使用更改及重算策略增加格式保留加密身分證字號的域值大小 (Increasing Domain Size in Format-preserving Encryption for Identification Card Numbers by Modify-Recalculate Strategy)

楊慶隆　劉書宇　許廷綸　謝宗智
國立東華大學 資訊工程學系
cnyang@gms.ndhu.edu.tw

摘要

格式保留加密(Format-preserving Encryption; FPE) 是一有趣且相對較新的技術，具有廣泛的應用範圍。FPE 是密文仍保留明文格式的加密方法，所以適用於數據模式無法更改、或者是軟硬體更新成本過高的系統。FPE 用於保護敏感但未分類(Sensitive But Unclassified; SBU)的資料，例如信用卡、銀行卡、社會安全碼、身份證、護照等號碼、甚至是個人的姓名、出生年月日、電子郵件帳號等 SBU 資料。對於 FPE 加密，秘密數據域值的大小很重要。所謂的域值就是輸入數據的量。小的秘密數據域值會對FPE加密方法有安全威脅。在原始 NIST 的 SP 800-38G 規格中，FPE 加密時的域值大小要求必須至少為 100，建議最好是1,000,000以上。但是現在SP800-38G的修訂版，建議已加強為要求也就是 FPE 的最小域值至少為 1,000,000。本論文使用更改及重算策略增加了FPE加密身分證字號的域值大小。我們的域值符合 NIST 的標準，解決了之前楊等人的格式保留加密身分證域值太小的缺點、提高了加密身分證的安全性。

關鍵詞：格式保留加密、敏感資料、FF1、AES-CBC 模態、身份證字號、更改及重算策略。

Abstract

Format Preserving Encryption (FPE) is an interesting and relatively recent technology with many applications. FPE is an encryption method in which the ciphertext retains the plaintext format, so it is suitable for legacy systems where the data model cannot be changed, or the cost of updating software and hardware is too high. FPE is used to protect sensitive but unclassified (SBU) data, such as credit card numbers, bank card numbers, social security numbers, ID card numbers, passport numbers, and even personal names, dates of birth, email accounts, etc. For FPE encryption, the domain size is important. The so-called domain size is the amount of input data, and small domain size poses a security threat to FPE encryption. In the NIST SP 800-38G specification, the domain size of FPE encryption is required to be at least 100, and it is recommended to be above 1,000,000. But now the revised version of SP 800-38G is recommended to be strengthened as a requirement (the minimum domain size of FPE should be at least 1,000,000). In this work, we use a modify-and-recalculate strategy to increase the domain size when using FPE to encrypt ID numbers. Our domain size is compliant with NIST standards. It solves the weakness of Yang et al.'s FPE encryption of ID cards, which has a small domain size, and enhances the security

Keywords: Format-preserving Encryption (FPE), Sensitive data, FF1, AES-CBC mode, Identification card number, Modify-and-recalculate.

1. 前言

格式保留加密 (Format-preserving Encryption; FPE) 的特性就是加密後的密文格式與加密前的明文格式完全相同。例如信用卡號是採用 ISO/IEC 7812編碼標準為16 位數字，以 FPE 加密後還是16位數字卡號。FPE 加密敏感但未分類的 SBU (Sensitive But Unclassified) 資料。一般 SBU 資料如信用卡號、社會安全碼、身份證字號、護照號碼、還有一般個資(姓名、出生年月日、PIN 號碼、電子郵件帳號)等。參考文獻 [1-6] 就是針對這些資料所提的 FPE 加密方法。FPE 好處是保留原始數據的格式和長度，這兩種特性可使用於資料格式特定的舊系統所以不需常更新代碼與系統[7]。最近更有人利用FPE加密後保留格式的優點提出用於網路層進行隱私保護[8]，以 FPE 加密混淆來源 IP 位址、網路連接埠、並保護網路封包標頭欄位。另外，使用 FPE 的一個好處是加密後這些資料不會受到額外的關注。傳統加密產生的隨機雜亂資料很容易察覺，但FPE加密並不容易發現資料是否被加密可減少對資料被惡意破壞。

從上面的敘述， FPE 確實是一有趣且有廣泛應用的加密技術。最近在參考文獻 [9]，楊等人設計了以 FF1格式保留加密中華民國國民身份證字號的機制，但其使用的域值 (輸入數據的量) 太小。小域值不能確保 FPE 加密的安全性。NIST 的 SP 800-38G 修訂版規格[10]要求 FPE 的域值至少為10^6。楊等人的加密機制並不符合安全規定。本論文使用更改及重算策略將FPE加密身分證字號的域值增加至10^9符合 NIST 標準(>10^6)，解決了楊等人加密機制域值太小的缺點、提高FPE加密身分證的安全性。本文結構如下，第二部分為文獻探討，介紹了身分證字號產生規則、FPE 加密演算法、及楊等人的 FF1身分證字號加密機制。第三部是我們所提的增加 FF1身分證字號加密域值的方法，包含了研究動機、更改及重算以增加域值大小的概念、及域值為10^9的 FF1身分證字號加/解密實作。第四部分為實作結果，最後則為結論。實作的程式碼也包含在附錄中。

