Implementacja algorytmu Mini-AES

13 listopada 2014

W odróżnieniu od systemy S-DES, tym razem zarówno klucz jak i tekst są ciągami 16 bitowymi. Operacje na nich będziemy wykonywać jak na macierzach posiadających 2 wiersze i 2 kolumny. W każdej komórce macierzy znajdować się będzie ciąg 4 bitów. Np. niech kluczem początkowym będzie ciąg bitów: $k_0 = [1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0]$. W postaci macierzowej ma on postać

$$\begin{bmatrix} 1,0,1,1 & 0,0,1,0 \\ 1,1,1,1 & 0,1,1,0 \end{bmatrix}. \tag{1}$$

Do implementacji algorytmu będą potrzebne dwie operacje matematyczne: mnożenie i dodawania takich macierzy. O ile dodawanie jest proste bo polega na normalnym dodawaniu macierzy modulo 2 np.

$$\begin{bmatrix} 1,0,1,1 & 0,0,1,0 \\ 1,1,1,1 & 0,1,1,0 \end{bmatrix} + \begin{bmatrix} 1,1,1,1 & 0,0,0,0 \\ 1,1,1,1 & 0,0,0,0 \end{bmatrix} = \begin{bmatrix} 0,1,0,0 & 0,0,1,0 \\ 0,0,0,0 & 0,1,1,0 \end{bmatrix}, \tag{2}$$

to mnożenie jest nieco bardziej skomplikowane. Po pierwsze jest to standardowe mnożenie macierzy. Np.

$$\begin{bmatrix} 1,0,1,1 & 0,0,1,0 \\ 1,1,1,1 & 0,1,1,0 \end{bmatrix} \cdot \begin{bmatrix} 1,1,1,1 & 0,0,0,0 \\ 1,1,1,1 & 0,0,0,0 \end{bmatrix} = \begin{bmatrix} 1,0,1,1*1,1,1,1+0,0,1,0*1,1,1,1 & 1,0,1,1*0,0,0,0+0,0,1,0*0,0,0,0 \\ 1,1,1,1*1,1,1,1+0,1,1,0*1,1,1,1 & 1,1,1,1*0,0,0,0+0,1,1,0*0,0,0,0 \end{bmatrix}, \tag{3}$$

Po drugie należy pamiętać, że chcemy operować na ciągach 4 bitowych, więc ich mnożenie powinno dać w wyniku również ciąg 4 bitowy. Przykładowo rozważmy iloczyn [1,0,1,1] oraz [1,1,1,1]. W pierwszym kroku po prostu mnożymy przez siebie te dwa ciągi otrzymując [1,0,1,1]*[1,1,1,1] = [1,0,1,1,0,0,0] + [0,1,0,1,1,0,0]+[0,0,1,0,1,1,0] + [0,0,0,1,0,1,1] = [1,1,0,1,0,0,1] (po drugim znaku równości działanie zostało rozpisane na wyniki pojedynczych mnożeń, proszę zwrócić uwagę że jest to właściwie odpowiednie przesuwanie bitów). By uzyskać wielomian 4 bitowy dokonujemy dzielenia wyniku [1,1,0,1,0,0,1] przez wielomian redukcyjny [1,0,0,1,1]. Dzielenie sprowadza się do odpowiedniego odejmowania wielomianu redukcyjnego:
1,1 ————
1,1,0,1,0,0,1 mod 1,0,0,1,1
1,0,0,1,1
x————
0,1,0,0,1,0,1
0,1,0,0,1,1
x————
0,0,0,0,0,1,1
x oznaczono dodawanie modulo 2. Resztą z dzielenia jest wielomian [0,0,1,1] i on jest również wynikiem poszukiwanej przez nas operacji mnożenia.
W algorytmie M-AES generacja klucza odbywa się w następujący sposób (proszę również zobaczyć schemat w pliku)

