# Wykład 7

**Temat: <u>Tryby działań symetrycznych algorytmów kryptograficznych</u>:**
**uwagi ogólne; tryby: elektronicznej książki kodowej – ECB, wiązania bloków zaszyfrowanych – CBC; szyfry strumieniowe; tryby: sprzężenia zwrotnego szyfrogramu – CFB, sprzężenia zwrotnego wyjściowego – OFB; tryb licznikowy; inne tryby działań symetrycznych szyfrów blokowych; zastosowania trybów pracy symetrycznych algorytmów kryptograficznych.**

## 7.1. Uwagi ogólne

Tryb pracy algorytmu łączy zwykle szyfr podstawowy, pewien rodzaj sprzężenia zwrotnego oraz proste operacje. Operacje są proste, ponieważ bezpieczeństwo algorytmu zależy od algorytmu a nie jego trybu pracy. Każdy z trybów pracy charakteryzuje się pewnymi cechami które determinują sposób jego wykorzystania. Taką cechą jest odporność na błędy. Z jednej strony pożądaną cechą może być możliwość odczytania tekstu jawnego, w przypadku wystąpienia błędów podczas transmisji (pojawią się przekłamania na pojedynczych pozycjach). W innych zastosowaniach taka cecha może się okazać wielką wadą, kompromitującą cały system. Wniosek jest prosty należy dobierać algorytm i tryb jego pracy zgodnie z wymaganiami systemu. Inne takie cechy to chociażby wydajność (czy długość szyfrogramu jest taka sama jak tekstu jawnego), możliwość wielokrotnego użycia do szyfrowania tego samego klucza.

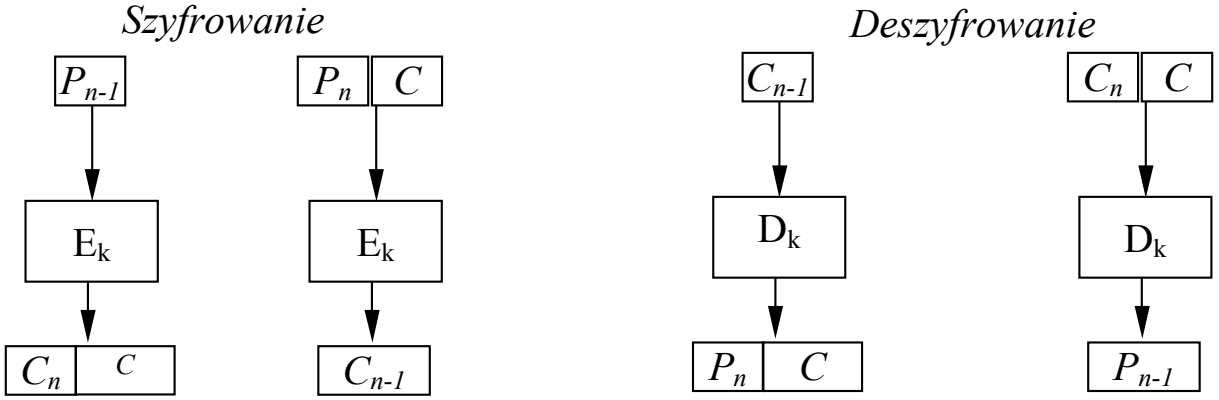
## 7.2. Tryb elektronicznej książki kodowej – ECB

Pierwszym omawianym trybem pracy symetrycznych algorytmów kryptograficznych będzie tryb elektronicznej książki kodowej *(Electronic Code Book – ECB)*. Tryb ten szyfruje każdy blok tekstu jawnego w blok szyfrogramu. Nazwa trybu odnosi się do teoretycznej możliwości utworzenia książki kodowej tekstu jawnego i odpowiadającego mu szyfrogramu wynikającej z takiej samej długości bloków przed i po zaszyfrowaniu (odszyfrowaniu). Jeśli długość bloku wynosi 64 bity, to istnieje $2^{64}$ możliwych wyrażeń wejściowych dla jednego klucza.

Tryb ten jest najłatwiejszy do zastosowania. Jego podstawową zaletą jest możliwość niezależnego szyfrowania każdego bloku wejściowego, z czego wynika możliwość szyfrowania bloków wejściowych bez zachowania kolejności. Cecha ta może być szczególnie przydatna dla plików, w których musi być zachowana możliwość dostępu losowego, na przykład w bazach danych. Jeśli baza danych będzie szyfrowana w trybie ECB to każdy jej rekord może być przetwarzany niezależnie od pozostałych, zakładając że każdy rekord to kilka bloków algorytmu symetrycznego. Przetwarzanie bloków można dodatkowo przyśpieszyć przez zastosowanie procesorów (urządzeń szyfrujących) pracujących równolegle. Każdy procesor będzie w takim przypadku pracował niezależnie, przetwarzając pewien podzbiór bloków wejściowych. Do zalet tego trybu pracy należy również zaliczyć zachowanie bezpieczeństwa podczas szyfrowania wielu wiadomości przy pomocy tego samego klucza. W przypadku wystąpienia błędu w szyfrogramie błędnie zostanie odszyfrowany tylko blok w którym pojawiło się przekłamanie, natomiast pozostałe bloki zostaną odszyfrowane poprawnie. Sprawa komplikuje się jeśli zostanie utracony lub dodany bit szyfrogramu, wtedy szyfrogram od miejsca wystąpienia błędu zostanie odszyfrowany błędnie. Ponieważ każdy blok przetwarzany jest niezależnie zatem wcześniejsze bloki zachowały właściwą budowę i zostaną poprawnie przetworzone. Zjawisku temu można przeciwdziałać poprzez mechanizmy zapewniające odtwarzanie granic bloków (np. wprowadzenie stałych struktur ramek).

Tryb ten posiada również pewne wady. Kryptoanalityk posiadając tekst jawny i szyfrogram dla kilku wiadomości może rozpocząć odtwarzanie książki kodowej bez znajomości klucza. Niebezpieczeństwo wynika również z tendencji do powtarzania fragmentów wiadomości czy też dużej nadmiarowości dla wiadomości wytwarzanych komputerowo. Ponieważ fragmenty te pojawiają się w różnych wiadomościach, to kryptoanalityk uzyskuje informacje umożliwiające rozpoczęcie ataku bazującego na analizie statystycznej, niezależnie od mocy szyfru blokowego. Podatność na tego typu ataki jest największa na początku i końcu wiadomości, gdzie istnieją ściśle zdefiniowane nagłówki i stopki nazywane niekiedy nagłówkami stereotypowymi i zakończeniami stereotypowymi.

Większość wiadomości nie dzieli się dokładnie na 64-bitowe (lub innego rozmiaru) bloki szyfrowania. Najczęściej okazuje się że ostatni blok jest krótszy, a przecież tryb ECB wymaga by wszystkie bloki były stałych rozmiarów. Aby poradzić sobie z tym problemem należy zastosować technikę dopełniania bloków. Dopełnianie polega na uzupełnieniu niepełnych bloków przez regularne wzorce – uzupełnianie zerami, jedynkami, na przemian zerami i jedynkami. Aby usunąć dopełnienie po odszyfrowaniu, należy podać liczbę pozycji dopełniania jako ostatni bajt wiadomości. Ostatni bajt informujący o dopełnieniu musi również pojawić się w sytuacji gdy ostatni blok jest pełny. W celu zaznaczenia ostatniego bajta tekstu jawnego można również użyć znaku końca pliku, a następnie wstawić dopełnienie po tym znaku. Na rysunku poniżej przedstawiono rozwiązanie alternatywne, zwane podkradaniem szyfrogramu. $\boldsymbol{P_{n-1}}$ jest ostatnim pełnym blokiem tekstu jawnego, a $\boldsymbol{P_n}$ jest końcowym i niepełnym blokiem tekstu jawnego. $\boldsymbol{C_{n-1}}$ jest ostatnim pełnym blokiem szyfrogramu, a $\boldsymbol{C_n}$ jest końcowym i niepełnym blokiem szyfrogramu. $\boldsymbol{C}$ jest wynikiem pośrednim i nie jest częścią przesyłanego szyfrogramu.