2. 文獻探討

2.1 國民身分證字號產生規則

中華民國國民身份證字號由10碼組成，第一碼為大寫英文字母，其餘九碼為阿拉伯數字。第一碼的大寫英文字母代表國民第一次所報的戶籍地區，二十六個英文字母中有些字母因為時代的演變目前已經沒有再使用。例如，1974年以後陽明山管理局虛位化，所以原先使用的Y已停止發放。而2010年以後台中縣、台南縣與高雄縣因縣市合併為直轄市的緣故， L、R、S 也停止發放。所以現今還有繼續在發放的字母只有22個。第二是用來區分性別1代表男性2代表女性，在110年後增加了8跟9分別代境外移入的男性與女性(因為佔比數極為少數，本論文加密機制僅考慮1與2)。第三碼原本是流水號的一部份但在2003年後，分別使用6、7、8、9來代表在台灣取得戶籍的外國人。第四到第九碼為流水號。第十碼則為檢查碼，其產生規則簡述如下。將第一個英文字母依照「國民身分證字號英文碼與縣市及轉換數值表」轉換為數字：轉換後的身份證字號共11位、其權重從左到右分別為 (1, 9, 8, 7, 6, 5, 4, 3, 2, 1, 1)。將每位數字與其對應的權重相乘，相加的結果若是10的倍數則身份證字號正確。

2.2 格式保留加密演算法 FPE

FPE 格式保留加密顧名思義就是密文與明文有相同格式的加密技術 [11]。一般而言，FPE 需符合 (i)明文與密文具有相同的格式與長度、(ii) 適用於各種的數字與符號、(iii) 適用於多種不同的明文長度、(iv) 明文長度短依然安全。所以 FPE 不直接使用區塊狀加密來進行加密，而是使用區塊加密技術來產生出隨機字串，並使用這個字串來與明文進行運算以完成加密。FPE 採用 Feistel 結構，先將明文分為 A_0 與 B_0 兩部分。對第 i 個 Round，將 B_i 經過 $F_K(\cdot)$ 函式後產生的字串 $F_K(B_i)$ 與 A_i 做運算得到 B_{i+1} 、而 B_i 指定給 A_{i+1} ，其中 $0 \le i \le (r-1)$ 。完成 r 個 Round (Round 0 ~ Round r) 操作後產生同樣長度的密文。FPE 解密是使用反向運算即可解回明文。FPE 使用了可變的 Tweak 參數增強加密的安全性，相同的明文也因為使用不同的 Tweak 參會有不同的密文。

NIST 設計了三種 FPE 加密方法: FF1、FF2 和 FF3，廣義來說都可歸類成 AES 加密模態。FF1 和 FF2兩種演算法有10 個 Round，而 FF3 只使用 8 個 Round。另外，FF2的每個 Round 使用主密鑰所產生的子密鑰加密、而 FF1和 FF3的每個 Round 則直接使用主密鑰加密。還有，FF1的每個 Round 使用 CBC 加密模態、而 FF2和 FF3的每個 Round 是使用簡單的 ECB 加密模態。

2.3 楊等人的 FF1身分證字號加密機制

因為 FF3已遭到密碼分析攻擊[12-14]、有安全漏洞，NIST 已修改 SP 800-38G FF3為 FF3-1 (SP800-38G Rev1) [10]。楊等人的身分證字號加密[9]是使用 FF1，將身分證字號分成兩個部分。第一部分是結合第一碼及第二碼映對至7^2 (radix 為7的2個碼字)、第二部分是第三碼到九碼流水號10^7 (radix 為10的7個碼字)。這兩部份分別使用兩個 FF1來處理。加密的身份證字號檢查碼是由加密後的第一、二部分依據檢查碼規則產生。第一部分7^2的映對設計是刪掉代表陽明山管理局的字母 Y。因為自從1974年後就已經停止發放至今已有50年，現在陽明山管理局的現行行政區是台北市。自從1974年停發後至今已有50年，所以可合理將 Y 去掉。這樣第一碼的基數為25和基數為2的第二碼共有50種組合。選擇使用7^2 (49種組合)來映對第這50種組合。多出來的一個組合，楊等人的機制是把台南縣的 R1和 R2做合併處理，以達到只有49種組合。因縣市合併，所以原先所使用的字母 L、R、S 已停發，而台中縣、台南縣、高雄縣中，又屬台南縣(R)的人口數最少。考慮處理的數量問題，R1/R2編成同一碼。利用身分證字號儲存索引位置的奇、偶數，解碼後分辨是 R1、或 R2。此索引值在楊等人的加密機制也被當作加密的 Tweak 參數。

