverse D-Box,SK_2);
7) $TK_2 = c_1$;
8) c_1 =AddRoundKey(c_1, RK_2);
9) c_1 =InvMixColumns(c_1);
   /* Round 2 */
10) c_1 =InvShiftRows(c_1);
11) c_1 =InvS_SubByte(c_1, Inverse D-Box, SK_1);
12) $TK_1=c_1$;
13) c_1=AddRoundKey(c_1, RK_1);
14) c_1 =InvMixColumns(c_1);
   /*Round 3*/
15) c_1=InvShiftRows(c_1);
16) c_1 =InvSubByte(c_1, Inverse D-Box);
17) p_1=AddRoundKey(c_1, RK_0);
18) $EK_1=SK_3 \oplus TK_1$
19) $EK_2=SK_3 +_2 TK_1$
20) $fk_1=SK_3 \odot_R TK_2$
21) For j = 2 to n
22) $p_j = c_j -_2 (c_{j-1} \odot_{IR} EK_1) \oplus fk_{j-1}$
23) $fk_j=c_j \oplus EK_2$
24) Next *j;*
25) return *P*

第四章　安全分析與比較

在本研究中，動態盒(D-BOX)的產生主要是藉由三個動態金鑰，即，DK_1, DK_2 與 DK_3 之內涵值，萃取重排旗標陣列的索引值來構成 D-BOX，故 DK_1, DK_2 與 DK_3 之亂碼品質為何至為重要。此三個動態金鑰主要是由輸入之字串經由資料收縮或擴充運算進行字串轉換改變而得，過程中多次使用 DASS 運算，故 DASS 的安全特性需近一步討論。其次，本研究重新改寫了 AES 的回合金鑰擴充演算法，i.e., GDBRS，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 DASS 之安全分析

DASS 演算法基本上是透過查表方法將字串進行轉換，其首先將長度為 n 位元之輸入字串分割成 k 個字元，並經由每個字元與其伴隨之動態位移量，即 *dsc*，加總後進行查表置換，故，在整個查表過程中，*dsc* 值的變化為 DASS 安全的所在。

dsc 的初始設定值為 256，經由前半段 For 迴圈作用，改寫了 *dsc* 之初始位移值，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，故，當相同字元出現在不同位置會有不同的 *dsc* 增量貢獻。

後半段的 For 迴圈中，循序地產生密文 c_i，而每一個 c_i 是透過第 i 個字元的 ASCII 值加上當時的 *dsc* 值查表而得，在步驟 9 中清楚地顯示出當時 *dsc* 的值除了累加前次 *dsc* 值外，還會受到前次明文字元 ASCII 值($vp[i\text{-}1]$)與當時位置 i 的乘積貢獻影響，可說是具有動態循序累積的作用，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，故與明文字串進行運算的 *dsc* 是動態改變的。基於前述分析，我們可以得出以下結論:

(1) 經由 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) DASS 產出的密文擁有極高的輸入敏感度，這是因為明文的任何細微改變，在前半段的 For 迴圈作用下，這些細微改變會被放大而反應至 *dsc* 之初始值。

(3) 經由動態循序累積作用，使 *dsc* 做不同的循序改變，進而動態的影響整個密文輸出，進而影響首次查表的位置，而在後半段的 For 迴圈中，對於前次明文值的改變或經由刪除/插入而改變明文位置,這些都會造成 dsc 的改變，進而反應輸出的敏銳表現。

