Laboratorium z kryptografii
Zajęcia 5-6: Szyfr Mini-AES

1 Zasada działania algorytmu

Szyfr Mini-AES operuje na 16 bitowych blokach tekstu.

1.1 Podstawowe operacje wykorzystywane w algorytmie

Algorytm operuje na specyficznej formie dodawania oraz mnożenia 16 bitowych ciągów danych. W celu ich przystępnego omówienia wygodnie jest posługiwać się formą macierzową ciągu szesnastu bitów $a = (a_0, a_2, \ldots, a_{14}, a_{15})$, tj.:

$$a = (a_0, a_2, \ldots, a_{14}, a_{15}) \quad \rightarrow \quad \begin{bmatrix} (a_0, a_1, a_2, a_3) & (a_4, a_5, a_6, a_7) \\ (a_8, a_9, a_{10}, a_{11}) & (a_{12}, a_{13}, a_{14}, a_{15}) \end{bmatrix} = \begin{bmatrix} a[0,0] & a[0,1] \\ a[1,0] & a[1,1] \end{bmatrix}$$

Przykładowo:

$$k_p = (1,0,1,1,0,0,1,0,1,1,1,1,0,1,1,0) \quad \rightarrow \quad \begin{bmatrix} (1,0,1,1) & (0,0,1,0) \\ (1,1,1,1) & (0,1,1,0) \end{bmatrix} = \begin{bmatrix} k_p[0,0] & k_p[0,1] \\ k_p[1,0] & k_p[1,1] \end{bmatrix}$$

Dodawanie ciągów 16 bitowych:

Dodawanie dwóch ciągów $a = (a_0, a_1, \ldots, a_{14}, a_{15})$ i $b = (b_0, b_1, \ldots, b_{14}, b_{15})$ wykonuje się w podobny sposób dodawania elementów dwóch macierzy 2×2, z tą różnicą że elementami macierzy są ciągi 4 bitowe, które sumowane są binarnie (mod2), tzn.:

$$\begin{aligned} a + b &= (a_0, a_1, \ldots, a_{14}, a_{15}) + (b_0, b_1, \ldots, b_{14}, b_{15}) \\ &= \begin{bmatrix} a[0,0] & a[0,1] \\ a[1,0] & a[1,1] \end{bmatrix} + \begin{bmatrix} b[0,0] & b[0,1] \\ b[1,0] & b[1,1] \end{bmatrix} \\ &= \begin{bmatrix} a[0,0] \oplus b[0,0] & a[0,1] \oplus b[0,1] \\ a[1,0] \oplus b[1,0] & a[1,1] \oplus b[1,1] \end{bmatrix} \end{aligned}$$

Przykładowo

$$\begin{aligned} k_p + f &= (1,0,1,1,0,0,1,0,1,1,1,1,0,1,1,0) + (1,1,1,1,0,0,0,0,1,1,1,1,0,0,0,0) \\ &= \begin{bmatrix} (1,0,1,1) & (0,0,1,0) \\ (1,1,1,1) & (0,1,1,0) \end{bmatrix} + \begin{bmatrix} (1,1,1,1) & (0,0,0,0) \\ (1,1,1,1) & (0,0,0,0) \end{bmatrix} \\ &= \begin{bmatrix} (1,0,1,1) \oplus (1,1,1,1) & (0,0,1,0) \oplus (0,0,0,0) \\ (1,1,1,1) \oplus (1,1,1,1) & (0,1,1,0) \oplus (0,0,0,0) \end{bmatrix} \\ &= \begin{bmatrix} (0,1,0,0) & (0,0,1,0) \\ (0,0,0,0) & (0,1,1,0) \end{bmatrix} \\ &= (0,1,0,0,0,0,1,0,0,0,0,0,0,1,1,0) \end{aligned}$$