3. FF1 身分證字號加密增加域值的方法

3.1 研究動機

在原始 SP 800-38G 規格中，FF1 和 FF3 加密時的域值大小要求必須至少為 100，建議最好是1,000,000以上。現在修訂的 SP800-38G Rev1[10]，建議已加強為一項要求，也就是 FF1 和 FF3-1的最小域值大小要求至少為 1,000,000。楊等人的身分證字號加密機制以兩個 FF1加密時，第二部分的域值10^7符合規定、但是第一部分的域值7^2太小並不符合安全規定。為了提高格式保留身份證字號加密的安全性，本文提出適用於中華民國國民身份證字號的 FF1加密方法，其域值為10^9符合 SP800-38G Rev1的規定需至少為1,000,000，且加密後依然有著合法的國民身份證字號格式。

3.2 更改及重算以增加域值大小的概念

我們將第一碼刪掉代表陽明山管理局的字母 Y (此刪除步驟與楊等人的身分證字號加密機制類似)，然後依序將 A1、B1、…、X1、Z1、A2、B2、…、X2、Z2編碼成0~49的50種組合，使用 FF1加密時我們的更改及重算策略有時會將此數值 +50時得到50~99。最後，此一數值是兩碼的十進位數字。將前兩碼的十進位數字及後七碼(不包括

檢查碼)視為同一域值。此域值10^9符合 NIST 對於 FF1加密最小域值的規定值10^6。

圖1是大域值格式保留身分證字號加密的更改及重算概念。首先是以 FF1對9個十進位數字做格式保留加密，其中前兩碼是圖1中的①原本的編碼數值[0, 49]。一般而言，FF1加密後的密文前兩碼可能會超過50。若是前兩個十進位數字小於50 (0~49)，則能轉換回 A1、B1、…、X2、Z2的格式，然後停止。若是前兩個位數大於等於50 (50~99)，則進入更改及重算策略如圖1的②所示，將原本的數值+50時得到50~99、再次以 FF1做格式保留加密。若加密後的前兩個位數小於50 停止，若還是大於等於50則更改身分證儲存欄位再執行①、②。因為我們使用儲存的索引位置當作 FF1加密時的 Tweak 參數，所以更換欄位後可以再重算①、②檢查其重算值是否在 [0, 49]之間。此更改及重算策略可以持續直至重算值在 [0, 49]，能轉換回 A1~Z2以達到格式保留。若前兩碼是"+50"，解密後也可解會正確的 A1~Z2。因為我們知道正確的前兩碼只有50種組合[0, 49]，所以若大於等於50就"–50"即可解回。

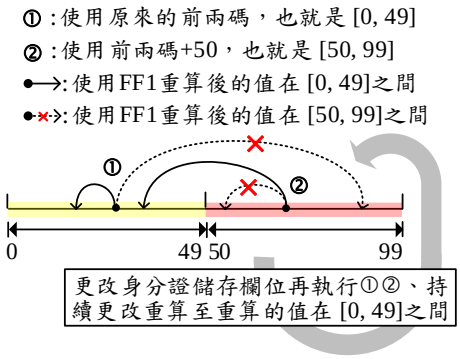

圖 1格式保留身分證字號加密的更改及重算

Eq. (1)呈現了這個更改及重算策略的概念。以身分證字號 T104453782為例，T1轉換成"19"然後以 FF1對9個十進位數字"190445378"做格式保留加密，結果是740400878。因為74≥50，所以更改19+50=69 再次對"690445378"加密，結果是690995383。因為69≥50，所以接下來就只能更換欄位對"190445378"、或"690445378"加密，值至密文前兩碼小於50。以這個身分證加密為例，更換欄位對"190445378"加密得到245995933。將"24"轉為 Z1並計算得檢查碼是"2"，最後符合格式保留加密後的身分證字號是 Z159959332。

$$
\begin{cases}
\underbrace{\text{T}}_{\text{radix 25 (刪除Y)}}\ \underbrace{1}_{\text{radix 2 (性別)}}\ \underbrace{0445378}_{\text{radix 10 (流水號)}}\ \underbrace{2}_{\text{檢查碼}} \Rightarrow \underbrace{19}_{\text{前兩碼}}\ \underbrace{0445378}_{\text{後七碼}} \xrightarrow{\text{FF1}} \underbrace{74}_{\text{前兩碼}}\ \underbrace{0400878}_{\text{後七碼}} \\
\underbrace{69}_{\text{前兩碼}}\ \underbrace{0445378}_{\text{後七碼}} \xrightarrow{\text{FF1}} \underbrace{69}_{\text{前兩碼}}\ \underbrace{0995383}_{\text{後七碼}} (\because 74 \geq 50,\ 19 \rightarrow 69) \\
\underbrace{19}_{\text{前兩碼}}\ \underbrace{0445378}_{\text{後七碼}} \xrightarrow{\text{FF1}} \underbrace{24}_{\text{前兩碼}}\ \underbrace{5995933}_{\text{後七碼}} (\because 69 \geq 50 \text{更改欄位重算}) \\
\Rightarrow \underbrace{\text{Z1}}_{\text{前兩碼}}\ \underbrace{5995933}_{\text{後七碼}}\ \underbrace{2}_{\text{檢查碼}}\ (\because 24 < 50, \text{得到前兩碼並計算檢查碼})
\end{cases}
\tag{1}
$$