由於 DASS 演算法具有優異的不可逆特性與與極佳的雪崩效應，作為一良好 Hash Function 建置的核心功能，而應用於資訊安全相關領域。

4.2 GDBPW 之安全分析

GDBPW 演算法是由輸入之 *PW* 依序執行 GISK(), GDK()與 DBG()等演算法以取得動態金鑰 DK_1, DK_2, DK_3 與 D-Box, 過程中 *PW* 經由金鑰擴展、3D 運算與多次的 DASS()單向混亂置換運算，使得 DK_1, DK_2 與 DK_3 為外界無法預測的亂數金鑰，而 DBG()藉由 DK_1, DK_2 與 DK_3 的內涵值對 FA[]之索引值進行萃取重排，而得到 D-Box，將繼承了對 PW 有極高的輸入敏感度、動態亂度與不可逆的特性，更重要的是，由於動態參數 h_1,h_2 與 jump 的加入而產生的 D-Box,即使此 D-Box 被外界得知，則由 D-Box 要逆向求出 IA2||IAR||IA1 是正確的，進而使得 IA2||IAR||IA1 逆向求出 DK_1, DK_2 與 DK_3 逆向求出 ISK_1,ISK_2 逆向求出 *PW* 為不可能的，亦即，即使 D-Box 為外界得知，亦無法由 D-Box 逆向求出 *PW*，更遑論 D-Box 是外界無法得知的，故 GDBPW 擁有極高的安全性。而透過結合使用 D-Box 的 DASS()運算與 3D 運算而產生的 RK_0~RK_3 與 *SK* 將擁有極高的隱密性與極低的關聯度。

4.3 IAESA 之安全分析

對於 IAESA 在安全上的表現，本研究將由五點，分別為密碼金鑰安全性、置換盒應用之強度、回合金鑰安全性、系統變數總長度以及整體系統複雜度等方向進行分析。

(1). 首先，在密碼金鑰上，原先 AES 採用固定長度的密碼金鑰作為保護，然而，固定的長度意味著該金鑰也將常為較易猜測的固定攻擊目標，相較之下，在 IAESA 中，*PW* 不受 128 位元長度的限制，在我們的實測中，其長度範圍為 8~800000 bits，在如此大範圍可變動長度的保護下，其長度的變化也要成為破密面臨的重要變動因素，因而有效提昇密碼金鑰的安全度。

(2). 在置換盒應用之強度表現中，AES 是固定的置換盒搭配規律的查表加密試，而 IAESA 是由動態置換盒(D-Box)搭配位移金鑰(*SK*)的位移作用進行加密，區塊中的相同明文字元在不同的位置，將會應對到不同的密文字元，另外，由於 D-Box 將隨 *PW* 的不同而不同，攻擊者在不知 *PW* 下亦將不知其內容，使得 SubBytes 運算安全度大大提升而有效增加了破密的困難度。

(3). AES 的回合金鑰主要是藉由已產出回合金鑰間進行⊕運算循序產生，固可透過⊕逆運算，由 RK_{10}, i.e., ($W_{40}W_{41}W_{42}W_{43}$)，反向運算依序取得 RK_9~RK_0，此亦即，AES 的前後回合金鑰有著極高的關聯性，大大降低了安全性；而 IAESA 是由 GDBRS 產出所需的回合金鑰與位移金鑰，這此加密金鑰群間均無固定關係且擁有高的複雜度，其相互關聯度極低，無法由前後回合金鑰或位移金鑰互相關聯求出，自然擁有較高的安全度。

(4). 由於 AES 加密機制中，S-Box 是固定而眾所皆知的，只有 11 個 Round Keys 是需要被破密的，故其系統變數總長度為 128*11=1408 bits 而在 IAESA 加密機制中，需要被破密的加密金鑰計有: RK_0 ~ RK_3 , SK_1~ SK_2 與 D-Box，其中 D-Box 共有 256!個可能，基於 $2^{1683} < 256! < 2^{1684}$，故，我們可得 D-Box 視為一具有 1683 位元長度的變數，由此可得 IAESA 系統變數總長度為 28*4+128*2+1683=2451bits。

