一個同態墊塞機制

A Homophony Padding Scheme

王銓祺[1]　賴威伸[2]　陳以德[3,*]　葉義雄[4]
[1] Chuan-Chi Wang,　[2] Wei-Shen Lai,　[3,*] I-Te Chen, and　[4] Yi-Shiung Yeh

[1]清雲科技大學資訊工程系
[1] Department of Computer Science and Information Engineering
Ching-Yun University, Taiwan
E-mail: wcc@mail.cyu.edu.tw

[2]建國科技大學資訊管理系
[2] Department of Information Management
Chien-Kuo Technology University, Taiwan
E-mail: wslai.csie84g@nctu.edu.tw

[3]高雄醫學大學通識教育中心
[3] Department of General Educations
Kaohsiung Medical University, Taiwan
E-mail: itchen@csie.nctu.edu.tw

[4]國立交通大學資訊工程系
[4] Department of Computer Science and Information Engineering
National Chiao-Tung University, Taiwan
E-mail: ysyeh@csie.nctu.edu.tw

摘　　要

許多密碼學的應用，都有位元墊塞的問題，如雜湊函數、區塊加密等等。學者們也提出了許多解決雜湊函數位元墊塞問題的方法。但對區塊加密演算法位元墊塞的問題，卻鮮少被討論。所以本論文以同態機制發展出一個解決區塊加密演算法位元墊塞問題的方法，此法同時可增強區塊加密演算法的安全性。

關鍵詞：墊塞、密碼學、區塊加密、雜湊函數、同態

Abstract

Padding problems observably occur in cryptography such as the hash functions and the block ciphers. There are several ways to solve padding problems of the hash functions. But, the padding problems concerning block ciphers are seldom discussed. Therefore, via homophonic mechanism, we propose a novel scheme to solve related padding problems and also to strengthen the security of block ciphers.

Keyword：padding、cryptography、block cipher、hash function、homophony

---

* 通訊作者

1. 簡介

因為訊息(message)的長度不一，在需要切割成區塊(block)的密碼學系統，如雜湊函數(one way hash function)、區塊加密演算法(block cipher)等，都會有墊塞位元(padding)的問題。以雜湊函數為例，在執行雜湊函數前，訊息需要切割成大小一致的區塊。各演算法區塊的大小不一，如 SHA-160 的區塊大小為 512 bits；而 SHA-384 跟 SHA- 512 則為 1024 bits。假定訊息長度為 N，而區塊長度為 b，通常 b 無法整除 N，所以最後一個區塊，需要有墊塞位元 [4~7]。

以 SHA-160 為例，訊息在進入 SHA-160 演算法前得先墊塞位元到足夠的長度。SHA-1 處理的訊息區塊均為 512-bit 的倍數，當最後一個區塊不滿 512 bits 時，就必須墊塞位元，使其成為 512-bit 的倍數。添加墊塞位元的方法為在最後一個區塊，先加入一個位元 ”1”，其餘補 “0”，直到總長度與 448 位元在模 512 中同餘 (長度 = 448 mod 512)。即使是最後一個區塊已為 448 位元在模 512 中同餘，仍會添加墊塞位元。原始訊息的長度以無正負符號(Unsigned)的 64-bit 整數表示後，添加到已墊塞位元訊息的最後面，正好使最後一個區塊成為 512-bit 的倍數。

除了兩個地方外，SHA-384 和 SHA-512 墊塞位元的方式，與 SHA-160 相同。這兩個不同的地方，我們列出於下：

1. SHA-384 和 SHA-512 補 “0”，直到總長度與 896 位元在模 1024 中同餘。
2. 原始訊息的長度以無正負符號(Unsigned)的 128-bit 整數表示後，添加到已墊塞位元訊息的最後面。

所以，SHA-384 和 SHA-512 墊塞後的總長度可被 1024 整除。

區塊加密演算法，有四種加密的模式：Electronic Code Book (ECB)、Cipher Block Chaining (CBC)、Cipher Feedback (DFB) 及 Output Feedback (OFB) 模式等。這四種模式均需要固定長度的區塊。然訊息在切割後，最後一個區塊的長度，不一定能符合區塊長度，所以區塊加密演算法也如同 SHA-160，有墊塞位元的問題。而這個問題鮮少被討論，所以，本論文專門針對區塊加密演算法，討論其墊塞位元，以解決區塊加密演算法墊塞位元問題的方法；同時增強區塊加密演算法的安全性。

一般用於Triple-Data Encryption Standard (DES[3])、RC6 及Advanced Encryption Standard (AES)的墊塞位元方法為加入足夠位元的"0"s, "1"s或亂數位元[8]。相對的，在解密時，必須知道從那個位元起是墊塞位元且加了多少的墊塞位元。並在解密後，把這些墊塞位元刪除。

2. 墊塞位元的策略

區塊加密演算法墊塞位元的方法，一般而言，有下列幾種方式[3][6]:

1. 墊塞 0-bit (0x00)
2. 墊塞 1-bit (0xFF)
3. 墊塞亂數 bits
4. 墊塞空白字元 space characters (0x20)
5. 墊塞 0-bit 並在最後一個 byte加入墊塞位元的長度
6. 用墊塞位元的長度，墊塞所有的位置
7. 墊塞 0x80後面跟足夠的0-bits及訊息的長度

如前所提到的，前三種方法，最常用於Triple-DES、RC6及AES；而加入一個位元 ”1”，其餘補 “0”及原來訊息長度的方法用於SHA及MD系列.