我們的更改及重算策略效率是非常高的，每個欄位有兩次 FF1的加密運算可達到格式保留。兩次都不成功的機率是1/4。更改 n 個欄位成功得到格式保留加密身分證字號的機率是 $p=1-(1/4)^n$，p = 75%、93.75%、98.44%、99.61%當 n=1~4時。因此使用更改及重算可增加域值、並有效完成加密和儲存。圖2是在 Excel 檔加密6筆身分證資料的例子: P297989542、M135513977、C129288054、C116918909、A135638059、T104453782。前5筆在同一欄位就成功 (註:第1、4筆第一次加密、其他第二次加密)，第6筆就是 Eq. (1)的例子，更改到第7欄位才成功而第6欄位可預留給其他身分證。最後加密後的身分證字號是: Q202489596、R185569478、C184338600、P127203599、A190188605、Z159959332。

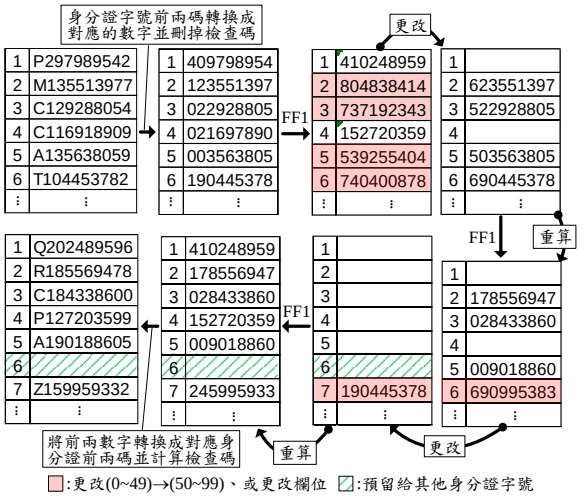

圖 2 域值為10^9的格式保留身分證字號加密例子

3.3 域值為10^9的 FF1身分證字號加/解密實作

演算法1、演算法2分別是域值為10^9的 FF1身分證字號加、解密演算法。描述演算法前，我們先定義一些符號。

表 1符號定義

符號	說明
X[1…10]	原本身分證字號
Y[1…10]	加密後的身分證字號
M(·)	M(·)函式是身分證前兩碼"A1~Z2" 50種組合依序映射至2個十進位數 [0, 49]的函式:例如 M(A1)=00、M(B1)=01、…、M(Z2)=49
$M^{-1}(\cdot)$	$M^{-1}(\cdot)$是 M(·)的反函式，但數值 m 和$(m+50)$有相同的解碼結果，其中$0\le m\le 49$。例如 $M^{-1}(00)=M^{-1}(50)$=A1、…、$M^{-1}(49)=M^{-1}(99)$=Z2
T	儲存索引位置用於 FF1和 $FF1^{-1}$的 Tweak 參數
FF1(·) / $FF1^{-1}(\cdot)$ I[1…9]/O[1…9]	FF1、$FF1^{-}$是 FPE 加、解密程式，其域值為10^9輸入輸出各為 I[1…9]、O[1…9]兩個9位數的十進位數字，也就是 FF1(I, T)=O、$FF1^{-1}$ (O, T)=I
P(·)	計算檢查碼的函式例如 X[10]=P(X[1…9])

圖3的演算法1是加密演算法，輸入是身分證字號X[1…10]、及其所儲存資料庫的索引位置(註: 此索引值我們會用於格式保留加、解密程式FF1和$FF1^{-1}$當作Tweak參數)。首先，使用映射函式將身分證字號前兩位X[1, 2]的"A1~Z2"50種組合轉換成對應的2個位數十進位數字[0, 49]。轉換完的X[1, 2]及X[3…9]當作FF1的輸入I[1…9]加密得O[1…9]= FF1(I[1…9])。若O[1, 2]值小於50時前兩碼就符合格式保留、加密就停止。如果O[1, 2]值大於等於50時則為無效加密，就進入更改及重算策略。將"I[1, 2]+50"後重算O[1…9]=FF1(I[1…9])，若O[1, 2]小於50時加密就停止。若此時O[1, 2]的值還是大於等於50，就把現在欄位更改至沒使用的欄位(註: 移出後的第欄位可預留給其他身分證字號)、並依據此儲存索引位置更動Tweak參數T和回復原始的I[1, 2]重複以上的步驟直至O[1, 2]值小於50。接下來，只需要將O[1, 2]轉換成Y[1, 2]=M^{-1}(O[1, 2])=A1~Z2、Y[3…9]=O[3…9]，並由檢查碼函式算得Y[10]= P(Y[1…9])。最後得到符合格式保留的身分證字號Y[1…10]。