(5). 最後，以整體系統之破密複雜度來看時，AES 僅透過單一固定長度金鑰(Cipher Key)與關聯性極高的回合金鑰保護加密系統，而 IAESA 採用幾乎沒有長度限制的密碼金鑰(*PW*)與複雜度高且相互關聯性極低的回合金鑰與位移金鑰保護加密系統，更重要的是，動態改變的 D-Box 完全由密碼金鑰來產生，在不知密碼金鑰的情況下當然不知 D-Box 內容，再加上 S_SubBytes 運算的加入，使得 IAESA 系統安全性將遠優於 AES

表 4.1 將 IAESA 與 AES-128 就不同的安全特性進行比較。

表 4-1 IAESA 與 AES-128 安全特性比較

特性/版本	IAESA	AES-128
密鑰長度	8~800000-bit (勝)	128-bit
置換盒強度	動態盒(D-Box) (勝)	固定盒(S-Box)
回合金鑰安全性	關聯性低，安全度較高 (勝)	關聯性高，安全度較低
系統變數總長度	2451-bit (勝) (128*4+128*2+1683)	1408-bit (128*11)
整體系統複雜度	優	劣

4.4 SSCUIA 之安全分析與比較

結合 IAESA 與分離結構加密機制的 SSCUIA，其中 IAESA 的安全分析如前所述，其安全度是優於 AES-128 的，而分離結構加密法是由初始加密程序與 2-to-*n* 區塊加密程序所構成，我們除了將分別探討其個別安全性外，亦將從整體結構來探討安全性。

4.4.1 初始加密之安全分析

首先，SSCUIA 之內部核心加密參數包括 D-Box、RK_0 ~ RK_3 與 SK_1~ SK_3，其中 D-Box 是經由 PW 透過 GDBPW 產生，在外界不知 PW 的情形下，D-Box 將擁有高度的不可預測性，而內部核心加密金鑰 RK_0 ~ RK_3 與 SK_1~ SK_3 是藉由 GDBPW 產生的三個動態金鑰 DK_1, DK_2 與 DK_3 透過 D-Box、3D 運算與 DASS 的作用而產出，同樣具有極佳的變異性與不可預測特性，使得 D-Box、RK_0 ~ RK_3 與 SK_1~ SK_3 擁有高度安全性。

其次，透過 IAESA 由第一個明文區塊 P_1 而產生的加密動態金鑰 TK_1、TK_2 在外界不知內部核心加密參數情況下，將無法由單筆密文區塊 C_1，逆向反推而得，亦即 TK_1、TK_2 與 P_1 擁有高度的安全性。

4.4.2 2-to-n 區塊加密之安全分析

用於 2-to-n 區塊加密的起始金鑰 EK_1、EK_2 與起始回授金鑰 fk_1，是藉由 SK_3、TK_1 與 TK_2 透過 3D 加密運算而產出，故不為外界所知而擁有高度安全性。而在循序邏輯回授機制下，每個密文區塊 C_j, $2 \leqq j \leqq n$ 的產出都會受到二個動態金鑰，i.e., $(p_j \oplus fk_{j-1})$與$(c_{j-1} \odot_R EK_1)$，的保護，有效增加了破密的困難度。

4.4.3 SSCUIA 整體安全分析

在 SSCUIA 中 P_1 被用來作為，2-to-n 區塊的加密參數，如此將增益加密參數能量，使得在相同的 PW 下，不同的明文將會有不同的加密環境，此將使得明文安全性得以更加提升，下面我們將在同一 PW 下來探討 SSCUIA 之整體安全性。