Mnożenie ciągów 16 bitowych

Mnożenie dwóch ciągów także można przedstawić w postaci mnożenia dwóch macierzy kwadratowych 2×2

$$\begin{aligned} a \cdot b &= (a_0, a_1, \ldots, a_{14}, a_{15}) \cdot (b_0, b_1, \ldots, b_{14}, b_{15}) \\ &= \begin{bmatrix} a[0,0] & a[0,1] \\ a[1,0] & a[1,1] \end{bmatrix} \cdot \begin{bmatrix} b[0,0] & b[0,1] \\ b[1,0] & b[1,1] \end{bmatrix} \\ &= \begin{bmatrix} a[0,0] \odot b[0,0] \oplus a[0,1] \odot b[1,0] & a[0,0] \odot b[0,1] \oplus a[0,1] \odot b[1,1] \\ a[1,0] \odot b[0,0] \oplus a[1,1] \odot b[1,0] & a[1,0] \odot b[0,1] \oplus a[1,1] \odot b[1,1] \end{bmatrix} \end{aligned}$$

gdzie specyficzna operacja mnożenia $\odot$ przekształca dwa ciągi czterobitowe $\alpha = (\alpha_0, \alpha_1, \alpha_2, \alpha_3)$ i $\beta = (\beta_0, \beta_1, \beta_2, \beta_3)$ jeden ciąg 4-bitowy $\gamma = (\gamma_0, \gamma_1, \gamma_2, \gamma_3)$. Przypisując każdemu z tych ciągów wielomian stopnia trzeciego nad ciałem $\mathbb{Z}_2$, np.:

$$(\alpha_0, \alpha_1, \alpha_2, \alpha_3) \quad \leftrightarrow \quad \alpha_0 x^3 + \alpha_1 x^2 + \alpha_2 x + \alpha_3,$$

działanie $\odot$ stanowi resztę z dzielenia $R(\alpha \cdot \beta | r)$ wielomianu $\alpha \cdot \beta$ przez wielomian redukcyjny $r = x^4 + x + 1$, tj,

$$\gamma_0 x^3 + \gamma_1 x^2 + \gamma_2 x + \gamma_3 \quad = \quad R\left[\left(\alpha_0 x^3 + \alpha_1 x^2 + \alpha_2 x + \alpha_3\right) \cdot \left(\beta_0 x^3 + \beta_1 x^2 + \beta_2 x + \beta_3\right) | (x^4 + x + 1)\right].$$

Wielomian redukcyjny r jest stały dla szyfru Mini-AES.
Przykładowo:

$$\begin{aligned}
k_p \cdot f &= (1,0,1,1,0,0,1,0,1,1,1,1,0,1,1,0) \cdot (1,1,1,1,0,0,0,0,1,1,1,1,0,0,0,0) \\
&= \left[\begin{array}{cc} (1,0,1,1) & (0,0,1,0) \\ (1,1,1,1) & (0,1,1,0) \end{array}\right] \cdot \left[\begin{array}{cc} (1,1,1,1) & (0,0,0,0) \\ (1,1,1,1) & (0,0,0,0) \end{array}\right] \\
&= \left[\begin{array}{cc} (1,0,1,1) \odot (1,1,1,1) \oplus (0,0,1,0) \odot (1,1,1,1) & (1,0,1,1) \odot (0,0,0,0) \oplus (0,0,1,0) \odot (0,0,0,0) \\ (1,1,1,1) \odot (1,1,1,1) \oplus (0,1,1,0) \odot (1,1,1,1) & (1,1,1,1) \odot (0,0,0,0) \oplus (0,1,1,0) \odot (0,0,0,0) \end{array}\right]
\end{aligned}$$

oraz

$$\begin{array}{lcl}
(1,0,1,1) \odot (1,1,1,1) & \leftrightarrow & (x^3 + x + 1) \odot (x^3 + x^2 + x + 1) \\
\left[(1,0,1,1) \cdot (1,1,1,1)\right] mod (1,0,0,1,1) & \leftrightarrow & \left[(x^3 + x + 1) \cdot (x^3 + x^2 + x + 1)\right] mod (x^4 + x + 1) \\
(1,1,0,1,0,0,1) mod (1,0,0,1,1) & \leftrightarrow & (x^6 + x^5 + x^3 + 1) mod (x^4 + x + 1)
\end{array}$$