1. Klucz rundy 0 k_0 jest parametrem początkowym
2. Klucz rundy 1 otrzymujemy z k_0 w następujący sposób modyfikując poszczególne elementy macierz ($k_0[0,0]$ to element w pierwszej kolumnnie i pierwszym wierszu):
   $k_1[0,0] = k_0[0,0]$+SBoxE[$k_0[1,1]$] + [0,0,0,1],
   $k_1[1,0] = k_0[1,0]+k_1[0,0]$,
   $k_1[0,1] = k_0[0,1]+k_1[1,0]$,
   $k_1[1,1] = k_0[1,1]+k_1[0,1]$,
   wszystkie dodawania oznaczają dodawanie modulo 2, [0,0,0,1] to stały wielomian, SBoxE[] to funkcja, której specyfikacja znajduje się w pliku cw3_schemat.pdf
3. Klucz rundy 2 otrzymujemy w następujący sposób modyfikując poszczególne elementy macierz
   $klucz_2[0,0] = k_1[0,0]$+SBoxE[$k_1[1,1]$] + [0,0,1,0],
   $k_2[1,0] = k_1[1,0]+k_2[0,0]$,
   $k_2[0,1] = k_1[0,1]+k_2[1,0]$,
   $k_2[1,1] = k_1[1,1]+k_2[0,1]$.

Dodatkowo potrzebne będą funkcje pomocnicze:

1. ZK działa na macierz $A = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix}$, ZK(A) $= \begin{bmatrix} a_{00} & a_{01} \\ a_{11} & a_{10} \end{bmatrix}$
2. MM działa na macierz A MM(A)=m.A gdzie . to mnożenie macierzy a m to stała macierz $m = \begin{bmatrix} 0,0,1,1 & 0,0,1,0 \\ 0,0,1,0 & 0,0,1,1 \end{bmatrix}$
3. $F_S box(A, p)$ gdzie p to E (szyfrowanie) lub D deszyfrowanie. $F_S box(A, E) = \begin{bmatrix} \text{SBoxE}[a_{00}] & \text{SBoxE}[a_{01}] \\ \text{SBoxE}[a_{10}] & \text{SBoxE}[a_{11}] \end{bmatrix}$

Wygenerowanie kluczy dla rund

1. Dodanie do bloków klucz rundy 0
2. Zastosowanie funkcji podstawienia na blokach, zamiana wierszy, mieszanie kolumn, dodanie klucza rundy 1
3. Zastosowanie funkcji podstawienia na blokach, zamiana wierszy, dodanie klucza rundy 2

Szyfrowanie przebiega w następujących krokach

1. Wygenerowanie kluczy k_0 = [1011001011110110],k_1 = [0001110011101010],k_2 = [0101011110111101]
2. Dodanie klucza rundy 0 do wiadomości t=[0011110011000011] A=[1000111000110101]
3. Zastosowanie funkcji $F_{SBox(A,E)}$ A=[0011000000011111]
4. Zastosowanie funkcji ZK(A) A=[0011000011110001]
5. Zastosowanie funkcji MM(A) A=[1000001001000011]
6. Dodanie klucza rundy 1 A= [1001111010101001]
7. Zastosowanie funkcji $F_{SBox(A,E)}$ A=[1010000001101010]
8. Zastosowanie funkcji ZK(A) A= [1010000010100110]
9. Dodanie klucza rundy 2 A= [1111011100011011]

Natomiast odszyfrowanie przebiega następująco:

1. Dodanie Klucza 2 rundy
2. Zastosowanie funkcji ZK(A)
3. Zastosowanie funkcji $F_S Box(A, D)$
4. Dodanie klucza rundy 1
5. Zastosowanie funkcji MM(A)
6. Zastosowanie funkcji ZK(A)
7. Zastosowanie funkcji $F_S Box(A, D)$
8. Dodanie klucza rundy 0

Punktacja (łącznie 10 punktów)

- 6 punktów - działający program dla dowolnego klucza i wiadomości
- 3 punktu - napisanie procedury mnożącej i dzielącej dowolne ciągi 4 bitowe (nie korzystając ze specyficznej struktury macierzy MM)
- 1 punkt - elegenckie zakodowanie SBoxów