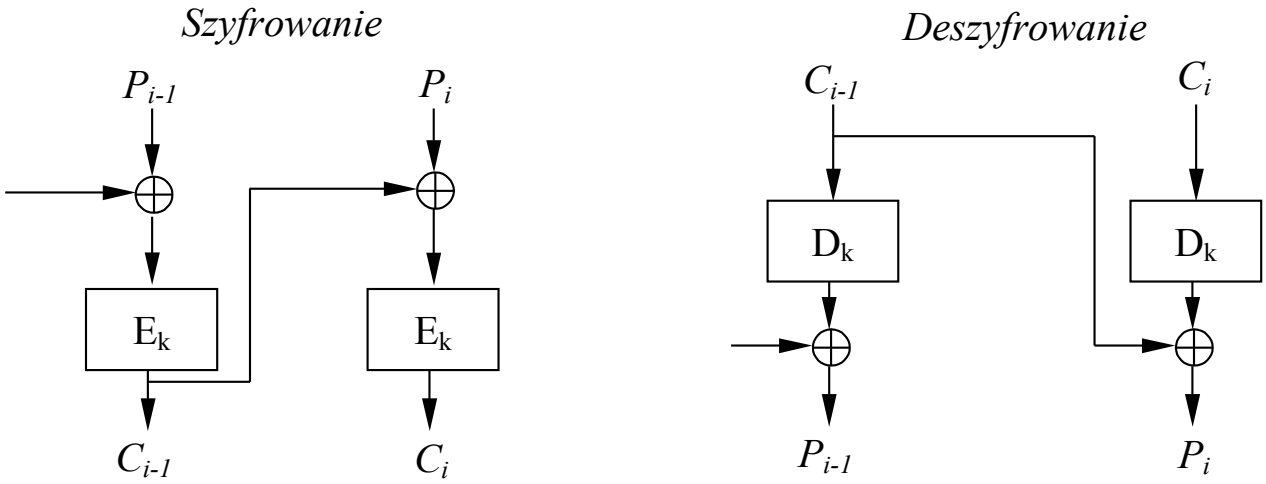

Rys. 7.1. Podkradanie szyfrogramu w trybie ECB

Największe zagrożenie przy stosowaniu trybu ECB wynika z niezależności bloków. Wcześniej wspomniano o zaletach tego rozwiązania (np. możliwość pracy równoległej) ale należy zwrócić uwagę na możliwość zmodyfikowania przez atakującego zaszyfrowanej wiadomości, nawet bez znajomości klucza. Atakujący by przeprowadzić udany atak tego typu musi poznać konstrukcję wiadomości. Poznawszy ją (np. przez podsłuchanie komunikacji) może zmodyfikować ją tak by odpowiadała jego zamierzeniom. Ta technika ataku nosi nazwę ***ataku z powtarzaniem bloku***. Oczywiście użytkownicy systemu mogą utrudnić zadanie atakującemu przez częstą zmianę klucza, ale to wymusi tylko na atakującemu szybszą pracę. Rozwiązaniem tych problemów może być dodanie skrótu MAC, ale

mimo to jest to podstawowy mankament szyfrów blokowych pracujących w trybie ECB. Napastnik nadal może usuwać, powtarzać lub zmieniać bloki miejscami.

### 7.3. Tryb wiązania bloków zaszyfrowanych – CBC

Tryb wiązania bloków wykorzystuje zjawisko sprzężenia zwrotnego. Wyniki szyfrowania poprzedniego bloku są danymi wejściowymi operacji szyfrowania bloku bieżącego. Zatem każdy blok szyfrogramu zależy nie tylko od bloku tekstu jawnego, ale od wszystkich poprzedzających go bloków tekstu jawnego. W trybie wiązania bloków zaszyfrowanych *(Cipher Block Chaining – CBC)* tekst jawny jest przed szyfrowaniem sumowany modulo 2 z poprzednimi blokami szyfrogramu. Na rysunku poniżej przedstawiono szyfrowanie z wiązaniem bloków. Po zaszyfrowaniu bloku tekstu jawnego w rejestrze sprzężenia zwrotnego znajduje się blok wynikowy. Zanim będzie szyfrowany kolejny blok tekstu jawnego, najpierw następuje sumowanie modulo 2 z zawartością rejestru sprzężenia zwrotnego. Wynik jest zarówno wartością szyfrogramu ale i kolejną wartością dla rejestru sprzężenia zwrotnego. Działania te są kontynuowane do końca wiadomości. Wynik szyfrowania każdego bloku jest zależny nie tylko od bloku reprezentującego tekst jawny ale i od poprzedzających go bloków.




Rys. 7.2. Procesy szyfrowania i deszyfrowania w trybie CBC

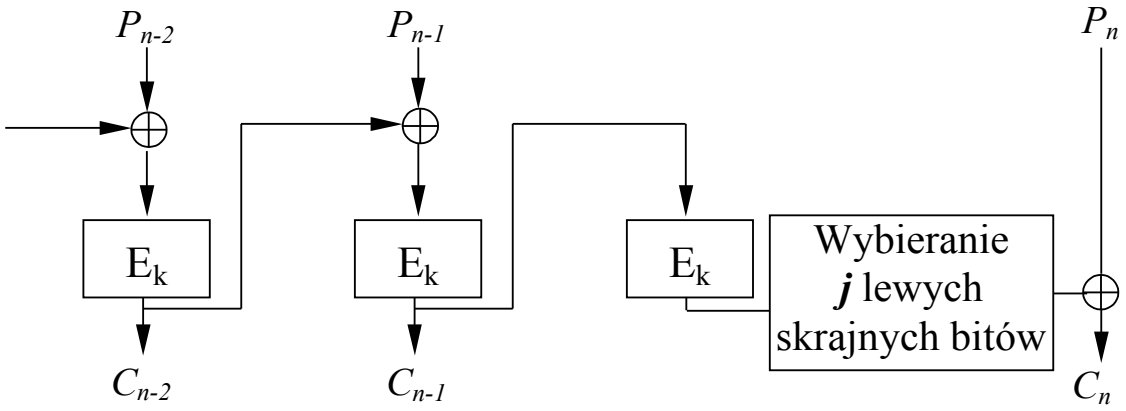
Odszyfrowanie przebiega w podobny sposób. Podobnie jak w procesie szyfrowania blok szyfrogramu po odszyfrowaniu jest zapamiętywany w rejestrze sprzężenia zwrotnego. Po odszyfrowaniu kolejnego bloku wynik jest sumowany modulo 2 z zawartością rejestru sprzężenia zwrotnego. Działania te są kontynuowane do wyczerpania bloków szyfrogramu. Matematycznie procesy szyfrowania i odszyfrowania można zapisać następująco:

$$C_i = E_K(P_i \oplus C_{i-1});$$

$$P_i = C_{i-1} \oplus D_K(C_i).$$

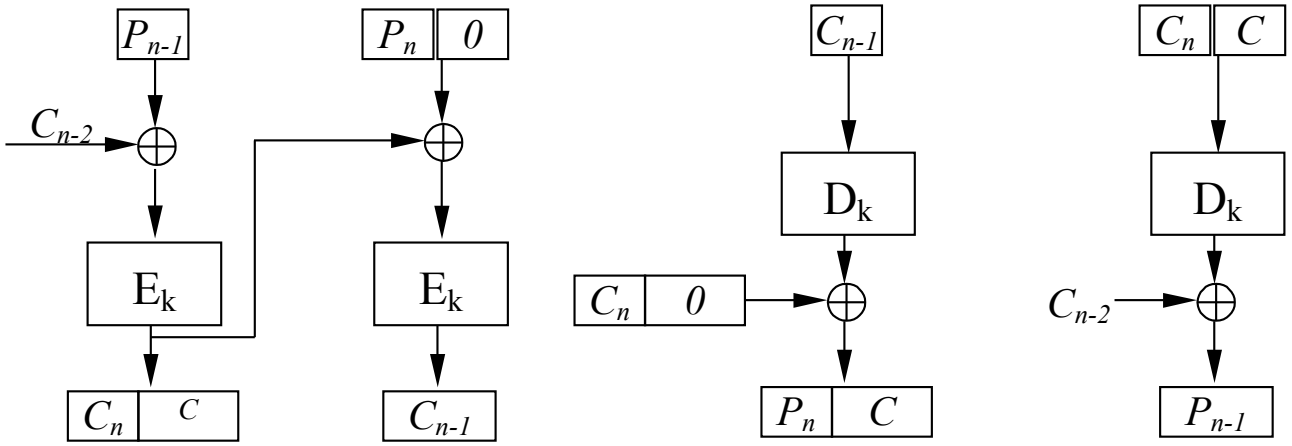
Oczywiście przy trybie CBC pojawiają się problemy które należy rozwiązać. Jeżeli do szyfrowania dwóch identycznych wiadomości użyjemy tego samego klucza to uzyskamy identyczny szyfrogram. Dwie wiadomości zaczynające się w identyczny sposób będą szyfrowane w taki sam sposób do momentu wystąpienia pierwszej różnicy. Ten problem dotyka przede wszystkim wiadomości charakteryzujących się wspólnymi nagłówkami (np. nagłówki dokumentów tekstowych, czy też format poczty elektronicznej). Te identyczne nagłówki umożliwią kryptoanalitykowi pozyskanie użytecznych informacji, które ułatwią mu atak. Aby uniknąć tych problemów można losowo wybierać dane które będą szyfrowane jako pierwsze. Ten blok danych losowych określa się jako wektor początkowy, zmienną początkującą lub początkową wartością związania. Wartość wektora początkowego nie ma znaczenia ponieważ nie reprezentuje żadnych użytecznych informacji, a ma za zadanie jedynie zapewnić unikalność każdej informacji. Jako wektor początkowy można użyć znacznika czasu, lub inne ciągi losowe. W procesie deszyfrowania wartość ta jest używana jako wartość początkowa rejestru sprzężenia zwrotnego. Dzięki zastosowaniu wektora początkowego zyskujemy pewność (pod warunkiem że wektor początkowy jest wartością niepowtarzalną) że każdy szyfrogram, nawet dla identycznego tekstu jawnego będzie wartością unikalną. Dodatkową zaletą tego rozwiązanie jest brak konieczności utrzymania wektora początkowego w tajemnicy, może on być nawet jawnie przesyłany z szyfrogramem.