3. 區塊加密演算法的密墊塞位元問題

對任一個區塊加密演算法 X，令原來訊息的長度為 N bytes每一個區塊的長度為 b bytes。若訊息經切割後，最後一個區塊剩餘b-p bytes，則我們必需墊塞 p bytes到最後一個區塊。區塊加密演算法 N, b 及 p 的關係，我們以Fig 1表示之：

Fig1. 區塊加密演算法 N, b 及 p 的關係

3.1 同態區塊加密演算法

第一個介紹同態區塊加密演算法 homophonic block cipher (往後以HBC為縮寫表示之) 的是1998年由謝志敏老師提出的的 homophonic DES [1]。之後葉義雄教授於2002年提出homophonic Rijndael AES [2]。

HBC的概念，可用於任一個區塊加密演算法 X。首先，我們分析區塊加密演算法 X，以便了解在何處加入墊塞位元是最好的。然後精巧地把亂數位元加入我們分析後最適合加入墊塞位元的位置來建構同態區塊加密演算法。藉由這些精巧加入的亂數位元，同態區塊加密演算法可以增加原來區塊加密演算法 X的安全性。舉例來說，在加入8 個亂數位元後，同態 DES^3 使用112 或 168 bits 的金鑰，把56 bits 的明文 (plaintext) 加密成 64 bits 的密文 (ciphertext)；如此的作法可以增加選擇明文攻擊法 (chosen-plaintext attack) 約 2^8 倍的時間 [1][2]。

3.2 解決區塊加密演算法墊塞位元問題的方法

HBC的概念，亦可用於任一個區塊加密演算法 X 來解決墊塞位元的問題。令區塊加密演算法 X 的區塊長度為 c 而 S_c 表示 c bytes 的串流位元。若我們在每一個區塊加入一個byte的亂數，則區塊加密演算法 X 把 S_{c-1} 對應到 S_c。在這個情況下，我們便建構了一個同態的區塊homophonic block (往後以HB為縮寫表示之)。HB 相對的輸入/輸出關係我們以Fig 2表示之。

Fig 2. HB 相對的輸入/輸出關係

能加入區塊加密演算法 X 區塊中的byte 數量，與分析區塊加密演算法 X 的結果有關。在此，我們假設我們在每一個區塊加入 b_{pad} 的亂數位元組。以往的方法是加入 p bytes 到訊息最後一個區塊中，如Fig 1 所示；而在同態區塊加密演算法 X，我們需要 $\lceil p/b_{pad} \rceil$ 個 HBs 來解決墊塞位元問題 (其中 $\lceil\ \rceil$ 表示高斯符號，例如$\lceil 3.5 \rceil = 4$)。若我們令$b_{pad} = 1$，則$p/1 = p$ 個HBs 將被用來解決墊塞位元問題，如Fig 2所示。

因為 p/ b_{pad} 不一定能整除，所以在最後一個區塊，可能會加入少於 b_{pad} 的亂數。也就是

1≤最後一個HB的墊塞位元≤ b_{pad}

如何加入這些墊塞位元來得到最大的擴散 (diffusion)與混亂 (confusion) 決定於分析區塊加密演算法 X 的結果。我們更進一步地設計四個使用HBs的模式：normal block, rear HBs, front HBs and fair HBs，如下所示：

(1) Normal block：不用加入HBs

(2) Rear HBs：HBs 在加在密文的後面

(3) Front HBs：HBs 在加在密文的前面

(4) Fair HBs：一半的HBs 加在密文前面，而另一半的 HBs 加在密文的後面。若 HBs 的個數為奇數，則多出來的一個HB加到密文的前面

加密後，我們另外加入一個 byte 的標頭 (header) 於密文前面，這一 byte 中的前兩個 bits 用來表示那一種使用HBs的模式；而後面6個 bits 用來表示使用HBs的數量。使用HBs的四種模式我們以 Fig 3表示之；而整個加入墊塞位元的演算法，我們以 Algorithm 3.1 表示之。