co daje:

$$\begin{array}{cccccccc}
 & & & & & 1 & 1 & 0 \\
\hline
 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\
\oplus & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
\hline
 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
\oplus & & 1 & 0 & 0 & 1 & 1 & 0 \\
\hline
 & & & & 0 & 0 & 1 & 1
\end{array}
\; mod(1,0,0,1,1)
\quad \leftrightarrow \quad
\begin{array}{ccccccc}
 & & & & x^2 & +x & \\
\hline
x^6 & +x^5 & +0x^4 & +x^3 & +0x^2 & +0x & +1 \\
-x^6 & & & -x^3 & -x^2 & & \\
\hline
 & +x^5 & +0x^4 & +0x^3 & +x^2 & +0x & +1 \\
 & -x^5 & & & -x^2 & -x & \\
\hline
 & & & & & x & +1
\end{array}
\; : (x^4 + x + 1)$$

1.2 Funkcje wykorzystywane w algorytmie

Szyfr Mini-AES używa dwóch dodatkowych operacji. Pierwszą operacja ZK zamienia miejscami cztery przedostatnie z czterema ostatnimi bitami ciągu wejściowego $a = (a_0, a_2, \ldots, a_{14}, a_{15})$, co można zapisać jako:

$$ZK(a) \quad = \quad ZK\left(\left[\begin{array}{cc} a[0,0] & a[1,0] \\ a[0,1] & a[1,1] \end{array}\right]\right) = ZK\left(\left[\begin{array}{cc} a[0,0] & a[1,0] \\ a[1,1] & a[0,1] \end{array}\right]\right)$$

Druga operacja $F_{SBox}(X, a)$, gdzie $X \in \{D, E\}, a \in \{0,1\}^{16}$, opiera się na wykorzystaniu funkcji $SBoxE, SBoxD : \{0,1\}^4 \to \{0,1\}^4$ w następujący sposób:

$$\begin{aligned}
F_{SBox}(X, a) &= F_{SBox}\left(X, \left[\begin{array}{cc} a[0,0] & a[1,0] \\ a[0,1] & a[1,1] \end{array}\right]\right) \\
&= \left[\begin{array}{cc} SBoxX(a[0,0]) & SBoxX(a[1,0]) \\ SBoxX(a[0,1]) & SBoxX(a[1,1]) \end{array}\right]
\end{aligned}$$

Funckje $SBoxE$ oraz $SBoxD$ przekształcają ciągi 4 bitowe na ciągi 4 bitowe w następujący sposób:

SBoxE				**SBoxD**			
ciąg wejściowy		ciąg wyjściowy		ciąg wejściowy		ciąg wyjściowy	
(0,0,0,0)	0x0	(1,1,1,0)	0xe	(0,0,0,0)	0x0	(1,1,1,0)	0xe
(0,0,0,1)	0x1	(0,1,0,0)	0x4	(0,0,0,1)	0x1	(0,0,1,1)	0x3
(0,0,1,0)	0x2	(1,1,0,1)	0xd	(0,0,1,0)	0x2	(0,1,0,0)	0x4
(0,0,1,1)	0x3	(0,0,0,1)	0x1	(0,0,1,1)	0x3	(1,0,0,0)	0x8
(0,1,0,0)	0x4	(0,0,1,0)	0x2	(0,1,0,0)	0x4	(0,0,0,1)	0x1
(0,1,0,1)	0x5	(1,1,1,1)	0xf	(0,1,0,1)	0x5	(1,1,0,0)	0xc
(0,1,1,0)	0x6	(1,0,1,1)	0xb	(0,1,1,0)	0x6	(1,0,1,0)	0xa
(0,1,1,1)	0x7	(1,0,0,0)	0x8	(0,1,1,1)	0x7	(1,1,1,1)	0xf
(1,0,0,0)	0x8	(0,0,1,1)	0x3	(1,0,0,0)	0x8	(0,1,1,1)	0x7
(1,0,0,1)	0x9	(1,0,1,0)	0xa	(1,0,0,1)	0x9	(1,1,0,1)	0xd
(1,0,1,0)	0xa	(0,1,1,0)	0x6	(1,0,1,0)	0xa	(1,0,0,1)	0x9
(1,0,1,1)	0xb	(1,1,0,0)	0xc	(1,0,1,1)	0xb	(0,1,1,0)	0x6
(1,1,0,0)	0xc	(0,1,0,1)	0x5	(1,1,0,0)	0xc	(1,0,1,1)	0xb
(1,1,0,1)	0xd	(1,0,0,1)	0x9	(1,1,0,1)	0xd	(0,0,1,0)	0x2
(1,1,1,0)	0xe	(0,0,0,0)	0x0	(1,1,1,0)	0xe	(0,0,0,0)	0x0
(1,1,1,1)	0xf	(0,1,1,1)	0x7	(1,1,1,1)	0xf	(0,1,0,1)	0x5