1. 由於不同的 P_1 將會產生不同的 TK_1、TK_2 與 C_1，故攻擊者在類似選擇明文攻擊中，雖可設定同一 P_1 而有不同的 P_2~ P_n 進行攻擊，然而：
   (a) 即使攻擊者求出 EK_1、EK_2 與 fk_1，亦無法由此逆向求得正確的 TK_1、TK_2 與 SK_3。
   (b) 即使 TK_1、TK_2 與 SK_3 為攻擊者求出，攻擊者亦無法由單筆資料, i.e., (P_1,C_1, TK_1,TK_2,SK_3)逆向求出內部核心加密參數 D-Box、RK_0 ~ RK_3 與 SK_1~ SK_2，亦即 2-to-n 區塊的相關資料將無助於內部核心加密參數的破密。
2. 由於只有(P_1,C_1)的資料才能有助於 PW 的破密，然而由於 IAESA 安全性高於 AES，而每次要由不同的 P_1 產生不同的 C_1 需要執行初始加密程序，這是較費時的，故在合理有限時間下，將不足以產生足夠的(P_j,C_j), $j \geqq 1$ 資料，來進行破密。
3. 在無法由明文/密文資料進行破密以取得 SSCUIA 的內部核心加密參數下，攻擊者或可直接進行猜測 PW 來進行破密，但因 SSCUIA 的 PW 為

8~800000 bits 的變數，不是 128 bit 固定長度變數，這將使直接猜測 *PW* 進行破密為不可能的。

基於前述三項分析證明 SSCUIA 的安全度不但高於 AES，且將達到實際安全的要求。

4.4.4 SSCUIA、IAESA 與 AES 之安全比較

本節中，我們將以下列安全特性，即，加密樣式、P_1 安全性、P_2 to P_n 安全性、回合金鑰安全性、系統變數總長度與整體安全度對 AES-128、IAESA 與 SSCUIA 進行比較，本內容如表 4-2 所示。

特性/版本	SSCUIA	IAESA	AES-128
加密樣式	循序邏輯 (優)	組合邏輯	組合邏輯
P_1 安全性	最佳	次佳	劣
P_2 to P_n 安全性	最佳	佳	劣
回合金鑰安全性	最佳	佳	劣
系統變數總長度	c_1: 8851-bit(優) (1024*4+1024*3+1683) $c_2 \sim c_n$: 1024*3=3072 (優)	$c_1 \sim c_n$: 2451-bit(次佳) (128*4+128*2+1683)	$c_1 \sim c_n$: 1408-bit(劣) (128*11)
整體安全度	最佳	次佳	劣

表 4-2　SSCUIA、IAESA 與 AES 安全特性比較

表 4-2 內容說明如下：

1. 加密樣式：循序邏輯較組合邏輯有較高的安全性。
2. P_1 安全性：SSCUIA 中，P_1 雖是透過 IAESA 機制加密產生密文，然因其長度單位為 1024 位元，高於 IAESA 的 128 位元，故安全性最佳。
3. P_2 to P_n 安全性：SSCUIA 中，每個密文區塊 C_j, $2 \leqq j \leqq n$ 的產出都會受到二個動態金鑰，i.e., $(p_j \oplus fk_{j-1})$與$(c_{j-1} \odot_R EK_1)$，的保護，故，共受到 2(n-1)個動態金鑰的保護，亦或，三個起始加密金鑰(使用暴力法)，即，EK_1, EK_2 與 fk_1 的保護，其保護金鑰總長度為 1024*3=3072 bits，為三個方法中最長的，故有較高的安全性。

4. 在回合金鑰的安全，雖然 SSCUIA 和 IAESA 用的回合金鑰產自 GDBRS，而 AES 用的則是 key expansion，然因金鑰長度的不同，故，SSCUIA 擁有最高安全性。
5. 系統變數總長度：SSCUIA > IAESA > AES-128。
6. 整體安全度：SSCUIA > IAESA > AES-128。

第五章 效能分析與比較

演算法的效能表現是時常被關心的重點，尤其是在於加解密，因此，本章節將會對於 DASS、GDBPW、IAESA 以及 SSCUIA 進行完整的效能分析。並且，為貼近真實用例，本章節的分析將使用不同長度之輸入以模擬現實使用情形。

