6. Tryby stosowania szyfrów blokowych i szyfry strumieniowe

6.1 Wprowadzenie

1.Mamy 2 zasadnicze kategorie szyfrów szyfry blokowe i szyfry strumieniowe

2. Szyfr blokowy działa na blokach o ustalonej długości (jednostkach tekstu). Formalnie rzecz biorąc szyfr blokowy nie jest szyfrem ponieważ nie działa na wiadomościach jawnych dowolnej długości. Każdy szyfr blokowy daje się jednak łatwo rozszerzyć do szyfru pracującego na dowolnie długich wiadomościach jawnych. W praktyce bloki są to najczęściej słowa binarne np. 64, 128 czy 256 bitowe.

3. Dlaczego stosujemy tryby. Ponieważ wiadomości szyfrowane *m* mogą być dowolnej długości musimy z szyfru blokowego zrobić „prawdziwy szyfr" pracujący na tekstach dowolnej długości. Musimy jednocześnie zrobić to tak zręcznie by nie ułatwić zadania kryptoanalitykowi.

Podstawowe tryby stosowania szyfrów blokowych to ECB (Electronic Code Book) i CBC (Cipher Block Chaining)

Uzupełnianie ostatniego bloku do pełnego bloku

Długość wiadomości jawnej nie musi być wielokrotnością długości bloku. Jeśli stosujemy szyfr blokowy musimy sobie z tym problemem jakoś poradzić.

1. Metoda z podaniem liczby dodanych symboli.

Wiadomość jawną *m* dopełniamy ustalonym symbolem (np. 0 lub 1) ale kilka ostatnich znaków (np. 8 w przypadku alfabetu binarnego) traktujemy jako liczbę w zapisie NKB informującą układ deszyfrujący o ilości dodanych symboli, które trzeba usunąć by uzyskać pierwotny tekst jawny *m*.

Uzyskujemy w ten sposób tekst o długości równej wielokrotności długości bloku.

2. Metoda z dodaniem symbolu EOF.

Możemy zastosować specjalny symbol EOF (end of file) nie występujący w tekście jawnym a następnie uzupełnić blok do końca ustalonymi znakami w sposób losowy.

Można też po EOF do uzupełniania zastosować odpowiedni fragment kryptogramu z przedostatniego bloku.

6.2 Tryb szyfrowania ECB – (ang. Electronic Code Book)

Zalety ECB

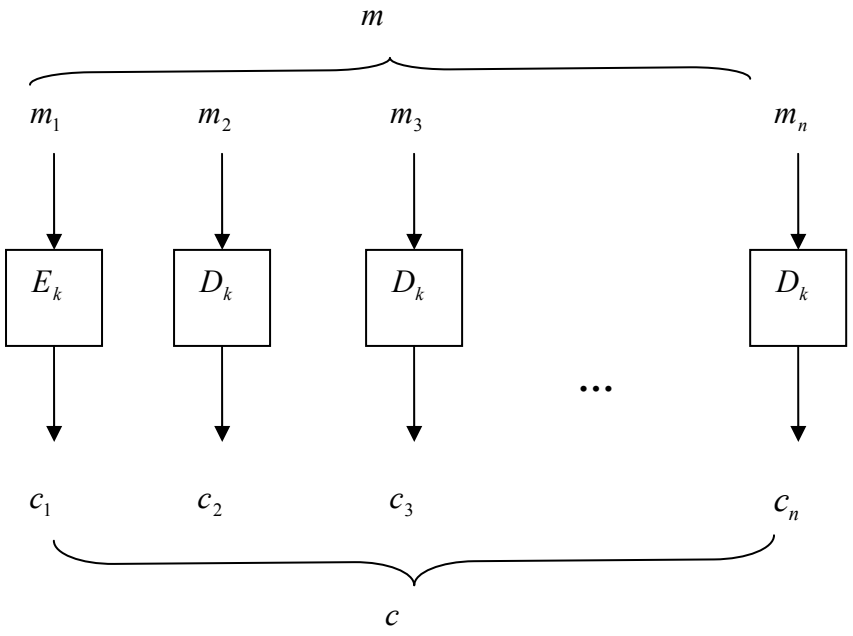
1.Łatwa naturalna deszyfracja

2.Brak, propagacji błędów do kolejnych bloków wiadomości zdeszyfrowanej przy zaistnieniu błędu transmisji w jakimś bloku przesyłanego kryptogramu lub celowego wprowadzenia błędu do bloku. Uszkodzenie lub utrata pojedynczych bitów nie ma wpływu na deszyfrowanie pozostałych części kryptogramu.