Tabela 1: Ciągi zwracane przez $SBoxE$ i $SBoxD$ w funkcji ciągu wejściowego

1.3 Generacja kluczy I i II rundy

Niech $a[i,j]$, gdzie $i,j \in \{0,1\}$ oznacza 4 bitowy ciąg znajdujący się w i-tym wierszu oraz j-tej kolumnie reprezentacji macierzowej ciagu bitów $a = (a_1, a_2, \ldots, a_{14}, a_{15})$. Kolejne elementy klucza pierwszej rundy otrzymywane są na podstawie 16 bitowego klucza początkowego k_p (przykładowo k_p = (1,0,1,1,0,0,1,0,1,1,1,1,0,1,1,0)) w następujący sposób:

$$\begin{aligned}
k_1[0,0] &= k_p[0,0] \oplus SBoxE(k_p[1,1]) \oplus (0,0,0,1) \\
k_1[1,0] &= k_p[1,0] \oplus k_1[0,0] \\
k_1[0,1] &= k_p[0,1] \oplus k_1[1,0] \\
k_1[1,1] &= k_p[1,1] \oplus k_1[0,1]
\end{aligned}$$

Przykładowo k_1 = (0,0,0,1,1,1,0,0,1,1,1,0,1,0,1,0).
Klucz rundy drugiej k_2 generuje się w sposób analogiczny przy wykorzystaniu klucza k_1, tzn.:

$$\begin{aligned}
k_2[0,0] &= k_1[0,0] \oplus SBoxE(k_1[1,1]) \oplus (0,0,1,0) \\
k_2[1,0] &= k_1[1,0] \oplus k_2[0,0] \\
k_2[0,1] &= k_1[0,1] \oplus k_2[1,0] \\
k_2[1,1] &= k_1[1,1] \oplus k_2[0,1]
\end{aligned}$$

Przykładowo k_2 = (0,1,0,1,0,1,1,1,1,0,1,1,1,1,0,1).

1.4 Struktura algorytmu

Szyfrowanie 16 bitowego tekstu t (t=(0,0,1,1,1,1,0,0,1,1,0,0,0,0,1,1)) polega na jego przetwarzaniu w następującej kolejności:

1. Dodawanie klucza początkowego k_p do tekstu t: $t = t + k_p$.
   t = (1,0,0,0,1,1,1,0,0,0,1,1,0,1,0,1).
2. Zastosowanie funkcji $F_{SBox}(E,t)$
   t = (0,0,1,1,0,0,0,0,0,0,0,1,1,1,1,1).
3. Zastosowanie funkcji $ZK(t)$
   t = (0,0,1,1,0,0,0,0,1,1,1,1,0,0,0,1).
4. Przemnożenie tekstu t przez ciąg bitów $m = (0,0,1,1,0,0,1,0,0,0,1,0,0,0,1,1)$: $t = m \cdot t$
   t = (1,0,0,0,0,0,1,0,0,1,0,0,0,0,1,1).