表 5-1 模擬測試運行環境之規格

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 基本運算子之效能表現

本節將針對基本運算子$\oplus$, $+_2$, $-_2$, $\odot_R$ 與$\odot_{IR}$ 進行效能分析，平均表現之結果採運算 1,250 萬次之平均值作為該運算子的運算耗時。

表 5-2 五種運算子對於常用運算寬度之平均運行時間(單位:奈秒)

Operator \ Key size	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

上述運算除 XOR 外雖皆由軟體實現，然而執行時間仍與 XOR 運算所花費之時間相去不遠，換句話說，假使皆由硬體實現上述之運算子，相信所有運算子在效率上的表現都是相當的。

5.2 DASS 與 GDBPW 之效能表現

DASS 與 GDBPW 皆有良好的使用彈性，兩種演算法皆可針對不受限的輸入資料長度完成預期的目的，然而在 GDBPW 中，對於不同長度的輸入將會改變其演算流程，並且，DASS 也會因其輸入長度的加長而增加其運行時間。

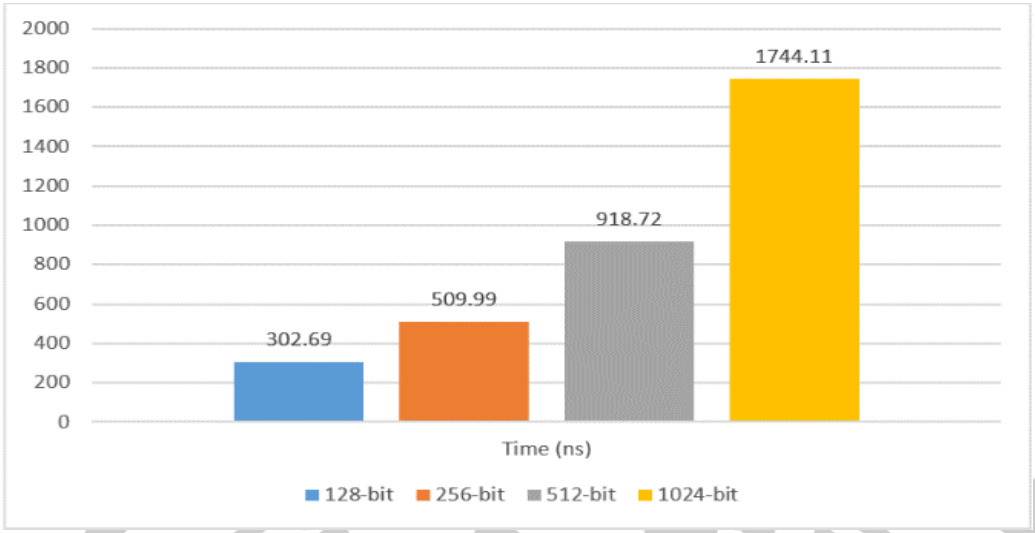

圖5-1 DASS對於不同輸入長度之平均執行時間(1,250萬次平均時間；單位:奈秒)