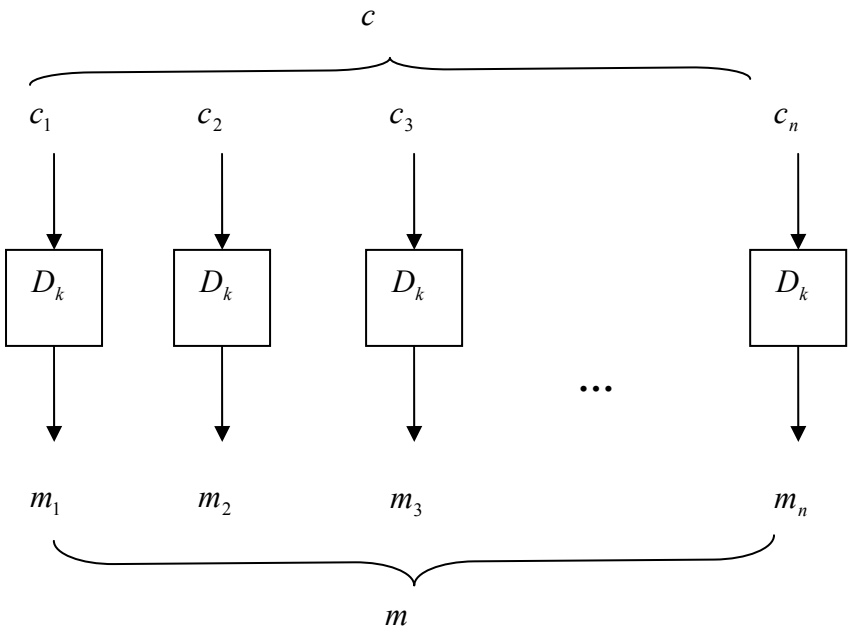
Wada ECB . Możliwość zmian szyfrogramu na zasadzie podstawienia bloku Możliwy jest więc atak na strony wymieniające dane zaszyfrowane nie wymagający złamania szyfru, oto przykład.

Przykład
Typowy atak na tryb ECB. Wyobraźmy sobie wymianę danych pomiędzy bankami w postaci kryptogramów stanowiących przekazy pieniężne. Jeśli zlokalizujemy pola w kryptogramie w których znajduje się zaszyfrowana informacja o przesyłanej kwocie to wymiana tego fragmentu kryptogramu prowadzi do realizacji zmodyfikowanej tranzakcji.

a) Szyfrowanie



b) Deszyfrowanie

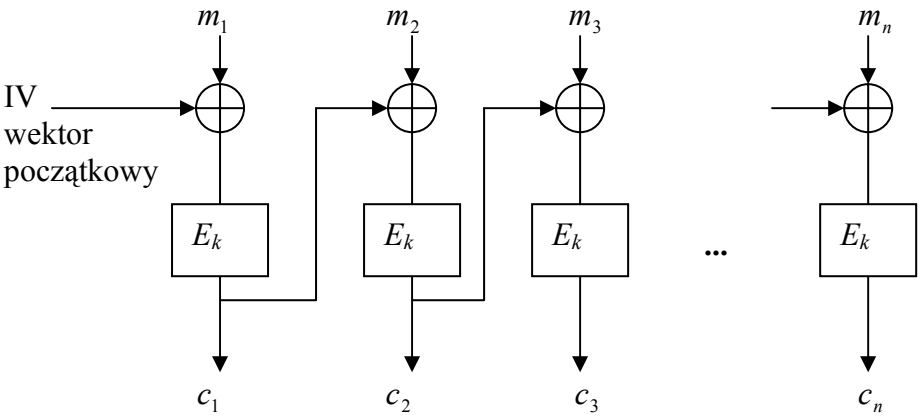


Rys. 6.1. Sposób działania trybu ECB a) szyfrowanie w trybie ECB b) deszyfrowanie w trybie ECB. Blok m_n jest ewentualnie uzupełniany.