【演算法1】: 域值為10^9的FF1身分證字號加密演算法
輸入：身分證字號X[1...10]、Tweak參數T
/* 參數 T 是儲存索引位置 */
輸出：加密後的身分證字號Y[1...10]
/* Y[1...0]需符合台灣身分證字號規則 */
步驟：
(1) I[1, 2]←M(X[1, 2]);
/* 將身分證字號前兩碼X[1, 2]的50種組合(A1~Z2)轉換成對應的2個位數十進位數字[0, 49] */
(2) I[3...9]←X[3...9];
(3) 對每一身分證字號做步驟**(3)**; /* 更改及重算策略 */
(3-1) O=FF1(I, T);
(3-2) if {O[1, 2] < 50} go to **(4)**;
(3-3) else { I[1, 2]=I[1, 2]+50;
O=FF1(I, T);
if O[1, 2] < 50 go to **(4)**;
}
(3-4) 把現在欄位更改至沒使用的欄位、並依據此儲存索引位置更新Tweak參數T; /*移出後的欄位可預留給其他身分證號 */
I[1, 2]=I[1, 2]− 50; /*回復原始的I[1, 2] */
go to **(3-1)**;
(4) Y[1, 2]=M^{-1}(O[1, 2]); /* 將數值[0, 49]映射至A1~Z2 */
Y[3...9]←O[3...9];
Y[10]←P(Y[1...9]); /* 計算檢查碼 */
輸出加密後的身分證字號Y[1...10];

圖 3 域值為10^9的 FF1身分證字號加密演算法

圖4的演算法2是解密演算法，過程與演算法1類似但是較簡單不需要"更改及重算策略"，這是因為解密後的前兩個十進位數字即使是大於50也能順利解回。因為 M^{-1}(·)反函式對*m*和(*m*+50)兩個值有相同的解碼結果，例如40和90，我們知道90是加密時在"更改及重算策略"將40+50得到90，所以都能解回M^{-1}(40)=M^{-1}(90)=P2。首先將Y[1…9]轉換成O[1…9]，其中O[1, 2]=M(Y[1, 2])、O[3…9]= Y[3…9])。由FF1解密函式得到$FF1^{-1}$(O[1…9])= I[1…9]。X[1, 2]=M^{-1}(I[1, 2])可將數值[0, 99]映射至A1~Z2，其中M^{-1}(·)函式對輸入50~99的解碼與0~49一樣。接下來X[3…9]=I[3…9]，並由檢查碼函式算得X[10]= P(X[1…9])。最後解回原來的身分證字號X[1…10]。

【演算法2】: 域值為10^9的FF1身分證字號解密演算法
輸入：加密後的身分證字號Y[1...10]、Tweak參數T
/* 參數 T 是儲存索引位置 */
輸出：解密後的身分證字號X[1...10]
步驟：
(1) O[1, 2]←M(Y[1, 2]);
/* 將身分證字號前兩碼Y[1, 2]的50種組合(A1~Z2)轉換成對應的2個位數十進位數字[0, 49] */
(2) O[3...9]←Y[3...9];
(3) 對每一O[1...9]做步驟**(3)**;
(3-1) I=$FF1^{-1}$(O, T);
(3-2) X[1, 2]=M^{-1}(I[1, 2]); /* 將數值[0, 99]映射至A1~Z2，註: M^{-1}(·) 函式對輸入50~99的解碼與0~49一樣 */
(3-3) X[3...9]←I[3...9];
(3-4) X[10]←P(X[1...9]); /* 計算檢查碼 */
(3-5) 輸出解密後的身分證字號X[1...10];

圖 4 域值為10^9的 FF1身分證字號解密演算法

4. 實作結果

本次實驗的資料庫使用 Excel，我們先試做10筆身分證測試資料。圖5是域值為10^9的 FF1身分證資料加密測試，呈現的欄位依序是 X[1…10]、I[1…9]、O[1…9]、Y[1…9]、和加了檢查碼 Y[10]的 Y[1…10]。若 I[1, 2] <50 (或是≥50)則表示在此欄位是做一次、或兩次才得到格式保留。圖5的 O[1, 2]都是小於50 (註:若第一次不成功就使用更改及重算策略達到符合格式保留、也就是O[1, 2]都是小於50才能轉換到 A1~Z2)。簡單起見，我們使用 1~10作為這 10 筆記錄的 Tweak 數值。

這10筆測試資料的前6筆就是圖 2的身分證字號加密例子。前5筆身分證資料以 I[1, 2] (<50)加密後，第1、4筆其 O[1, 2] 就(<50)而第2、3、5筆 O[1, 2]≥50，所以需使用更改及重算策略。最後再次檢查其 O[1, 2] 小於50，加密停止。例如以第一筆身分證資料 P297989542為例， I[1…9]=(409798954)因為 M(P2) =40。FF1(409798954)=(410248959)。此時，O[1, 2] =41 (<50)所以加密停止。因為 M^{-1}(41)=Q2，所以 Y[1…9]=Q20248959。使用計算檢查碼函式得到 Y[10]=6，最後格式保留的加密身分證字號是 Y[1…10]=(Q202489596)。最後， 10筆測試資料的測試結果是:第1、4、8、9筆資料在同欄位第一次就達到 O[1, 2]<50(以實線方形虛所框)， 第2、3、5、10筆資料在同欄位第二次達到 O[1, 2]<50(以虛線方形所框)。第6筆資料與第7筆資料互換欄位，第6筆資料第一次就 O[1, 2]<50、而第7筆資料在第六欄位第二次才 O[1, 2]<50。