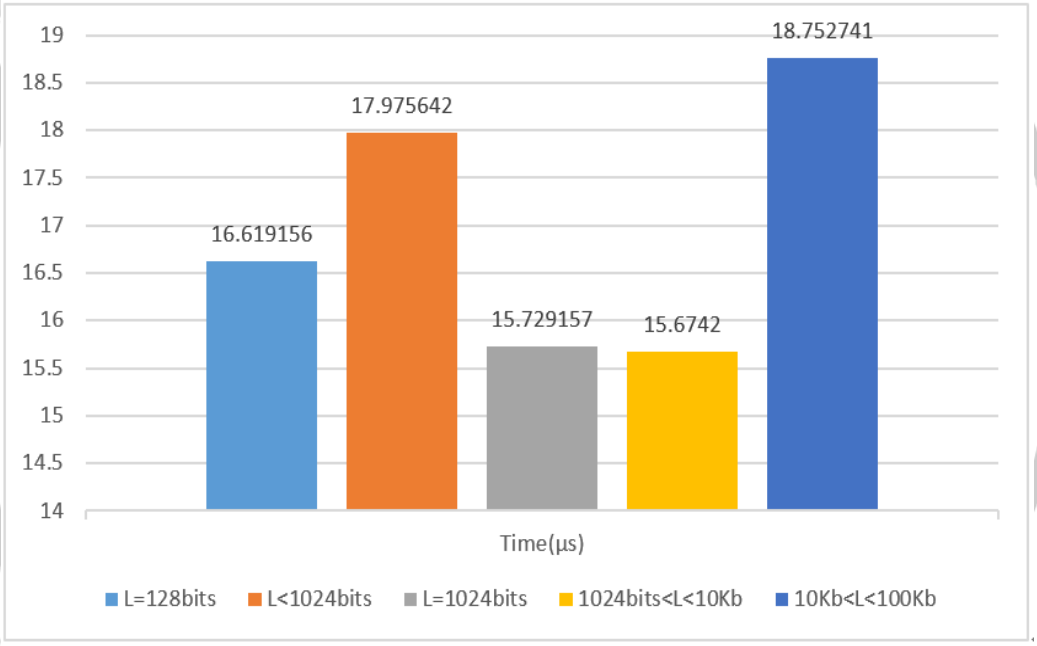

圖5-2 GDBPW對於不同輸入長度之平均執行時間 (200萬次平均時間；L為Input length；單位:奈秒)

如圖 5-1 所示， DASS 的執行時間如預期的成長了，然而其成長特性為良好的線性成長，此外，如圖 5-2 所示，GDBPW 的效能花費也足以滿足大多數的系統並提供高效的環境初始化。

5.3 IAESA 效能分析

在效能方面,經過改良後的IAESA的確比原本的AES有著顯著的改良，並且，為提供更多的安全性，IAESA 中的置換盒生成與金鑰生成雖然比原先更為耗時，然而 IAESA 的加密系統在經過一次的初始化後便可持續提供快速的加解密服務，因此當使用的時間愈久，IAESA 所提供的效能優勢就為顯

著。

5.4 SSCUIA 效能分析

由效能方面，SSCUIA 的融合了原本的 IAESA，有著顯著的改良，並為提供更多的速度，SSCUIA 中的置換盒生成與金鑰結了 IAESA 的 GDBRS 速度上可能較耗時，然而 SSCUIA 的加密系統在經過一次的初始化後便可持續提供快速的加解密服務，且在 2-to-n 區塊加密上加速比 IAESA 快，甚至比原版 AES 加速多，因此當加密資料量越大，SSCUIA 所提供的效能優勢就愈比原版 AES 更加顯著，表 5-3 和 5-4 可以理論評估出整體執行速度的比較。

表 5-3　AES 區塊耗時表(微秒)

	AES
SubBytes	0.186
AddRoundKey	0.192
ShiftRows	0.062
MixColumns	0.291

表 5-4 AES 與 SSCUIA 理論評估執行速度表(微秒)

	AES	SSCUIA
(key expansion +Initial block) (μs)	1.54+0.731*8 =7.388 key expansion+ 10*shifting row+ 10*SubBytes+ 11*AddRoundKey+ 9*MixColumns	27.05+0.21+0.40=27.66 GDBRS+ [3*shifting row+ 1*SubBytes +2* S-SubBytes+ 4*AddRoundKey+ 2*MixColumns]+ [1*($\oplus$) +1*($+_2$) +1*($\odot_R$)] /*the time consuming of EK_1, EK_2 and fk_1 */
p_2~p_n (μs)	0.731*(n-1)*8 [10*shifting row+ 10*SubBytes+ 11*AddRoundKey+ 9*MixColumns]	0.5561*(n-1) [2*($+_2$)+ 1*($\oplus$)+ 1*($\odot_R$)] /*the time consuming of c_j and fk_j */