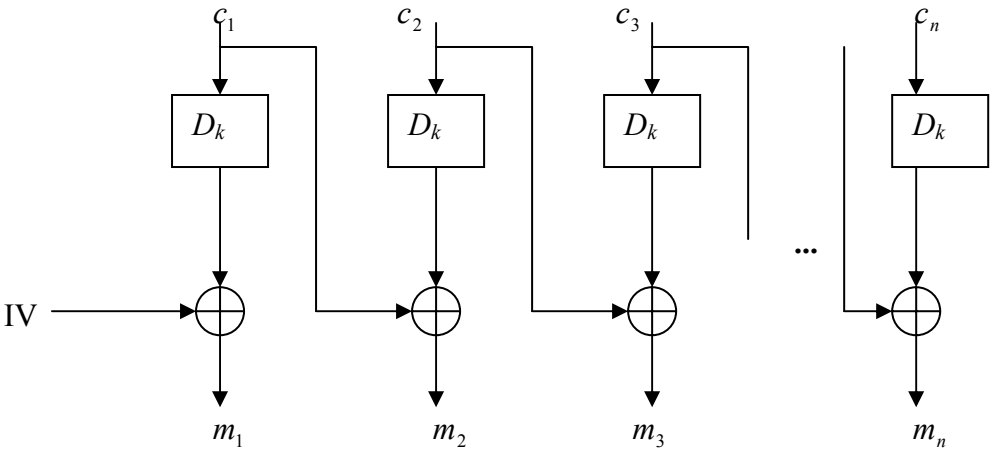
6.3 Tryb szyfrowania CBC

Tryb szyfrowania CBC (ang. Cipher Block Chaining CBC) to inaczej tryb „wiązania bloków zaszyfrowanych” .

Szyfrowanie w trybie CBC jest metodą, dzięki której ten sam blok tekstu jawnego jest szyfrowany w różnych miejscach w różny sposób.



Deszyfracja w trybie CBC



Rys. 6.2 Sposób działania trybu CBC

Osiągane jest to w trybie CBC w następujący sposób. Rozważmy algorytm szyfrujący bloki długości r. Jeśli tekst jawny m składa się z bloków $m_1, m_2, m_3, ..., m_n$ o długości r, to kryptogram uzyskany za pomocą klucza k składa się z bloków $c_1, c_2, c_3, ..., c_n$ (również o długości r) zdefiniowanych dla każdego $i \in <1, n>$ następująco:

$$c_i = E_k(m_i \oplus c_{i-1}),$$

gdzie E_k jest blokową funkcją szyfrującą (bloki o długości r) dla klucza k oraz przyjmujemy, że $c_0 = IV$. Ciąg początkowy lub jak mówimy wektor początkowy IV (ang. Initial Value) występujący w powyższym wzorze jest generowany losowo i przesyłany w sposób niezaszyfrowany.

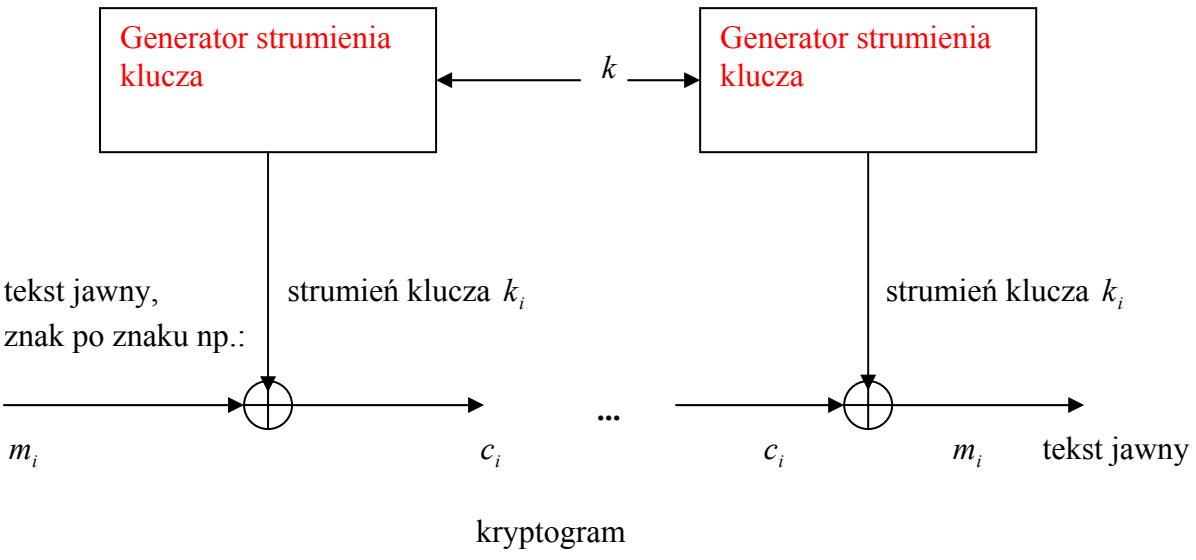
Deszyfracja natomiast opisywana jest wzorem: dla każdego $i \in <1, n>$

$$m_i = c_{i-1} \oplus D_k(c_i).$$

przy czym jak poprzednio $c_0 = IV$ i D_k oznacza funkcję deszyfrującą dla bloków długości r i klucza k):

6.4 Szyfry strumieniowe

Załóżmy, że mamy do zaszyfrowania tekst jawny $(m_i)_{i=1}^{N_0}$ o długości N_0, gdzie $m_i \in \{0,1\}$. Generujemy pseudolosowy ciąg bitów $(s_i)_{i=1}^{N_0}$ i szyfrujemy tekst jawny jako ciąg bitów $(c_i)_{i=1}^{N_0}$, gdzie $c_i = s_i \oplus a_i$ dla każdego i=1,2,..., N_0.