Problem długości ostatniego bloku jest rozwiązywany po przez jego dopełnianie podobnie jak w trybie ECB. Jednak tryb CBC udostępnia nam metody które umożliwiają uzyskanie szyfrogramu o takiej samej długości jak tekst jawny. Jeżeli potrzebujemy szyfrogramu o długości równej długości tekstu jawnego to ostatni, niepełny blok musimy zaszyfrować w inny sposób. Przyjmijmy, że ostatni blok ma ***j*** bitów. Po zaszyfrowaniu ostatniego pełnego bloku ponówmy szyfrowanie, wybierzmy ***j*** najbardziej znaczących bitów z lewej strony i dodajmy je modulo 2 z niepełnym blokiem w celu uzyskania szyfrogramu. Na rysunku poniżej przedstawiono taki sposób szyfrowania ostatniego, niepełnego bloku. Metoda ta umożliwia atakującemu modyfikowanie ostatnich bitów szyfrogramu, co może okazać się szczególnie uciążliwe w sytuacji gdy modyfikowane bity zawierają ważne informacje.




Rys. 7.3. Szyfrowanie ostatniego bloku w trybie CBC

Problemów tych można uniknąć stosując metodę z podkradaniem szyfrogramu, której zaletą jest to, że wszystkie bity tekstu jawnego są poddane działaniu algorytmu szyfrującego. Schematyczny przebieg procesu szyfrowania w trybie CBC z wykorzystaniem metody z podkradaniem szyfrogramu przedstawiono na rysunku poniżej. $\boldsymbol{P_{n-1}}$ jest ostatnim, pełnym blokiem tekstu jawnego, a $\boldsymbol{P_n}$ jest końcowym, niepełnym blokiem tekstu jawnego. $\boldsymbol{C_{n-1}}$ jest ostatnim pełnym blokiem szyfrogramu, a $\boldsymbol{C_n}$ jest końcowym, krótkim i niepełnym blokiem szyfrogramu. $\boldsymbol{C}$ jest wynikiem pośrednim.




Rys. 7.4. Podkradanie szyfrogramu w trybie CBC

Tryb CBC, bazuje na zjawisku sprzężenia zwrotnego. W przypadku wystąpienia błędu w bloku tekstu jawnego, błąd ten będzie miał odzwierciedlenie w bloku szyfrogramu odpowiadającemu temu blokowi oraz w wszystkich kolejnych blokach szyfrogramu. Jednak nie ma to większego znaczenia bowiem w procesie odszyfrowania efekt ten zostanie odwrócony, a odtworzony tekst jawny będzie zawierał tylko pierwotny błąd. Bardziej powszechne są jednak błędy w szyfrogramie. W trybie CBC jeden błąd w szyfrogramie zmienia jeden blok i jeden bit odtworzonego tekstu jawnego. Blok w którym pojawia się przekłamanie jest całkowicie zniekształcony, natomiast następny blok ma jednobitowy błąd na tej samej pozycji co błąd w bloku zniekształconym. Kolejne bloki nie zawierają błędów, więc tryb CBC jest samoodtwarzający. Tryb CBC jest trybem samosynchronizującym, aczkolwiek synchronizacja jest obecna na poziomie bloków. Bardziej niebezpieczne niż przekłamanie bitowe w szyfrogramie jest pojawienie się dodatkowego bitu lub utrata bitu z strumienia szyfrogramu. Jeżeli wystąpi taka sytuacja to następuje przesunięcie granic bloków, a operacja szyfrowania będzie wytwarzać śmieci. Aby temu zapobiegać system szyfrowania pracujący w trybie CBC musi zapewniać niezmienność struktury bloków albo przez ujęcie w ramki, albo przez przechowywanie danych w porcjach wieloblokowych.

Atakujący system kryptograficzny pracujący w trybie CBC może dodawać kolejne bloki na końcu szyfrogramu a działania te nie będą zauważane przez użytkownika. W pewnych sytuacjach działania takie mogą być niepożądane i aby temu zapobiegać należy nadawać tekstowi jawnemu pewną strukturę taką, aby wiedzieć, w którym miejscu wiadomość kończy się, i móc wówczas wykryć dodatkowe bloki. Atakujący może również bazować w swoim ataku na fakcie że zmiana bitu szyfrogramu powoduje zmianę jednego bitu w bloku kolejnym po całkowicie zniekształconym. Aby tego uniknąć należy całą wiadomość uzupełniać nadmiarowością lub uwierzytelnianiem. Istnieje jeszcze jedna właściwość trybu CBC na który należy zwrócić uwagę. Mimo że cechy tekstu jawnego są ukrywane w trybie CBC, to jednak bardzo długie wiadomości nadal będą zawierać pewne wzorce. Na postawie paradoksu dnia urodzin można przewidzieć, że pojawią się identyczne bloki po $2^{m/2}$ blokach, przy czym m jest rozmiarem bloku. Dla bloku 64-bitowego jest to ok. 34GB, tak więc wiadomość musi być rzeczywiście długa, aby stało się to problemem.

### 7.4. Szyfry strumieniowe

Symetryczne systemy kryptograficzne są realizowane z wykorzystaniem dwóch typów szyfrów. Te dwa typy to szyfry blokowe i strumieniowe. Szyfry strumieniowe przekształcają tekst jawny w szyfrogram kolejno bit po bicie. Najprostsza implementacja szyfru została przedstawiona na rysunku poniżej. Generator strumienia klucza wytwarza strumień bitów: $k_1$, $k_2$, $k_3$, ... $k_i$. Aby wytworzyć szyfrogram należy wygenerowany strumień klucza zsumować modulo 2 z ciągiem bitów tekstu jawnego $p_1$, $p_2$, $p_3$, ... $p_i$:

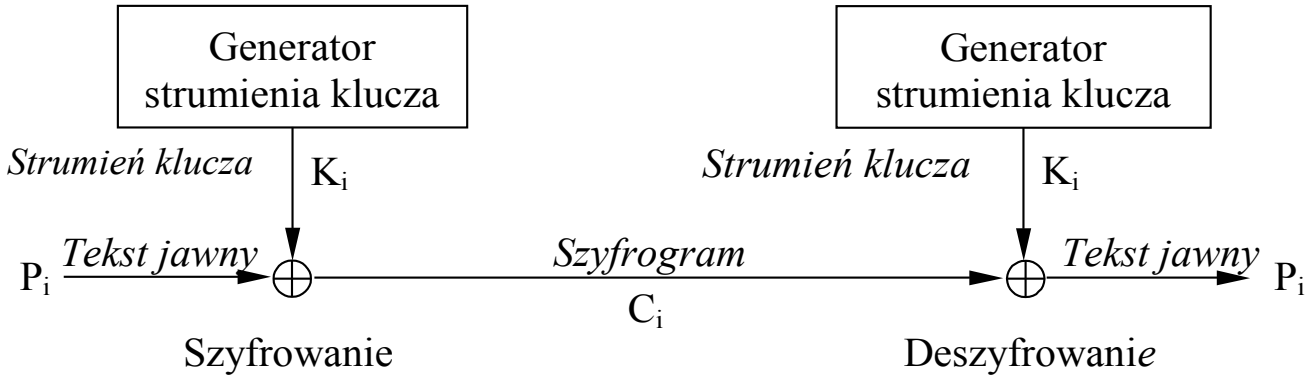
$$c_i = p_i \oplus k_i.$$

Aby odszyfrować uzyskany szyfrogram $c_i$ należy zsumować go modulo 2 z identycznym strumieniem klucza jak podczas procesu szyfrowania. W ten sposób zostanie odtworzony tekst jawny:

$$p_i = c_i \oplus k_i.$$

Wzór ten jest prawdziwy bowiem:

$$p_i \oplus k_i \oplus k_i = p_i$$

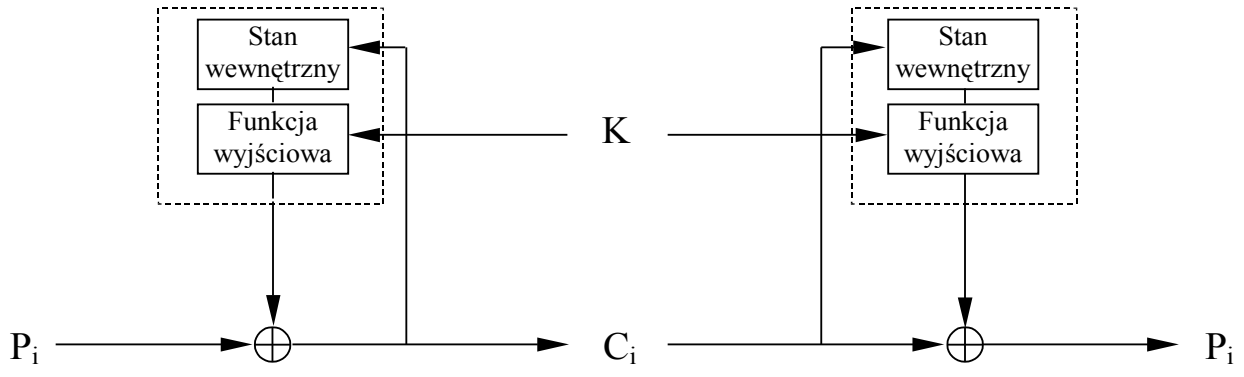



Rys. 7.5. Szyfr strumieniowy

Bezpieczeństwo systemów opartych na tego typu szyfrach strumieniowych zależy wyłącznie od wewnętrznych właściwości generatora strumienia klucza. Przykładowo jeżeli generator strumienia klucza wytwarza nieskończony ciąg zer, to szyfrogram będzie równy tekstowi jawnemu, a cała operacja nie będzie miała sensu. Oczywiście jest to sytuacja krytyczna i w rzeczywistych warunkach jest mało prawdopodobna, aczkolwiek nie można jej wykluczyć. Jeżeli generator wytwarza powtarzający się wzorzec 16-bitowy, to algorytm będzie zwykłym sumatorem modulo 2 z bardzo małym stopniem zabezpieczenia. Idealnym rozwiązaniem jest stworzenie generatora wytwarzającego nieskończony strumień bitów losowych. Otrzymujemy w ten sposób klucze jednorazowe i doskonałe zabezpieczenie. Jednak w rzeczywistości generator wytwarza ciąg bitów zdeterminowanych. Poziom bezpieczeństwa systemu będzie tym większy im bardziej „losowy" (a właściwie zbliżony do losowego) będzie generowany strumień klucza.

Istnieje również inne niebezpieczeństwo które może kompromitować cały system. Jeżeli generator strumienia klucza wytwarza za każdym, kiedy zostanie włączony, ten sam ciąg bitów, to złamanie takiego systemu będzie bardzo proste. Wystarczy, że atakujący pozyska tekst jawny oraz odpowiadający mu szyfrogram, to po przez zsumowanie modulo 2 tych strumieni bitów pozyska ciąg wyjściowy generatora klucza. Ponieważ każda nowa wiadomość będzie szyfrowana przy pomocy tego samego strumienia klucza to atakujący, posiadający strumień klucza ma zapewniony dostęp do całych zasobów chroniony przez system. Aby zapobiec takiej sytuacji stosuje się dodatkowe klucze. W takim przypadku ciąg wyjściowy generatora strumienia klucza jest funkcją klucza. Każda zmiana klucza powoduję zmianę strumienia wyjściowego generatora uniemożliwiając przeprowadzenie ataku na system. Oczywiście jeżeli klucz wejściowy generatora nie będzie zmieniany to nadal istnieje niebezpieczeństwo ataku na system po przez odtworzenie klucza, bazując na odpowiadającej sobie parze tekst jawny – szyfrogram. Szyfry strumieniowe są szczególnie użyteczne w szyfrowaniu nigdy niekończącego się strumienia informacji (np. łączy teletransmisyjnych).

Na rozwiązaniu z dodatkowym kluczem bazują samosynchronizujące szyfry strumieniowe zwane też szyframi z szyfrogramem samokluczującym. Dodatkowo, każdy bit ciągu klucza jest funkcją pewnej stałej liczby poprzednich bitów szyfrogramu. Na rysunku poniżej przedstawiono schemat działania samosynchronizującego szyfru strumieniowego. Stan wewnętrzny generatora strumienia klucza jest funkcją ***n*** poprzednich bitów szyfrogramu. Złożoność kryptograficzna zawiera się w funkcji wyjściowej, która pobiera stan wewnętrzny i wytwarza bit strumienia klucza. Ponieważ stan wewnętrzny zależy od ***n*** poprzednich bitów szyfrogramu, generator strumienia klucza wykorzystywany w procesie deszyfrowania będzie automatycznie zsynchronizowany z generatorem strumienia klucza do szyfrowania po odebraniu ***n*** bitów szyfrogramu. Aby zsynchronizować nadajnik z odbiornikiem, przesyłaną wiadomość poprzedza się losowym nagłówkiem o długości ***n*** bitów. Nagłówek ten jest szyfrowany, przesyłany a następnie deszyfrowany po stronie odbiorczej. Deszyfrowanie będzie nieprawidłowe dla n początkowych bitów (nagłówek który nie zawiera żadnych użytecznych informacji), po czym nastąpi synchronizacja nadajnika z odbiornikiem. Problem przy stosowaniu samosynchronizujących szyfrów strumieniowych jest propagacja błędów. Każde przekłamanie na jednym bicie w szyfrogramie powoduje po stronie odbiorczej błędne wytworzenie n bitów strumienia klucza. Zatem dla każdego błędu w szyfrogramie wystąpi ***n*** odpowiadających mu błędów w tekście jawnym do czasu, aż zniekształcone bity opuszczą rejestr stanu wewnętrznego generatora strumienia klucza. Samosynchronizujące szyfry strumieniowe są również podatne na ataki powtórzeniowe, aby zminimalizować tego typu niebezpieczeństwa należy wykorzystać znaczniki czasu.



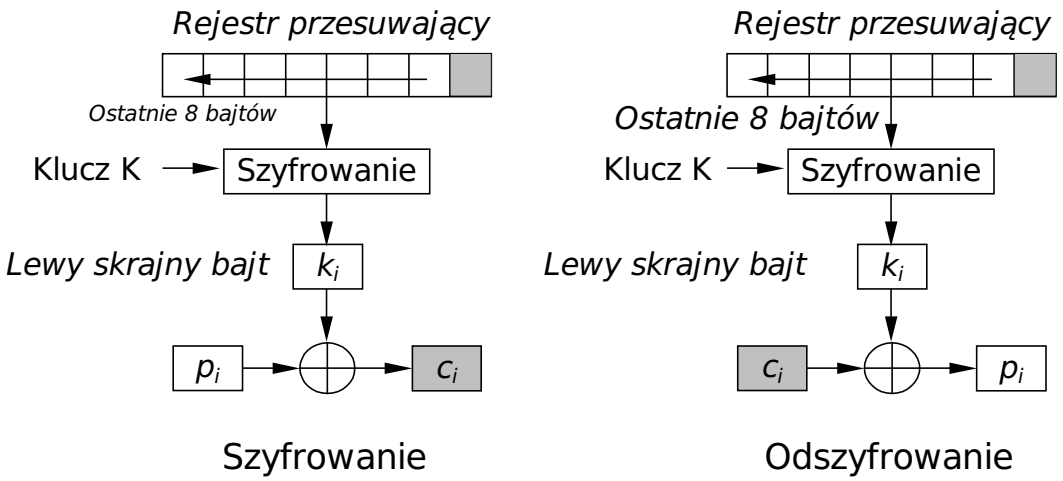

Rys. 7.6. Generator strumienia klucza samosynchronizującego szyfru strumieniowego

