Implementacja szyfru S-DES

12 października 2015

Jest to uproszona wersja systemu DES, systemu korzystającego z szyfrowania z kluczem symetrycznym. Do odszyfrowania używa się tego samego klucza co do szyfrowania. Parametrami wejściowymi do programu są: tekst początkowy (t) oraz klucz początkowy k_p (mający 10 bitów). Poniższy opis prawdopodobnie może być zrozumiany jedynie w połączeniu ze schematami z pliku.

1. Wiadomość początkową przekształcamy do formy bitowej (w tym ćwiczeniu od razu będziemy pracować z plikami bitowymi). Pierwszy krok algorytmu polega na podziale wiadomości na bloki 8 bitowe. Jeżeli końcowy blok nie zawiera dokładnie 8 bitów należy uzupełnić go 0. Np. niech tekstem będzie zmienna w=[1,1,1,1,0,0,0].

2. Szyfrowanie tekstu przebiega w dwóch rundach. Do każdej potrzeba klucza uzyskanego z klucza początkowego.

3. Klucze do szyfrowania uzyskujemy z klucza początkowego w następujący sposób (proszę zobaczyć również schemat blokowy w pliku cw2_schemat.pdf). Załóżmy, że k_p = [1,1,0,0,0,0,0,0,1,1].

   (a) Na ten ciąg bitowy działamy permutacją, której przypiszemy symbol P10. Permutacja ta działa według schematu $[0,1,2,3,4,5,6,7,8,9] \longrightarrow [2,4,1,6,3,9,0,8,7,5]$, co oznacza, że w ciągu wyjściowym na 0 pozycji ma znaleźć się element z 2 pozycji ciągu wejściowego (proszę zwrócić uwagę, że numeracja zaczyna się od 0 elementu), na pozycji 1 element z pozycji 4 itp. Przykład P10(k_p) = [0,0,1,0,0,1,1,1,0,0].

   (b) Następnie dzielimy tak otrzymany ciąg na dwa ciągi: pierwszy składający się z pierwszych pięciu bitów [0,0,1,0,0] oraz analogicznie drugi [1,1,1,0,0]. Każdą z ciągów przekształcamy, przesuwając bity w lewo o 1 czyli dla pierwszego ciągu operacja ma postać $[0,0,1,0,0] \longrightarrow [0,1,0,0,0]$ ($k_0^1 = [0,1,0,0,0]$) a dla drugiego $[1,1,1,0,0] \longrightarrow [1,1,0,0,1]$, ($k_1^1 = [1,1,0,0,1]$). Ogólnie w przyjętej wcześniej konwencji operację tą zdefiniujemy jako SL1, jej schemat dany jest przez $[0,1,2,3,4] \longrightarrow [1,2,3,4,0]$.

   (c) Następnie łączymy obie połówki (w kolejności $k_0^1 + k_0^2$) otrzymując $[0,1,0,0,0,1,1,0,0,1]$. Na tak otrzymany ciąg działamy permutacją, która działa według schematu $[0,1,2,3,4,5,6,7,8,9] \longrightarrow [5,2,6,3,7,4,9,8]$. Permutacja ta wybiera 8 z 10 bitów i oznaczymy ją jako P10w8. W naszym przykładzie P10w8([0,1,0,0,0,1,1,0,0,1])= [1,0,1,0,0,0,1,0]. Ten ciąg bitów jest kluczem 1 rundy (oznaczenie $k_1 = [1,0,1,0,0,0,1,0]$). Klucz drugiej rundy otrzymujemy poprzez przesunięcie uprzedni otrzymanych ciągów k_0^1 i k_0^2 o dwa bity w lewo. SL2 : $[0,1,2,3,4] \longrightarrow [2,3,4,0,1]$. Tak otrzymane ciągi łączymy i działamy na nie permutacją P10w8. Wynik tych operacji w naszym przykładzie to ciąg $[0,0,0,0,1,1,1,1]$. Ten ciąg stanowi klucz 2 rundy $k_2 = [0,0,0,0,1,1,1,1]$

4. Teraz omówimy zasadniczą procedurę szyfrującą

   (a) Tekst do zaszyfrowania t= [1,0,1,1,1,0,0,0] dzielimy na dwa 4 bitowe ciągi [1,0,1,1], [1,0,0,0].

   (b) Tworzymy dwie kopie drugiego ciągu. Na pierwszą z nich działamy operacją P4w8: $[0,1,2,3] \longrightarrow [3,0,1,2,1,2,3,0]$. U nas P4w8([1,0,0,0]) = [0,1,0,0,0,0,0,0,1].