Rys. 6. 3 Układ realizujący szyfr strumieniowy

Szyfrowanie opisane jest równaniem

$$c_i = m_i \oplus k_i$$

a deszyfrowanie równaniem

$$m_i = c_i \oplus (-k_i).$$

W przypadku gdy $m_i, c_i, k_i \in Z_2$ to wzór opisujący deszyfrowanie można zapisać jako $m_i = c_i \oplus k_i$.

Jeśli generator strumienia klucza (ang. keystream generator) jest generatorem generującym biały szum dyskretny to otrzymujemy idealny system kryptograficzny.

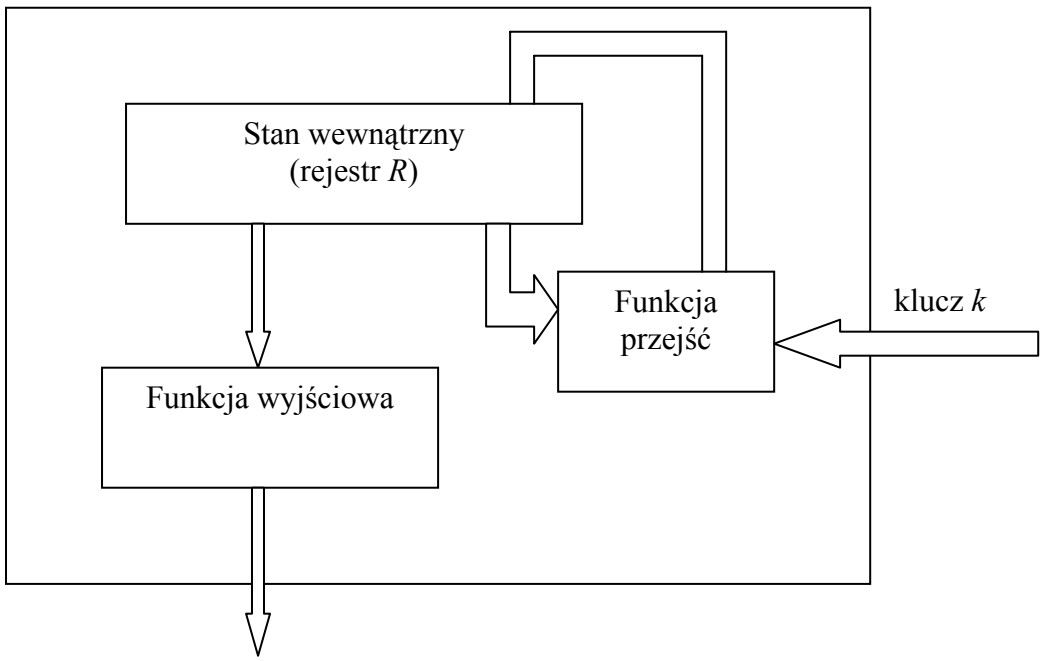
Typowy generator strumienia klucza w postaci automatu Moore'a. Inne rozwiązanie generatora strumienia klucza to maszyna liniowa (rejestr liniowy).

Analogiczną koncepcję szyfru strumieniowego otrzymamy dla tekstu jawnego $(m_i)_{i=1}^{N_0}$ o długości N_0, gdzie $m_i \in Z_n$ i strumienia klucza $(k_i)_{i=1}^{N_0}$, gdzie $k_i \in Z_n$. Szyfrowanie opisane jest wówczas równaniem

$$c_i = m_i \oplus_n k_i$$

gdzie $c_i, k_i \in Z_n$ a deszyfrowanie równaniem

$$m_i = c_i \oplus_n (-k_i).$$



Rys. 6. 4 Typowy generator strumienia klucza w postaci automatu Moore'a.

Literatura

[1] A. Menezes, P. Oorschot, S. Vanstone; Handbook of Applled Cryptography; CRC Press Inc., 1997. (treść jest na stronie: http://cacr.math.uwaterloo.ca/hac)

[2] J.Stokłosa, T.Bilski,T.Pankowski; Bezpieczeństwo danych w systemach informatycznych; PWN, Warszawa 2001.

[3] M.Kutyłowski, W.Strothmann; Kryptografia, teoria i praktyka zabezpieczania systemów komputerowych; Oficyna Wydawnicza Read Me, Warszawa 2001.

[4] B.Schneier; Kryptografia dla praktyków; WNT, 2002.

[5] W.Stallings; Ochrona danych w sieci i intersieci; WNT; Warszawa 1997.