在本研究中，IAESA 將運算回合數由 10 減至 3，且 p_2~p_n 以三維運算的

方式做加密。在表 5-3 中，區塊長度為 1024 位元，根據理論運算的結果，key expansion +Initial block 的耗時，AES 為 7.388(μs)，SSCUIA 為 27.66(μs)，若 10KB 時，SSCUIA 之平均加/解密速度約為 AES 的 6 倍，當 100MB，則 SSCUIA 之平均加/解密速度約為 AES 的 11 倍，當資料越大，則加倍的倍率越高，計算推估佐證了我們的理論，因此，SSCUIA 在加/解密的效能表現上皆優於 AES。

表 5-5 SSCUIA 與 AES 理論評估執行速度比較表與倍率(微秒)

	SSCUIA	AES	Ratio
10KB	70.48	463.53	6
100KB	460.86	4575.40	10
1MB	4364.68	45694.16	10
10MB	43402.90	456881.66	11
100Mb	433785.10	4568756.66	11

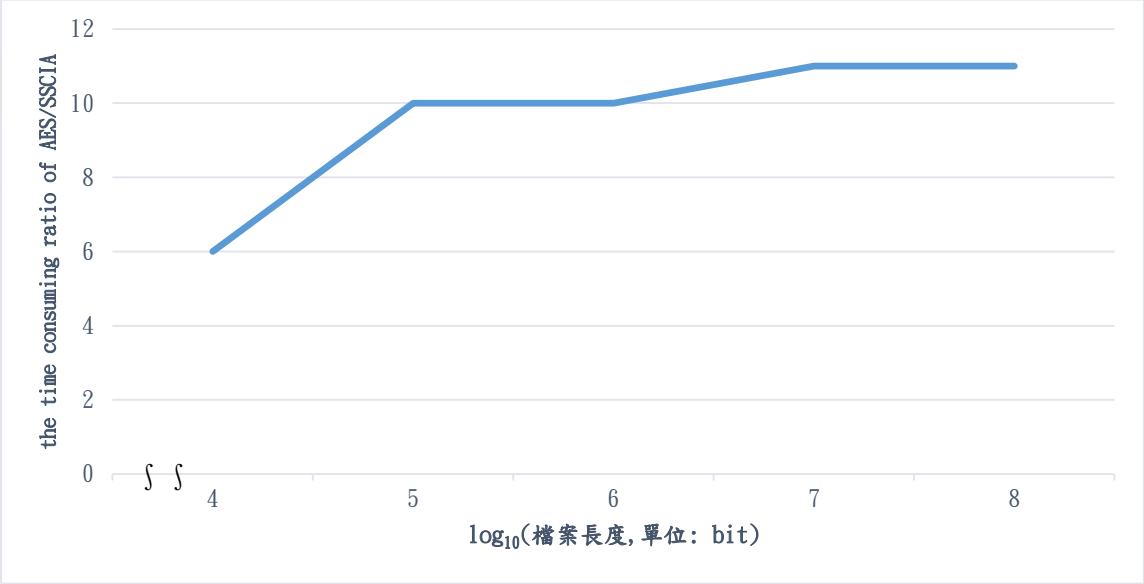

圖5-3 SSCUIA與AES理論評估執行速度相差倍率折線圖

第六章 結論與展望

本研究提出之方法，即，結合 IAESA 與分離結構加密機制的 SSCUIA，經由理論分析比較與效能實測得到下列結論:

(1) 在 GDBPW 演算法中，其 *PW* 長度介於 8 ~ 800000 bits，而 D-Box 的產生不是藉由修改 S-Box 而得到，且因繼承了 DASS 演算法的特性，使得經由 GDBPW 產出的 D-Box 擁有極佳的彈性、極高的亂度、非常優異的輸入敏感度與極佳不可逆等特性。