   (c) Do tego ciągu dodajemu binarnie klucz odpowiedniej rundy. Dla rundy pierwszej w naszym przykładzie
   Xor([0,1,0,0,0,0,0,0,1],[1,0,1,0,0,0,0,1,0])= $[0,1,0,0,0,0,0,0,1] \oplus [1,0,1,0,0,0,0,1,0] = [1,1,1,0,0,0,0,1,1]$.

   (d) wynik dodawania ponownie dzielimy na dwa 4 bitowe ciągi i działamy na nie funkcjami SBox. Są to funkcje nieliniowe zapewniające bezpieczeństwo systemu. Ich definicje znajdują się w pliku txt umiszczonym w dropboxsie. W naszym przykładzie SBox1(1110) = [1,1], SBox2(0011)=[0,0].

   (e) Tak otrzymane ciągi łączymy w ciąg 4 bitowe [1,1,0,0]. Dziłamy na niego permutacją $P4 : [0,1,2,3] \longrightarrow [1,3,2,0]$. Tutaj P4([1,1,0,0])=[1,0,0,1].

   (f) Ciąg po permutacji P4 dodajemy binarnie do pierwszych 4 bitów tekstu t. Tutaj $Xor([1,0,1,1],[1,0,0,1]) = [0,0,1,0]$.

   (g) Teraz łączymy ciąg z poprzedniego kroku z ostatnimi 4 bitami wiadomości. Tutaj $[0,0,1,0]+[1,0,0,0] = [0,0,1,0,1,0,0,0]$. Jest to wynik zasadniczej procedury szyfrującej.

   Teraz możemy przedstawić pełen proces szyfrowania wiadomości w = [1,1,1,1,0,0,0].

(a) Na wiadomość w działamy permutacją wstępną PW:$[0,1,2,3,4,5,6,7] \longrightarrow [1,5,2,0,3,7,4,6]$. U nas PW([1,1,1,1,0,0,0])= [1,0,1,1,1,0,0,0].

(b) Tak otrzymany ciąg szyfrujemy przy użyciu klucza 1 rundy $[1,0,1,1,1,0,0,0] \longrightarrow [0,0,1,0,1,0,0,0]$.

(c) Wiadomość z poprzedniego kroku modyfikujemy poprzez zamianę miejsc pierwszych i ostatnich 4 bitów $[0,0,1,0,1,0,0,0] \longrightarrow [1,0,0,0,0,0,1,0]$.

(d) Ciąg z kroku poprzedniego szyfrujemy przy użyciu klucza 2 rundy. W wyniku otrzymujemy [0,1,0,1,0,0,1,0].

(e) Na końcu ciąg bitów permutujemy przy pomozy permutacji odwrotnej PO:$[0,1,2,3,4,5,6,7] \longrightarrow [3,0,2,4,6,1,7,5]$, tutaj PO([0,1,0,1,0,0,1,0])=1000110.

5. Odszyfrowanie przebiega w sposób odwrotny:

   (a) Permutacja wstępna PW

   (b) Szyfrowanie przy użycia klucza 2 rundy

   (c) Zamiana miejsc pierwszej i drugiej połowy ciągu bitów

   (d) Szyfrowanie przy pomocy klucza 1 rundu

   (e) Permutacja odwrotna PO

Lista permutacji

- PW: $[0,1,2,3,4,5,6,7] \longrightarrow [1,5,2,0,3,7,4,6]$
- PO: $[0,1,2,3,4,5,6,7] \longrightarrow [3,0,2,4,6,1,7,5]$
- P10: $[0,1,2,3,4,5,6,7,8,9] \longrightarrow [2,4,1,6,3,9,0,8,7,5]$
- P10w8: $[0,1,2,3,4,5,6,7,8,9] \longrightarrow [5,2,6,3,7,4,9,8]$
- P4w8: $[0,1,2,3] \longrightarrow [3,0,1,2,1,2,3,0]$
- SL1: $[0,1,2,3] \longrightarrow [1,2,3,0]$
- SL2: $[0,1,2,3] \longrightarrow [2,3,0,1]$
- P4: $[0,1,2,3] \longrightarrow [1,3,2,0]$

Punktacja (łącznie 10)

- 6 punktów -prawidłowo działające szyfrowanie i odszyfrowanie dla dowolnego początkowego klucza i wiadomości (wiadomość i klucz może być ustawiana np. w programie, nie ma konieczności wczytywania z pliku).
- 2 punkty - eleganckie zakodowanie permutacji, tzn. najlepiej funckja której podaje się wzór ciągu wyjściowego.
- 2 punkty - eleganckie zakodowanie SBoxów, tzn. nie lubimy dużej ilości intrukcji warunkowych typu if.