Jeżeli klucz jest generowany niezależnie od strumienia wiadomości to otrzymujemy synchroniczny szyfr strumieniowy zwany też algorytmem z kluczem samokluczującym. W takim przypadku generatory strumienia klucza zarówno po stronie nadawczej jak i po stronie odbiorczej muszą generować identyczne strumienie klucza. Taki system działa poprawnie do momentu utraty synchronizacji pomiędzy nadajnikiem a odbiornikiem. Przyczyną takiego stanu może być utrata bitu podczas transmisji szyfrogramu, albo przekłamanie bitowe na jednym z generatorów (generatory nie wytwarzają identycznych strumieni klucza). Jeżeli wystąpi jedna z powyższych sytuacji to od momentu wystąpienia błędu każdy znak szyfrogramu będzie odszyfrowany nieprawidłowo. Aby powrócić do prawidłowej pracy systemu należy ponownie zsynchronizować nadajnik z odbiornikiem. Jednak działanie to nie jest proste ponieważ proces synchronizacji należy przeprowadzić w taki sposób aby nie powtórzyć żadnej części strumienia klucza. Zatem wymaganie takie wyklucza synchronizacje po przez ponowne inicjowanie obu generatorów. Zaletą synchronicznych szyfrów strumieniowych jest brak propagacji błędów w przypadku przekłamania w szyfrogramie. Wszystkie zarówno poprzedzające jak następujące po nim bity zostaną odszyfrowane prawidłowo. Generatory używane w systemach pracujący w oparciu o synchroniczne szyfry strumieniowe muszą być zdeterminowane (identyczne strumienie klucza po stronie nadawczej jak i odbiorczej). Takie generatory noszą nazwę okresowych, ponieważ ciągi bitów są powtarzalne. Jednak nie stanowi to problemu jeżeli długość okresu generatora jest o wiele dłuższa od liczby bitów wytwarzanych przez ten generator między zmianami klucza. Jeżeli długość okresu jest mniejsza od długości tekstu jawnego, to pewne części tekstu jawnego będą szyfrowane w taki sam sposób, a to stanowi już poważną wadę. Można przyjąć że długość okresu powinna być o kilka rzędów większa niż zakłada ilość bitów poddawanych szyfrowaniu pomiędzy zmianami klucza. Synchroniczne szyfry strumieniowe uniemożliwiają dokonywanie manipulacji szyfrogramem polegających na dodawaniu lub wycinaniu bitów z szyfrogramu, ponieważ takie działanie powoduje utratę synchronizacji i będzie natychmiast wykryte. Atakujący może jednak dokonywać zmian kolejności bitów w szyfrogramie, a działania takie nie zostaną wykryte. Synchroniczne szyfry strumieniowe są podatne na atak wstawieniowy. Jednak jeżeli nie będziemy używać tego samego ciągu klucza do szyfrowania różnych wiadomości możemy zapewnić ochronę przed tego typu atakiem.

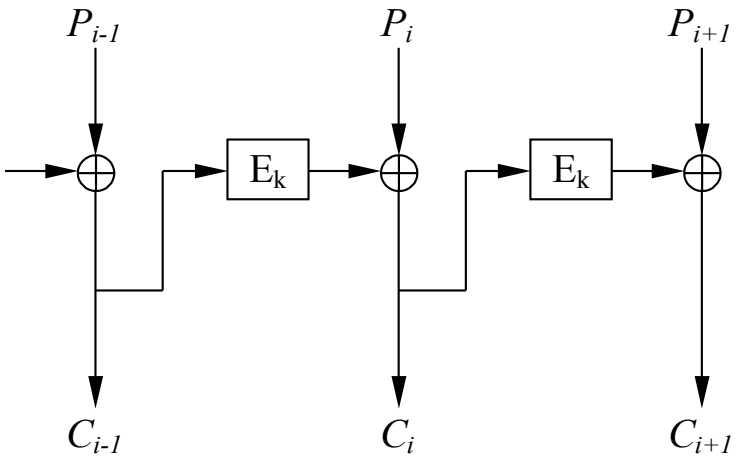
### 7.5. Tryb sprzężenia zwrotnego szyfrogramu – CFB

Szyfry blokowe mogą być również zaimplementowane jako samosynchronizujące szyfry strumieniowe. Taki tryb pracy nazywa się trybem sprzężenia zwrotnego szyfrogramu *(Cipher Feedback)*. Tryb CFB eliminuje konieczność posiadania tekstu jawnego o długości jednego bloku by rozpocząć proces szyfrowania lub deszyfrowania. Właściwość ta jest szczególnie użyteczna w chronionym środowisku sieciowym gdzie terminal musi przekazywać do komputera głównego każdy znak zaraz po jego wprowadzeniu (tryb znakowy). W takim przypadku tryb CBC okazuje się niemożliwy do zastosowania. Natomiast tryb CFB spełnia te wymagania. Tryb CFB umożliwia szyfrowanie danych w jednostkach mniejszych niż rozmiar bloku. W sytuacji środowiska sieciowego przedstawionej powyżej należy szyfrować jeden znak na raz (tzw. 8-bitowy CFB). Długość jednostki używanej do szyfrowania zawiera się w przedziale od ***1*** bitu do wartości odpowiadającej długości bloku.

Na rys. 7.7 przedstawiono schemat działania trybu 8-bitowego CFB z algorytmem 64-bitowym. Algorytm w trybie CFB operuje na kolejce (rejestrze) o rozmiarze bloku wejściowego (na rysunku 64 bity). W pierwszej fazie rejestr wypełniono wartością początkową (wektor początkowy). Po zaszyfrowaniu zawartości rejestru 8 bitów skrajnych z lewej strony dodaje się modulo 2 z pierwszy 8-bitowym znakiem tekstu jawnego w celu uzyskania pierwszego znaku 8-bitowego szyfrogramu. Ten znak może być następnie przesłany. Te same 8 bitów podaje się na 8 bitów skrajnych z prawej strony kolejki, a wszystkie pozostałe bity są przesuwane o 8 pozycji w lewo. Odrzuca się najbardziej znaczące 8 bitów. Następny znak szyfruje się dokładnie w ten sam sposób. Odszyfrowanie jest odwrotnością tego procesu. Po obu stronach, szyfrowania i odszyfrowania, algorytm blokowy stosuje się w trybie szyfrującym.




Rys. 7.7. Tryb 8-bitowego sprzężenia zwrotnego szyfrogramu CFB




Rys. 7.8. Tryb *n*-bitowego CFB z użyciem algorytmu blokowego *n*-bitowego

Rysunek powyżej przedstawia tryb ***n***-bitowego CFB dla algorytmu o bloku długości ***n*** bitów. Wtedy proces szyfrowania i deszyfrowania możemy zapisać w następujący sposób:

$$C_i = P_i \oplus E_K(C_{i-1});$$

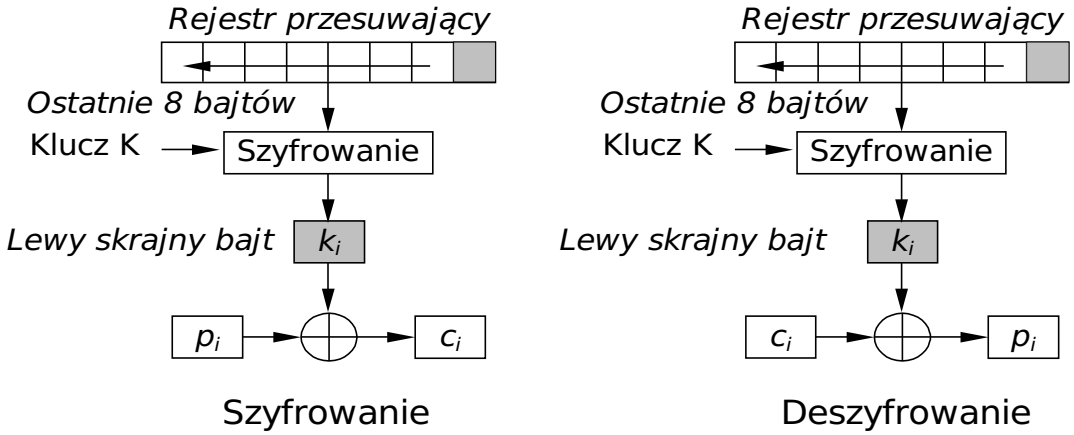
$$P_i = C_i \oplus E_K(C_{i-1}).$$

Algorytm w trybie CFB podobnie jak algorytm w trybie CBC łączy znaki tekstu jawnego w ten sposób, że szyfrogram zależy od całego poprzedzającego tekstu jawnego. W celu zainicjowania procesu CFB rejestr przesuwający musi zostać wypełniony wartością początkową (wektor początkowy). Wektor ten musi być niepowtarzalny, w przeciw przypadku kryptoanalityk może odtworzyć związany z nim tekst jawny.