(2) GDBRS，使用 GDBPW 來產生 D-Box，再用產生的 D-Box 作為查表的依據，故透過三維運算與 DASS 演算法而產生的 Round keys(*RKs*)與 Shifting keys(*SKs*)，不但擁有極高的亂度且 *RKs*、*SKs* 本身之間、*RKs* 與 *SKs* 相互之間的關聯度極低，大大增加了破密的困難度。

(3) 表 4.1 清楚呈現 IASEA 在五項安全特性指標皆比 AES 擁有更佳的安全性，亦即，IAESA 的安全度將高於 AES-128。

(4) SSCUIA 與 AES 由於加密模式的不同，前者為循序邏輯而後者為組合邏輯，使 SSCUIA 的安全性得以提昇，更因分離結構加密法的加入，使得第一個明文區塊(P_1)成為後續 2-to-n 明文區塊的加密參數，如此大大提升了破密的困難度。

(5) 基於 IAESA 優於 AES 的安全性表現，加上分離結構法的貢獻，使得 2-to-n (明文，密文)區塊資料幾乎對破密沒有幫助，使得 SSCUIA 的安全度遠高於 AES，並達到實際安全之要求。

(6) 根據理論評估與實測結果顯示， SSCUIA 其平均加/解密速度為 AES 的 6 倍至 11 倍，當資料越大，其倍率越高。

SSCUIA 之安全度，經由定性分析比較，遠高於 AES 之安全度，然而，如何進一步地定量分析 SSCUIA 之各項安全性，使其能夠更廣泛被接受而推廣使用，實為今後的努力目標。

參考文獻

[1]. FuTURE Mobile Communication Forum, 5G White Paper. http://projects.sigma-orionis.com/choice/wp-content/uploads/2015/01/5G-SIG-white-paper-first-version.pdf, Accessed on November 2015.

[2]. 維基百科編者, “5G,” 維基百科，自由的百科全書, 12 11 2018. [線上]. Available:https://zh.wikipedia.org/wiki/5G?fbclid=IwAR2DxTPGPrgQOBOnzeMS7s81tYn2xeDx9Q-4CVzOp8FYW6J6zCejYSegdgk. [存取日期: 12 11 2018].

[3]. 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 Lecture Notes in Computer Science, French, Riviera, 2010.

[4]. Biryukov and D. Khovratovich, "Related-key cryptanalysis of the full AES-192 and AES-256," in Lecture Notes in Computer Science, Tokyo, Japan, 2009.

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

[6]. ,J. Kim, S. Hong, and B. Preneel, Related-key rectangle attacks on reduced aes-192 and aes-256. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 4593 LNCS:225–241, 2007

[7].http://www.etnet.com.hk/www/tc/news/topic_news_detail.php?category=special&newsid=6164 Accessed on March 29, 2018

[8]. https://5g.co.uk/guides/how-fast-is-5g/ Accessed on April 9, 2018

[9]. http://www.atmarkit.co.jp/ait/articles/1506/18/news019.html, Accessed on June 18, 2016

[10]. Federal Information Processing Standards Publication 197, 2001; Barkan and Biham, 2002.

[11]. O. Dunkelman , N. Keller, and A. Shamir “Improved single-key attacks on 8-round AES-192 and AES-256”, pp. 6-24,2010

[12]. 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.

[13] 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.

[14]. J.J. Liu, Y.L. Huang, F.Y. Leu, X.Y. Pan, and L.R. Chen, “Generating Dynamic Box by using an Input String,” The 2017 International Symposium on Mobile Internet Security, International Convention Center Jeju, Jeju Island, Republic of Korea,

October 19 – 22, 2017. (Accepted, EI)
[15]. 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.
[16]. V. Kaul, V. A. Bharadi, P. Choudhari, D. Shah and S. K. Narayankhedkar, “Security Enhancement for Data Transmission in 3G/4G Networks,” in ICCUBEA, 2015, pp. 95-102.
[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].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,” in ICCST, 2014, no. 7045184.