5. Dodawanie klucza rundy pierwszej k_1 do tekstu t: $t = t + k_1$.
   t = (1,0,0,1,1,1,1,0,1,0,1,0,1,0,0,1).

6. Zastosowanie funkcji $F_{SBox}(E, t)$
   t = (1,0,1,0,0,0,0,0,0,1,1,0,1,0,1,0).

7. Zastosowanie funkcji $ZK(t)$
   t = (1,0,1,0,0,0,0,0,1,0,1,0,0,1,1,0).

8. Dodawanie klucza rundy drugiej k_2 do tekstu t: $t = t + k_2$.
   t = (1,1,1,1,0,1,1,1,0,0,0,1,1,0,1,1).

Wynik ostatniej operacji jest ostatecznym szyfrogramem.

Deszyfrowanie odbywa się poprzez przeprowadzenie kolejnych operacji na szyfrogramie:

1. Dodawanie klucza rundy drugiej k_2 do szyfrogramu s: $s = s + k_2$.
2. Zastosowanie funkcji $ZK(s)$
3. Zastosowanie funkcji $F_{SBox}(D, s)$
4. Dodawanie klucza rundy pierwszej k_1 do s: $s = s + k_1$.
5. Przemnożenie s przez ciąg bitów $m = (0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1)$: $s = m \cdot s$
6. Zastosowanie funkcji $ZK(s)$
7. Zastosowanie funkcji $F_{SBox}(D, s)$
8. Dodawanie klucza początkowego k_p do s: $s = s + k_p$.

2 ZADANIA

1. Dla zadanych w konsoli dwóch 4-bitowych ciągów $a \in \{0, 1\}^4$ i $b \in \{0, 1\}^4$ (podanych w zapisie szesnastkowym), napisać program dokonujący operacji $a \odot b$ i obliczający wartość funkcji $SBoxE(a \odot b)$ oraz $SBoxD(a \odot b)$. Wynik mnożenia $\odot$ i wartości zwracane przez funkcje $SBox$ mają być wyświetlane przez program.

2. Napisać program szyfrujący algorytmem Mini-AES przyjmujący w konsoli argumenty „**klucz**“ oraz „**tekst**” będącymi ciągami znaków kodowanymi w zapisie szesnastkowym. W przypadku kiedy zadany **klucz** jest krótszy niż 4 znaki heksadecymalne lub długość zadanego **tekstu** nie jest wielokrotnością, należy odpowiednio uzupełnić je zerami po prawej stronie ciągu. Program na wyświetlać wygenerowane klucze I i II rundy oraz otrzymany szyfrogram.

Punktacja - łącznie 10 punktów

- 3 punkty - poprawnie działające mnożenie $\odot$
- 1 punkty - poprawnie działające funkcje SBox
- 3 punkty - poprawna generacja kluczy I i II
- 3 punkty - otrzymanie poprawnego szyfrogramu

3 WEKTORY TESTOWE

key:	0xb2f6	0x32f6	0xffff	0xffff	0xfff	0x456
text:	0x3cc3	0x3cc3	0x3cc3	0xffff	0xfff	0xabcd
key1:	0x1cea	0x9462	0x9966	0x9966	0x00f0	0xb8d8
key2:	0x57bd	0x6406	0x0f69	0x0f69	0xc333	0xaf77
step1:	0x8e35	0x0e35	0xc33c	0x0000	0x0000	0xeead
step2:	0x301f	0xe01f	0x5115	0xeeee	0xeeee	0x0069
step3:	0x30f1	0xe0f1	0x5151	0xeeee	0xeeee	0x0096
step4:	0x8243	0xc2d3	0x5151	0xeeee	0xeeee	0x1c8a
step5:	0x9ea9	0x56b1	0xc837	0x7788	0xee1e	0xa452
step6:	0xa06a	0xfbc4	0x5318	0x8833	0x0040	0x62fd
step7:	0xa0a6	0xfb4c	0x5381	0x8833	0x0004	0x62df
final:	0xf71b	0x9f4a	0x5ce8	0x875a	0xc337	0xcda8

Tabela 2: Wektory testowe dla szyfru MINI-AES. Kolejne wartości „step“ odzwierciedlają odpowiednie kroki algorytmu szyfrującego opisanego w rozdziale 1.4.