W trybie CFB błąd występujący w tekście jawnym wpływa na cały szyfrogram i powraca na swoje miejsce po odszyfrowaniu. W przypadku błędu w szyfrogramie pojawia się jeden błąd w tekście jawnym. Następnie błąd ten jest propagowany do rejestru przesuwającego, powodując zniekształcenia szyfrogramu tak długo, aż przesunie się do drugiego końca rejestru. W 8-bitowym trybie CFB jeden błąd zniekształca łącznie 9 bajtów szyfrogramu. W ogólnym przypadku ***n***-bitowy błąd w szyfrogramie spowoduje błędne odszyfrowanie bloku bieżącego i następnych ***m/n-1*** bloków, przy czym ***m*** jest rozmiarem bloku. Algorytm pracujący w trybie CFB jest algorytmem samo odtwarzającym ze względu na błędy synchronizacji. Błąd wnika do rejestru przesuwającego, w którym zniekształca 8 bajtów danych do chwili, aż opuści rejestr z drugiej strony. Tryb CFB jest przykładem zastosowania szyfru blokowego jako samosynchronizującego szyfru strumieniowego (na poziomie bloku).

### 7.6. Tryb sprzężenia zwrotnego wyjściowego – OFB

Tryb sprzężenia zwrotnego wyjściowego *(Output Feedback)* jest metodą działania szyfru blokowego jako synchronicznego szyfru strumieniowego. Działanie algorytmów w trybie OFB jest bardzo podobne do pracy w trybie CFB. Podstawowa różnica między oboma trybami polega na tym, że w trybie OFB ***n*** bitów poprzedniego bloku wyjściowego jest przesuwanych na skrajne prawe pozycje kolejki. Odszyfrowanie jest odwrotnością tego procesu. Takie działanie określa się ***n***-bitowym OFB. Po obu stronach, szyfrowania i odszyfrowania, algorytm blokowy jest stosowany w swoim trybie szyfrującym. Metoda ta nosi nazwę trybu wewnętrznego sprzężenia zwrotnego, ponieważ mechanizm sprzężenia zwrotnego jest niezależny zarówno od ciągu tekstu jawnego, jak i ciągu szyfrogramu. Na rysunku poniżej przedstawiono tryb 8-bitowego sprzężenia zwrotnego wyjściowego OFB.




Rys. 7.9. Tryb 8-bitowego sprzężenia zwrotnego wyjściowego OFB

Jeśli rozmiar bloku algorytmu wynosi ***n***, to ***n***-bitowy tryb OFB możemy opisać w następujący sposób:

$$C_i = P_i \oplus S_i,$$

$$P_i = C_i \oplus S_i,$$

$$S_i = E_K(S_{i-1}),$$

gdzie $S_i$ jest stanem niezależnym od tekstu jawnego i szyfrogramu.

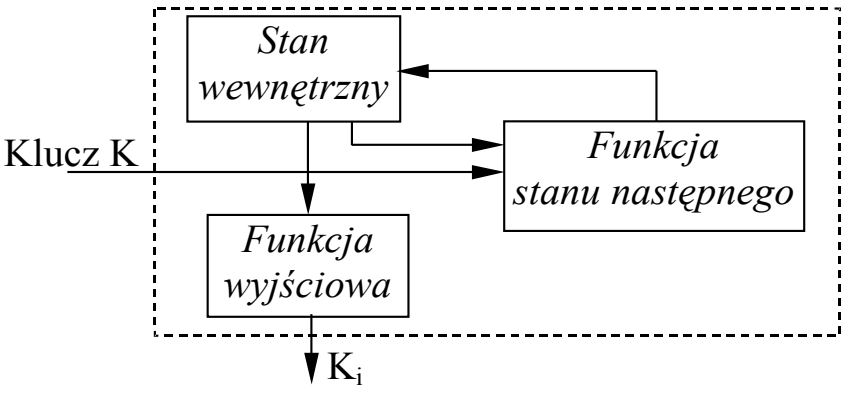
Zaletą pracy w trybie OFB jest to, że większość pracy obliczeniowej może być wykonana w trybie off-line, zanim powstanie tekst jawny. W momencie pojawienia się wiadomości przeznaczonej do szyfrowania może ona być zsumowana modulo 2 z ciągiem wyjściowym algorytmu w celu utworzenia szyfrogramu. W trybie OFB rejestr przesuwający musi być inicjowany niepowtarzalnym wektorem początkowym, aczkolwiek wartość ta nie musi być tajna. Zaletą trybu OFB jest nie rozprzestrzenianie się błędów w szyfrogramie. Błąd bitowy w szyfrogramie uwidacznia się błędnym odtworzeniem jednego bitu tekstu jawnego. Problemem jest natomiast utrata synchronizacji. Jeżeli wartości rejestrów przesuwających po stronie nadawczej i odbiorczej nie są identyczne to tekst jawny zostanie odtworzony nieprawidłowo. Zatem należy stosować mechanizmy wykrywania utraty synchronizacji i narzędzia umożliwiające prawidłowe odtworzenie zawartości rejestrów przesuwających w celu odzyskania synchronizacji. Analiza trybu OFB wykazała że powinien on być stosowany wtedy, gdy rozmiar sprzężenia zwrotnego jest taki sam jak rozmiar bloku (algorytm 64-bitowy wymaga 64 bitowego OFB itp.). W trybie OFB strumień klucza jest dodawany modulo 2 z tekstem przeznaczonym do szyfrowania. Strumień klucza może się ewentualnie powtórzyć, ale nie powinno to wystąpić dla tego samego klucza. W takim przypadku system jest właściwie pozbawiony ochrony. Gdy rozmiar sprzężenia zwrotnego równa się rozmiarowi bloku, wówczas szyfr blokowy dokonuje permutacji wartości ***m***-bitowych (przy czym ***m*** jest długością bloku), a średnia długość cyklu wynosi $2^m - 1$. Dla bloku 64-bitowego jest to bardzo długa liczba. Gdy rozmiar sprzężenia zwrotnego n jest mniejszy od długości bloku, wówczas średnia długość cyklu zmniejsza się do ok. $2^{m/2}$. Dla 64-bitowego szyfru blokowego jest to wartość zaledwie $2^{32}$ – zbyt mała.

Również szyfry strumieniowe mogą pracować w trybie OFB. Wtedy klucz wpływa na działanie funkcji następnego stanu. Funkcja wyjściowa nie zależy od klucza. Złożoność kryptograficzna spoczywa w funkcji następnego stanu, zatem zależy również od klucza. Metoda ta jest zwana trybem ze sprzężeniem wewnętrznym, ponieważ mechanizm sprzężenia ma charakter wewnętrzny w stosunku do algorytmu generowania klucza.

### 7.6. Tryb licznikowy

Tryb licznikowy rozwiązuje problem występujący w trybie OFB – zbyt krótkich ciągów wyjściowych w porównaniu z długością bloku. Natomiast właściwości tego trybu w zakresie synchronizacji i propagacji błędów są identyczne jak w trybie OFB. Szyfry te wykorzystują sekwencję liczb jako ciąg wejściowy algorytmu. W trybie licznikowym wartością wejściową rejestru zamiast danych wyjściowych algorytmu szyfrowania są dane licznika. Po każdym bloku szyfrującym licznik zwiększa się o pewną stałą wartość, typowo ***1***.

Szyfry strumieniowe mogą być również zaimplementowane w trybie licznikowym. Charakteryzują się prostymi funkcjami stanu wewnętrznego i skomplikowanymi funkcjami wyjściowymi zależnie od klucza. Tę technikę pokazano na rysunku poniżej. Funkcja następnego stanu jest zazwyczaj prosta np. licznik zwiększający wartość o ***1*** względem swego poprzedniego stanu. W przypadku szyfru strumieniowego w trybie licznikowym jest możliwe generowanie ***i***-tego bitu klucza $k_i$ bez konieczności wytworzenia wszystkich poprzednich bitów klucza. Po prostu należy ustawić licznik ręcznie na ***i***-ty stan wewnętrzny i wygenerować pożądany bit. Ta metoda jest użyteczna do zabezpieczania plików z danymi o dostępie swobodnym – można odszyfrować wybrany blok danych bez konieczności odszyfrowania całego pliku.