圖6是圖5加密後身分證字號 Y[1…10]的解密結果，呈現的欄位依序是 Y[1…10]、O[1…9]、I[1…9]、X[1…9]、和 X[1…10]。解密比加密簡單，這是因為使用更改及重算策略的加密結果一定小於50、另外 M^{-1}(·)函式對輸入50~99的解碼與0~49一樣，它可以解碼 I[1, 2]=0~99。圖6確實呈現了正確解密結果，解回原10筆身分證測試資料。

A	B	C	D	E	F
	X[1...10]	I[1...9]	O[1...9]	Y[1...9]	Y[1...10]
1	P297989542	409798954	410248959	Q20248959	Q202489596
2	M135513977	623551397	178556947	R18556947	R185569478
3	C129288054	522928805	028433860	C18433860	C184338600
4	C116918909	021691890	152720359	P12720359	P127203599
5	A135638059	503563805	009018860	A19018860	A190188605
6	Z226239238	992623923	048179478	E18179478	E181794788
7	T104453782	190445378	245995933	Z15995933	Z159959332
8	W228564386	472856438	028356993	C18356993	C183569932
9	A227572428	752757242	303262747	F23262747	F232627471
10	F216712162	801671216	307176271	F27176271	F271762713

圖 5 域值為10^9的 FF1身分證資料加密測試

H	I	J	K	L	M
	Y[1...10]	O[1...9]	I[1...9]	X[1...9]	X[1...10]
1	Q202489596	410248959	409798954	P29798954	P297989542
2	R185569478	178556947	623551397	M13551397	M135513977
3	C184338600	028433860	522928805	C12928805	C129288054
4	P127203599	152720359	021691890	C11691890	C116918909
5	A190188605	009018860	503563805	A13563805	A135638059
6	E181794788	048179478	992623923	Z22623923	Z226239238
7	Z159959332	245995933	190445378	T10445378	T104453782
8	C183569932	028356993	472856438	W22856438	W228564386
9	F232627471	303262747	752757242	A22757242	A227572428
10	F271762713	307176271	801671216	F21671216	F216712162

圖 6 域值為10^9的 FF1身分證資料解密測試

5. 結論

FPE 加密的域值大小是輸入數據的量。小的域值對 FPE 加密有安全威脅，參考文獻[13]就是當域值小於2^9的有效攻擊方法。依據 NIST“SP 800-38G”的修訂版， FF1加密的最小域值需為10^6。本論文使用更改及重算策略增加域值大小。我們的域值為10^9符合 NIST 的標準，解決了楊等人的 FF1格式保留加密身分證域值太小的缺點。更改及重算策略有極高的效率。每欄位有兩次 FF1加密運算可達格式保留，當更動4次欄位就有$1-(1/4)^4=99.61\%$。國民身分證已開始第二碼增加8跟9代表境外移入的男、女性，未來我們工作著重於研究加入第二碼的新狀況、並維持規定域值的身分證加密。另外，也會使用 NIST 其它他格式保留加密例如 FF3-1、或 FF2，比較這三種 FPE 技術在加密身分證字號的效能。