Fig 3. 密文的標頭

Algorithm 3.1
Input: Plaintext: P
Size of plaintext: N bytes
Block size of block cipher X: *c* bytes
Padding bytes: *p*
Padding bytes in each block: b_{pad}
Usage of HBs: Normal, Rear HBs, Front HBs or Fair HBs.
Output: The C_{HB}= Header || ciphertext (Where || means concatenate)
Begin
//Set header
Case
: Normal: header :=”00xxxxxx”
: Rear HBs: header :=”01xxxxxx”
: Front HBs: header :=”10xxxxxx”
: Fair HBs: header :=”11xxxxxx”
Set the last 6 bits of header := p / b_{pad}
//process
According to Usage of HBs, break a plaintext into block size *b* or *b*- b_{pad}
for i = 1 to $\lceil p / b_{pad} \rceil$ do
Insert random bytes into relative HBs according to Usage of HBs
End
Encrypt those block include normal blocks and HBs to get ciphertext
C_{HB} := Header || ciphertext
End

上述演算法的輸入為一明文而輸出 C_{HB} 則為header || ciphertext。對每一區塊加密前，我們把訊息切割成大小為 b 的區塊並加入適當的亂數到相對應的HBs。對每一區塊加密後，我們加入 header 而得到輸出 C_{HB}。

根據 header 的前兩個 bit，解密者可以知道使用那一種 HBs 的模式；而根據 header 的後六個 bit 解密者知道加入了幾個 HBs。解密者可以分別用區塊加密演算法 X 及同態區塊加密演算法 X 來解密一般的區塊及 HBs，並在解密後，於適當的位置，拿掉加入的那些亂數。

4. 結論

本文指出了雜湊函與區塊加密演算法墊塞位元的問題。我們蒐集並整理了現有解決墊塞位元問題的方法，並提出了一個以同態機制來解決區塊加密演算法位元墊塞問題的方法。而此法不但解決了位元墊塞的問題；同時也增強了區塊加密演算法的安全性。

參考文獻

[1] Tsu-Miin Hsieh, Yi-Shiung Yeh, Yung-Cheng Hsieh, Chan-Chi Wang, “A homophonic DES,” Information Processing Letters, Volume: 66, Issue: 6, June 30, 1998, pp. 317-320.

[2] Yi-Shiung Yeh and Ching-Hung Hsu, "A homophonic rijndael," Journal of Information & Optimization Sciences, Vol. 23, No. 1, 2002, pp.177.184.

[3] "Using Padding in Encryption, " http://www.di-mgt.com.au/cryptopad.html. July 8, 2003.

[4] William Stalling, Cryptography and Network Security: Principle and Practice, 3rd Edition, Prentice Hall, 2003.

[5] Alfred J. Menezes, Paul C. Van, Oorschot and Scott A. Vanstone, Handbook of Applied Cryptography, Boca Raton, FL: CRC Press, 1996.

[6] Niels Ferguson and Bruce Schneier, Practical Cryptography, John Wiley & Sons, 2003.

[7] National Institute of Standards and Technology, NIST FIPS 180-2, " Secure Hash Signature Standard (SHS)," U.S. Department of Commerce, August 1 2002. http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf

[8] Kelsey J., Schneier B., Wagner D., and Hall C. "Cryptanalytic Attacks on Pseudorandom Number Generators," Fast Software Encryption, Fifth International Workshop Proceedings (March 1998), Springer-Verlag, 1998, pp. 168-188. http://www.schneier.com/paper-prngs.html

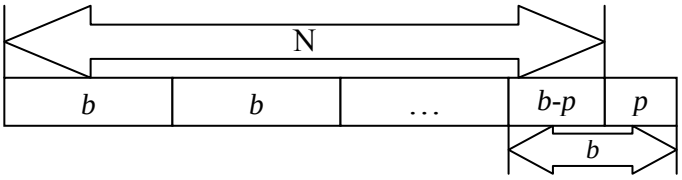

Fig 1. 區塊加密演算法 N, b 及 p 的關係

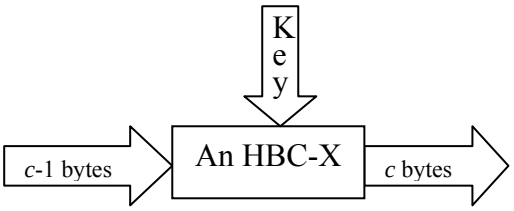

Fig 2. HB 相對的輸入/輸出關係

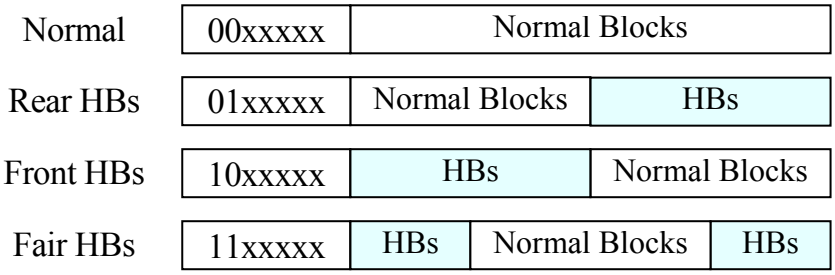

Fig 3. 密文的標頭