Rys. 7.10. Generator strumienia klucza w trybie sprzężenia zwrotnego licznikowego

## 2.8. Inne tryby działań symetrycznych szyfrów blokowych

Oprócz wcześniej omówionych głównych, najczęściej używanych trybów pracy algorytmów blokowych istnieją również inne tryby mniej lub bardziej użyteczne. Najczęściej stanowią one modyfikacje wcześniej przedstawionych trybów. W tej części wykładu zostaną właśnie one przedstawione.

Pierwszym omawianym trybem będzie tryb wiązania blokowego *(block chaining) – BC*. W tym trybie wyznacza się sumę modulo 2 ciągu wejściowego szyfru blokowego ze wszystkimi poprzednimi blokami szyfrogramu. Podobnie jak np. w trybie CBC, proces rozpoczyna się od przypisania do rejestru wartości określonej wektorem początkowym. Proces ten można zapisać w następujący sposób:

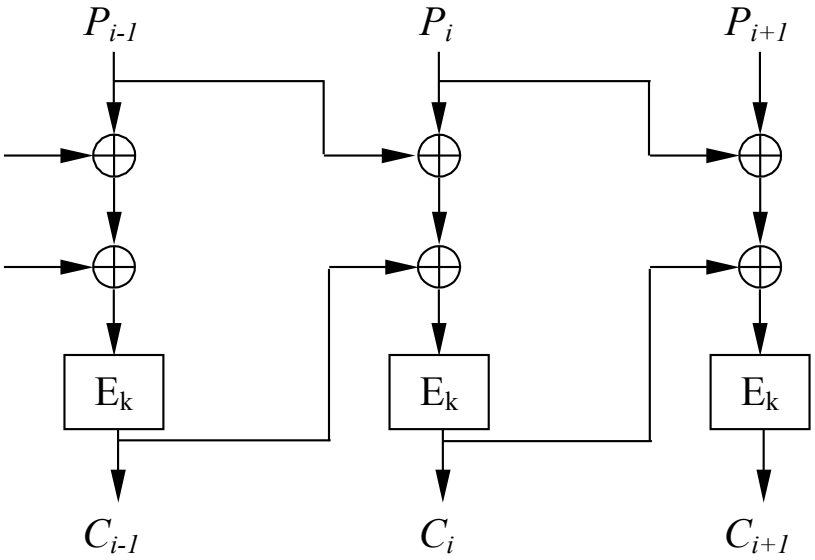
$$C_i = E_K(P_i \oplus F_i);$$
$$P_i = F_i \oplus D_K(C_i);$$
$$F_{i+1} = F_i \oplus C_i.$$

Podstawowym problemem związanym z trybem BC jest rozprzestrzenianie się błędów w tekście jawnym. W przypadku przekłamania w szyfrogramie błędnemu odszyfrowaniu podlegać będą wszystkie kolejne bloki szyfrogramu od momentu wystąpienia błędu. Wynika to z tego że odszyfrowanie bloku szyfrogramu zależy od wszystkich poprzednich bloków szyfrogramu.

Podobny do CBC jest również tryb propagującego wiązania bloków zaszyfrowanych *(propagating cipher block chaining) – PCBC*. Różnica między oboma trybami wynika z tego, że tryb PCBC sumuje modulo 2 aktualny blok tekstu jawnego z poprzedzającym go blokiem tekstu jawnego czy też szyfrogramu przed lub po szyfrowaniu. Proces ten można zapisać:

$$C_i = E_K(P_i \oplus C_{i-1} \oplus P_{i-1});$$
$$P_i = C_{i-1} \oplus P_{i-1} \oplus D_K(C_i).$$

Tryb PCBC był zastosowany w wersji 4 algorytmu Kerberos do realizacji szyfrowania i sprawdzania spójności w jednym przebiegu algorytmu. W trybie PCBC pojawienie się błędu w szyfrogramie spowoduje niepoprawne odszyfrowanie wszystkich następujących po nim bloków. Oznacza to, że sprawdzenie bloku standardowego przy końcu wiadomości zapewni spójność całej wiadomości. Niestety pojawia się pewien problem przy pracy w tym trybie. Zmiana miejscami dwóch bloków szyfrogramu spowoduje niepoprawne odszyfrowanie dwóch odpowiadających im bloków tekstu jawnego. Ale błędy te będą wyeliminowane ze względu na naturę sumowania modulo 2 tekstu jawnego i szyfrogramu. Tak więc, jeśli w module sprawdzającym spójność następuje przeglądanie jedynie kilku ostatnich bloków odszyfrowanego tekstu jawnego, to może pojawić się błąd polegający na tym, że zostanie zaakceptowana częściowo zaśmiecona wiadomość. Omawiany tryb z tego powodu w wersji 5 systemu Kerberos został zastąpiony trybem CBC.




Rys. 7.11. Tryb propagujący wiązania bloków zaszyfrowanych PCBC

Wariantem trybu CBC jest tryb wiązania bloków zaszyfrowanych z sumą kontrolną *(cipher block chaining with checksum) – CBCC*. Obliczaną na bieżąco sumę modulo 2 wszystkich kolejnych bloków tekstu jawnego dodaje się do ostatniego bloku tekstu jawnego przed zaszyfrowaniem. Zmiana w dowolnym bloku szyfrogramu powoduje zmianę odszyfrowanego ciągu wyjściowego ostatniego bloku. Jeśli ostatni blok zawiera dowolny rodzaj kontroli spójności lub stałą, to można sprawdzić spójność odszyfrowanego tekstu jawnego, używając małego nakładu pracy.

Odmianą trybów OFB i ECB jest tryb sprzężenia zwrotnego wyjściowego z funkcją nieliniową *(output feedback with a nonlinear function) – OFBNLF*, w której klucz zmienia się wraz z każdym blokiem. Tryb ten można opisać w następujący sposób:

$$C_i = E_{Ki}(P_i);$$
$$P_i = D_{Ki}(C_i);$$
$$K_i = E_K(K_{i-1.})$$

Błąd ***1***-bitowy w szyfrogramie rozprzestrzenia się tylko w jednym bloku tekstu jawnego. Jeśli jednak ***1*** bit zniknie lub zostanie wstawiony, to pojawi się nieskończony ciąg błędów. Ten tryb pracy działa wolno w przypadku algorytmów ze skomplikowanymi operacjami na kluczu, jak np. DES.

Poza omówionymi trybami istnieją jeszcze inne, jednak nie tak powszechnie używane. Wiązanie bloków tekstu jawnego *(plaintext block chaining) – PBC* jest podobny do CBC z tą różnicą, że to poprzedni blok tekstu jawnego jest dodawany modulo 2 do bloku tekstu jawnego zamiast do bloku szyfrogramu. Sprzężenie zwrotne tekstu jawnego *(plaintext feedback) – PFB* jest z kolei podobny do trybu CFB z tą różnicą, że to tekst jawny jest wykorzystywany do organizacji sprzężenia zwrotnego. Te dwa tryby umożliwiają przeprowadzenie ***ataku z wybranym tekstem jawnym*** z zamiarem uodpornienia na ***atak ze znanym tekstem jawnym***. Istnieje też metoda wiązania bloków zaszyfrowanych różnic tekstu jawnego *(cipher block chaining of plaintext difference) – CBCPD*.

## 7.8. Zastosowania trybów pracy symetrycznych algorytmów kryptograficznych

Cztery podstawowe tryby pracy algorytmów symetrycznych (ECB, CBC, OFB, CFB) dostarczają mechanizmy które powinny zaspokoić większość potrzeb stawianych systemom kryptograficznym. Tryby te nie są nadmiernie skomplikowane i prawdopodobnie nie powodują zmniejszenia stopnia ochrony systemu. Istnieją również inne tryby (np. omawiane tryby PBC, PFB, CBCPD) często bardziej skomplikowane które jednak nie zwiększają poziomu bezpieczeństwa, a jedynie wzrasta złożoność obliczeniowa systemu. Właściwości każdego z tych trybów nie są wcale lepsze ani pod względem propagacji błędów, ani cech związanych z usuwaniem błędów. Dlatego właśnie poniżej zostaną przedstawione cechy czterech podstawowych trybów, które predysponują je do pewnych zastosowań, a wykluczają inne. Na końcu tej części wykładu zamieszczono tabelę zawierającą najbardziej charakterystyczne cechy trybów ECB, CBC, OFB oraz CFB.