致謝

本研究部分成果由國科會計畫編號 NSTC 112-2221-E-259-007-MY2 補助，特此致謝。

附錄

附錄詳列了實作程式碼及更詳細的實驗模擬。圖7是實作域值為10^9的 FF1格式保留身份證字號加密程式碼 (解密程式碼類似加密所以省略)。其中 FF1中的 AES-CBC 模態，是以 PyCryptodome 程式庫完成。圖8、(a)和(b)則是更詳細的加、解密實驗模擬。Tweak 參數值是由921~945。圖8、(a)紅色線所框之處是更動欄位一次(或一次以上)才達到格式保留。例如，欄位923的 I[1, 2]是55它是由5+50而得，所以是在這個欄位試第二次才成功。以欄位931、及945為例，它的 I[1, 2]分別是26 (<50)及47 (≥50) ，但是不需更換欄位即得到格式保留。圖8、(b)的解密過程是不需要“更改及重算策略”，所以沒有紅色線所框之處。

```
import math
import pandas as pd
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
from Crypto.Random import get_random_bytes

# 讀取 16bytes 的密鑰
f = open('key_for_16_bit.txt')
key = f.read()
f.close()

def CIPH(key, X):
    """
    # 輸入: key: 加密密鑰，類型為bytes，長度應為16位元組。
    # X: 要加密的資料，類型為bytes，長度應為16位元組。
    # 使用 PyCtyptodome 程式庫實作、選擇 CBC模態
    cipher = AES.new(key, AES.MODE_CBC)
    iv = cipher.iv
    ciphertext = cipher.encrypt(X)
    return (iv, ciphertext)
def PRF(key, X):
        m = int(len(X) / 16) # 計算X可以分成多少個16位元組的塊
    zero = 0
    Y_0 = zero.to_bytes(16, 'big') # 初始化Y為16位元組的0值
    X_list = [X[i*16 : (i+1)*16] for i in range(m)] # 將X分解為多個16位元組
for i in range(m):
        if i == 0:
            Y_0 = int.from_bytes(Y_0, 'big')  # 將Y從 bytes 轉換為整數
            X = int.from_bytes(X_list[i], 'big')  # 將當前塊X從 bytes 轉換為整數
            data = Y_0 ^ X  # 對Y和X進行XOR運算
            data = data.to_bytes(16, 'big')  # 將結果由整數轉換回 bytes
            Y = CIPH(key, data)  # 使用密鑰對數據進行加密
        Y = int.from_bytes(Y, 'big')  # 將Y從 bytes 轉換為整數
        X = int.from_bytes(X_list[i], 'big')  # 將當前塊X從 bytes 轉換為整數
        data = Y ^ X  # 與前一密文塊進行XOR運算
        data = data.to_bytes(16, 'big')  # 將結果轉換回 bytes
        Y = CIPH(key, data)
    return Y
def NUM_radix(n, radix):
    #  將數字n轉換成給定基數radix的字串表示形式。
     if n == 0:
        return 0
    a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 'A', 'B', 'C', 'D', 'E', 'F']
    b = []
    while n > 0:
        b.append(a[n % radix])
        n = n //
    b.reverse()  # 反轉列表以得到正確的順序
    return int(''.join(map(str, b)))
def str_radix_m(n, m, radix):
    #  給定一個數字n和長度m，將數字轉換為基數radix的字串確保長度為m
    X = [0] * m
    for i in range(m - 1, -1, -1):
        X[i] = n % radix  # 獲取當前基數的餘數
        n = n // radix  # 更新數字為商
    return ''.join(map(str, X))
def FF1(plaintext, radix, key, tweak):
    # 輸入: plaintext: 待加密的文本字串、radix: 基數，用於確定加密的基數
    #key: 加密密鑰，類型為bytes長度16位元組、tweak參數: 類型為bytes。
    n = len(plaintext)  # n為plaintext字串的字元長度
    u = math.floor(n / 2)  # 回傳小於等於x的最大整數
    v = n - u
    A = plaintext[:u]
    B = plaintext[u:]
    b = math.ceil((math.ceil(math.log2(radix) * v)) / 8)
    # math.ceil(x)回傳小於等於x的最大整數
    d = 4 * math.ceil(b / 4) + 4
    tweak = tweak.to_bytes(2, 'big')
    t = len(tweak)
    P = bytes([1, 2, 1]) + radix.to_bytes(3, 'big') + bytes([10, u%256]) +
      n.to_bytes(4, 'big') + t.to_bytes(4, 'big')
    # P = [1]^1 || [2]^1 || [1]^1 ||[radix]^3 || [10]^1 || [u mod 256]^1 || [n]^4 || [t]^4
```

```
for i in range(10):
    # Q = tweak || [0]^(-t-b-1) mod 16 || [i]^1 ||[NUM_radix(B)]^b
    Q = tweak + (0).to_bytes((-t - b - 1) % 16, 'big') + i.to_bytes(1, 'big') +
    NUM_radix(int(B), radix).to_bytes(b, 'big')
    R = PRF(key, P + Q)
    y = int.from_bytes(R[1], 'big')
    y = NUM_radix(y, 2)
    if i % 2 == 0:
        m = u
    else:
        m = v
    c = (NUM_radix(int(A), radix) + y) % radix ** m #c轉換為長度為m的字串
    C = str_radix_m(c, m, radix)
    A = B
    B = C
return str(A) + str(B)
```

圖 7 域值為10^9的 FF1身份證字號加密程式碼

	X[1...10]	I[1...9]	O[1...9]	Y[1...9]	Y[1...10]
921	G265044536	816504453	367059958	L27059958	L270599587
922	K217345436	351734543	352284548	K22284548	K222845489
923	F149100465	554910046	054965046	F14965046	F149650460
924	A231193800	253119380	258669435	A28669435	A286694352
925	I174577631	587457763	142963318	O12963318	O129633185
926	A108520948	000852094	006352149	A16352149	A163521498
927	U264509673	456450967	011956522	B11956522	B119565221
928	C269371567	776937156	282442211	D22442211	D224422111
929	H229937951	822993795	373449300	M23449300	M234493003
930	J236147520	843614752	398620302	O28620302	O286203023
931	B294314542	269431454	274931509	C24931509	C249315090
932	T200987994	440098799	490099299	Z20099299	Z200992990
933	X166008810	736600881	237105886	X17105886	X171058868
934	G118341893	061834189	062334194	G12334194	G123341949
935	F220445165	302044516	357045066	K27045066	K270450665
936	E100027462	040002746	090003246	J10003246	J100032467
937	P226454496	402645449	402695449	P22695449	P226954491
938	G236058615	313605861	318605911	G28605911	G286059117
939	E134807965	043480796	098481346	J18481346	J184813462
940	C244232621	774423262	279928317	C29928317	C299283178
941	E284156862	798415686	303420736	F23420736	F234207360
942	E207759496	290775949	295775999	E25775999	E257759991
943	N208461223	380846122	431396627	S21396627	S213966275
944	D185017898	038501789	088552289	I18552289	I185522893
945	W277458062	477745806	483295861	X23295861	X232958616

(a)

	Y[1...10]	O[1...9]	I[1...9]	X[1...9]	X[1...10]
921	L270599587	367059958	816504453	G26504453	G265044536
922	K222845489	352284548	351734543	K21734543	K217345436
923	F149650460	054965046	554910046	F14910046	F149100465
924	A286694352	258669435	253119380	A23119380	A231193800
925	O129633185	142963318	587457763	I17457763	I174577631
926	A163521498	006352149	000852094	A10852094	A108520948
927	B119565221	011956522	456450967	U26450967	U264509673
928	D224422111	282442211	776937156	C26937156	C269371567
929	M234493003	373449300	822993795	H22993795	H229937951
930	O286203023	398620302	843614752	J23614752	J236147520
931	C249315090	274931509	269431454	B29431454	B294314542
932	Z200992990	490099299	440098799	T20098799	T200987994
933	X171058868	237105886	736600881	X16600881	X166008810
935	G123341949	062334194	061834189	G11834189	G118341893
934	K270450665	357045066	302044516	F22044516	F220445165
936	J100032467	090003246	040002746	E10002746	E100027462
937	P226954491	402695449	402645449	P22645449	P226454496
939	G286059117	318605911	313605861	G23605861	G236058615
938	J184813462	098481346	043480796	E13480796	E134807965
940	C299283178	279928317	774423262	C24423262	C244232621
941	F234207360	303420736	798415686	E28415686	E284156862
942	E257759991	295775999	290775949	E20775949	E207759496
943	S213966275	431396627	380846122	N20846122	N208461223
944	I185522893	088552289	038501789	D18501789	D185017898
945	X232958616	483295861	477745806	W27745806	W277458062

(b)

圖8 詳細的實驗模擬: (a) 加密 (b) 解密

參考文獻

[1] H. Shi et al., “Understanding Digit-only Financial Account Passwords: ID card, Structure, and Security,” International Journal of Intelligent Systems, vol. 37, pp. 8635-8652, 2022.

[2] D. Prakash, “FPRC5: Improving the Data Security using Format PreservedRC5 Block Cipher Algorithm in Credit Card Application,” ICCES 2021, pp. 775-785, 2021.

[3] C. Lee et al., “Novel Encryption Method of GPS Information in Image File Using Format-Preserving Encryption,” Advances in Intelligent Systems and Computing vol. 994, pp. 815-823, 2020.

[4] J. Zou et al., “Improved Prefix Based Format-Preserving Encryption for Chinese Names,” China Comm., pp. 78-90, 2018.

[5] S. Gupta, “Ensuring Data Security in Databases Using Format Preserving Encrypt ion,” Confluence-2018, pp. 214-217, 2018.

[6] B. Cui et al., “A Data Masking Scheme for Sensitive Big Data based on Format-Preserving Encryption,” IEEE ICSE and IEEE EUC, pp. 518-524, 2017.

[7] N. Chaudhari, “A Cloud Security Approach for Data at Rest Using FPE,” IJCCSA 2015, vol. 5, pp. 11-16, 2015.

[8] M. Mi´covi´c et al., “Network Layer Privacy Protection Using Format-Preserving Encryption,” Electronics, vol. 12, 4800, 2023.

[9] 楊慶隆，黃擴增，曾誌嶽，林泓毅，“以格式保留演算法FF1的身分證字號加密實作” TANET 2023, Nov., 2023。

[10] M. Dworkin, “NIST SP 800-38G Rev. 1, Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption,” doi.org/10.6028/NIST.SP.800-38Gr1-draft, 2019.

[11] W. Stallings, Cryptography and Network Security: Principles and Practice, 8th edition, Pearson Education, 2020.

[12] F.B. Durak and S. Vaudenay, “Breaking the FF3 Format-Preserving Encryption Standard over Small Domains,” https://eprint.iacr.org/2017/521, 2017.

[13] V.T. Hoang et al., “Attacks Only Get Better: How to Break FF3 on Large Domains,” https://eprint.iacr.org/2019/244, 2019.

[14] O. Amon et al.' "Three Third Generation Attacks on the Format Preserving Encryption Scheme FF3," EUROCRYPT 2021, LNCS vol. 12697, 2021.