Tryb ECB charakteryzuje prostota i szybkość działania, ale niestety jest również najsłabszym trybem pracy algorytmów blokowych. Tryb ten jest podatny na ***ataki za pomocą powtarzania bloków***, oraz najłatwiejszy do kryptoanalizy. Te cechy poważnie ograniczają możliwości stosowania trybu ECB bez ograniczania poziomu bezpieczeństwa całego systemu. Jednak istnieją obszary w których tryb ten może być stosowany. Tryb ten nadaje się do szyfrowania krótkich danych losowych, takich jak inne klucze.

Tryb CBC nadaje się do szyfrowania plików. Zapewnia dobrą ochronę danych, oraz ogranicza możliwość pojawienia się błędów synchronizacji. Tryb ten okazuje się również bardzo przydatny dla programowych implementacji algorytmów kryptograficznych.

Tryb CFB, jest zwykle wybierany do zastosowań które wymagają indywidualnego szyfrowania np. pojedynczych znaków. W takiej sytuacji najlepszym rozwiązaniem będzie zastosowanie 8-bitowego CFB (znak zapisywany na 8 bitach).

Tryb OFB zalecany jest do zastosowań w środowiskach bardzo podatnych na błędy, ponieważ tryb ten nie rozprzestrzenia błędów. Tryb ten jest stosowany w systemach z transmisją synchroniczną o dużej szybkości.

Natomiast do szyfrowania zwykłego tekstu jawnego należy używać jednej z metod: CBC, CFB, OFB. Wybór konkretnej metody zależy od indywidualnych wymagań danego systemu.

Tabela 7.1

Właściwości pracy trybów blokowych algorytmów kryptograficznych

<table>
<tr><th>Nazwa trybu</th><th colspan="2">Właściwości trybu</th></tr>
<tr><td rowspan="8">ECB</td><td colspan="2">Bezpieczeństwo</td></tr>
<tr><td>Zalety</td><td>Wady</td></tr>
<tr><td>- Ten sam klucz można użyć do szyfrowania większej liczby wiadomości.</td><td>- Charakterystyczne fragmenty tekstu jawnego nie są ukrywane.<br>- Ciąg wejściowy szyfru blokowego nie ma charakteru losowego – jest dokładnie taki sam jak tekst jawny.<br>- Istnieje możliwość manipulacji tekstem jawnym. Bloki szyfrogramu mogą być usuwane, powtarzane lub zamieniane miejscami.</td></tr>
<tr><td colspan="2">Wydajność</td></tr>
<tr><td>Zalety</td><td>Wady</td></tr>
<tr><td>- Szybkość jest taka sama jak szyfru blokowego.<br>- Przetwarzanie może być zrównoleglone.</td><td>- Szyfrogram jest co najwyżej dłuższy od tekstu jawnego o wielkość dopełniania.<br>- Brak możliwości dokonywania obliczeń wstępnych.</td></tr>
<tr><td colspan="2">Odporność na błędy</td></tr>
<tr><td>Zalety</td><td>Wady</td></tr>
<tr><td></td><td>- Błąd w szyfrogramie wpływa na jeden cały blok tekstu jawnego.<br>- Błąd synchronizacji nie jest odtwarzalny.</td></tr>
</table>

<table>
<tr><td rowspan="6">CBC</td><td colspan="2">Bezpieczeństwo</td></tr>
<tr><td>Zalety</td><td>Wady</td></tr>
<tr><td>- Charakterystyczne fragmenty tekstu jawnego są ukrywane przez sumowanie modulo 2 z poprzednimi blokami szyfrogramu.<br>- Ciąg wejściowy szyfru blokowego ma charakter losowy przez sumowanie modulo 2 z poprzednimi blokami szyfrogramu.<br>- Więcej niż jedna wiadomość może być zaszyfrowana tym samym kluczem.</td><td>- Możliwość manipulacji tekstem jawnym. Bloki mogą być usuwane z początku i końca wiadomości. Bity pierwszego bloku mogą być zmieniane i powtarzanie umożliwia dokonanie pewnych zamierzonych zmian.</td></tr>
<tr><td colspan="2">Wydajność</td></tr>
<tr><td>Zalety</td><td>Wady</td></tr>
<tr><td>- Szybkość jest taka sama jak szyfru blokowego.<br>- Deszyfrowanie może być zrównoleglone i ma właściwości dostępu swobodnego.</td><td>- Szyfrogram jest co najwyżej dłuższy od teksty jawnego o wielkość dopełnienia, nie licząc wektora początkowego.<br>- Nie ma możliwości dokonywania obliczeń wstępnych.<br>- Proces szyfrowania nie może być zrównoleglony.</td></tr>
<tr><td colspan="2">Odporność na błędy</td></tr>
<tr><td>Zalety</td><td>Wady</td></tr>
<tr><td></td><td>- Błąd w szyfrogramie wpływa na jeden cały blok tekstu jawnego i odpowiadający mu bit w bloku następnym.<br>- Błąd synchronizacji nie jest odtwarzalny.</td></tr>
</table>

<table>
<tr><td rowspan="6">CFB</td><td colspan="2">Bezpieczeństwo</td></tr>
<tr><td>Zalety</td><td>Wady</td></tr>
<tr><td>- Charakterystyczne fragmenty tekstu jawnego są ukrywane.<br>- Ciąg wejściowy szyfru blokowego ma charakter losowy.<br>- Więcej niż jedna wiadomość może być zaszyfrowana tym samym kluczem, przyjmując że będzie używany inny ciąg wektora początkowego.</td><td>- Możliwość manipulacji tekstem jawnym. Bloki mogą być usuwane z początku i końca wiadomości. Bity pierwszego bloku mogą być zmieniane i powtarzanie umożliwia dokonanie pewnych zamierzonych zmian.</td></tr>
<tr><td colspan="2">Wydajność</td></tr>
<tr><td>Zalety</td><td>Wady</td></tr>
<tr><td>- Szybkość jest taka sama jak szyfru blokowego wtedy gdy sprzężenie zwrotne jest tego samego rozmiaru co dla bazowego szyfru blokowego.<br>- Szyfrogram jest tej samej długości co tekst jawny, nie licząc wektora początkowego.<br>- Deszyfrowanie może być zrównoleglone i ma właściwości dostępu swobodnego.</td><td>- Jest możliwe dokonywanie pewnych obliczeń wstępnych, zanim blok pojawi się, a poprzedni blok szyfrogramu może być zaszyfrowany.<br>- Szyfrowanie nie może być zrównoleglone.</td></tr>
<tr><td colspan="2">Odporność na błędy</td></tr>
<tr><td>Zalety</td><td>Wady</td></tr>
<tr><td>- Błędy synchronizacji dla całych bloków są odtwarzalne. Tryb 1-bitowego CFB może odtworzyć swoje działanie po wstawieniu lub utracie 1 bitu.</td><td>- Błąd w szyfrogramie wpływa na jeden bit odpowiadającego mu tekstu jawnego i cały następny blok tekstu jawnego.</td></tr>
</table>

<table>
<tr><td rowspan="6">OFB<br>(licznikowy)</td><td colspan="2">Bezpieczeństwo</td></tr>
<tr><td>Zalety</td><td>Wady</td></tr>
<tr><td>- Charakterystyczne cechy tekstu jawnego są ukrywane.<br>- Ciąg wejściowy szyfru blokowego ma charakter losowy.<br>- Więcej niż jedna wiadomość może być zaszyfrowana tym samym kluczem.</td><td>- Bardzo łatwa jest manipulacja tekstem jawnym. Dowolna zmiana w szyfrogramie bezpośrednio wpływa na tekst jawny.</td></tr>
<tr><td colspan="2">Wydajność</td></tr>
<tr><td>Zalety</td><td>Wady</td></tr>
<tr><td>- Szybkość jest taka sama jak szybkość szyfru blokowego.<br>- Szyfrogram jest tej samej długości co tekst jawny, nie licząc wektora początkowego.<br>- W trybie licznikowym istnieje możliwość zrównoleglenia pracy.</td><td>- Jest możliwe dokonywanie pewnych obliczeń wstępnych, zanim wiadomość się pojawi.<br>- Przetwarzanie w trybie OFB nie może być zrównoleglone.</td></tr>
<tr><td colspan="2">Odporność na błędy</td></tr>
<tr><td>Zalety</td><td>Wady</td></tr>
<tr><td>- Błąd w szyfrogramie wpływa na odpowiadający mu bit tekstu jawnego.</td><td>- Błędy synchronizacji nie są odtwarzalne.</td></tr>
</table>