Szyfry strumieniowe na przykładzie algorytmów A5/1 i Rabbit

mgr inż. ROBERT POZNAŃSKI, Instytut Maszyn Matematycznych, Warszawa

Przez długi czas szyfry strumieniowe pozostawały w cieniu bardziej popularnych szyfrów blokowych. Jednym z pierwszych algorytmów blokowych wyznaczających standardy był powstały w 1975 r. DES – *Data Encryption Standard*. Jego działanie polegało na szyfrowaniu bloków po 64 bity na raz. W dzisiejszych algorytmach standardem są bloki długości 128 bitów i więcej. Szyfry strumieniowe działają w trochę inny sposób. Wynikiem ich działania jest strumień bitów, który służy do zaszyfrowania wiadomości. Można więc o nich powiedzieć, że są bardziej generatorami strumienia klucza niż algorytmami szyfrującymi. Jednym z takich algorytmów był zaprezentowany w 1987 r. A5/1. Jego przeznaczeniem było szyfrowanie transmisji w sieciach GSM i pod to konkretne rozwiązanie został projektowany. Innym ciekawym przykładem podejścia do szyfrów strumieniowych jest pokazany w 2003 r. algorytm Rabbit. Jest to algorytm ogólnego przeznaczenia, który może zostać użyty w praktycznie każdym zastosowaniu. Na początku warto jednak zapoznać się z ogólną budową i ideą szyfrowania strumieniowego.

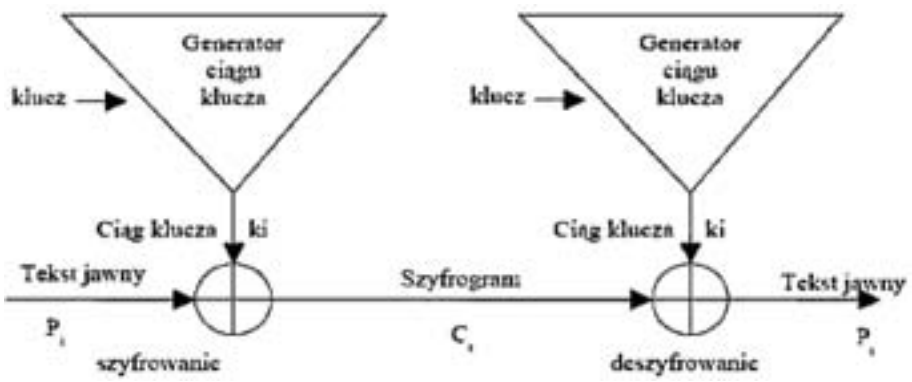
Szyfry strumieniowe

Szyfry strumieniowe są algorytmami, które przekształcają tekst jawny w szyfrogram kolejno bit po bicie – *rys. 1*.

Generator strumienia klucza wytwarza strumień bitów K, który jest dodawany modulo 2 z ciągiem bitów tekstu jawnego P celem wygenerowania strumienia bitów szyfrogramu $C = P \oplus K$.

Bezpieczeństwo systemu całkowicie zależy od wewnętrznych właściwości generatora strumienia klucza. Jeżeli generator strumienia klucza wytwarza nieskończony ciąg zer, to szyfrogram będzie równy tekstowi jawnemu i cała operacja nie będzie miała sensu. Jeżeli generator strumienia klucza wytwarza powtarzający się wzorzec 16-bitowy, to algorytm będzie zwykłym sumatorem modulo 2 z bardzo małym (pomijalnym) stopniem zabezpieczenia. Jeżeli generator strumienia klucza wytwarza nieskończony strumień bitów losowych (nie pseudolosowych), to otrzymujemy klucz jednorazowy i doskonałe zabezpieczenie. W rzeczywistości szyfry strumieniowe to coś pośredniego między prostą sumą modulo 2 i szyfrowaniem z kluczem jednorazowym. Generator strumienia klucza wytwarza ciąg bitów, który wygląda losowo, ale w rzeczywistości jest ciągiem zdeterminowanym, który może być bezbłędnie odtworzony podczas odszyfrowywania. Im bliższy postaci losowej jest ciąg wyjściowy generatora strumienia klucza, tym trudniejsze dla kryptoanalityka będzie jego złamanie. Jak można przypuszczać, zadanie zbudowania generatora strumienia klucza wytwarzającego losowo wyglądający ciąg nie należy do łatwych. Szyfrowanie strumieniowe polega na szyfrowaniu informacji kluczem złożonym ze strumienia danych (bitów lub znaków), nie krótszym od szyfrowanej informacji. Szyfry strumieniowe dzielą tekst M na części lub bity 1, 2,.., m, a następnie każdy element jest szyfrowany kluczem k_i należącym do strumienia kluczy.

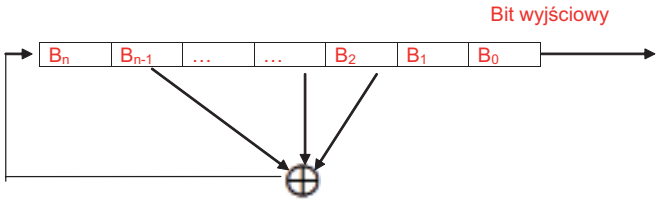
Szyfr strumieniowy jest okresowy, jeśli strumień klucza powtarza się po *T* znakach, dla pewnego ustalonego *T*. W przeciwnym razie szyfr jest nieokresowy. Do okresowych szyfrów strumieniowych należą np. szyfry generowane przez maszyny rotorowe (Enigma – okres większy niż 26^k, gdzie k oznacza liczbę rotorów, oryginalnie k = 3 oraz k = 5, 26^5 = 11 881 376). Natomiast szyfr jednokrotny i szyfry z kluczem bieżącym są nieokresowymi szyframi strumieniowymi. Okres jest bardzo istotnym parametrem generatora. Decyduje on jak długo można ten generator stosować bez zmiany parametrów początkowych. Z teorii na temat szyfru z kluczem jednorazowym wiadomo, że niedopuszczalne jest użycie dwa razy tego samego klucza. Oznacza to, że nie można używać generatora dłużej niż wynosi jego okres, gdyż groziłoby to właśnie powtórzeniem tego samego ciągu klucza. Dlatego ważne jest, aby kres generatora był jak najdłuższy, co pozwoliłoby długo używać tego samego generatora bez zmiany jego parametrów. Ponieważ konieczna jest zmiana klucza wraz z każdą wiadomością, algorytmy strumieniowe nie są zazwyczaj używane do szyfrowania wydzielonych wiadomości. Bardziej użyteczne są one w szyfrowaniu bardzo długich strumieni informacji. Mogą to być na przykład transmisje sygnałów wideo, audio. Ponieważ generator strumienia klucza musi wytwarzać te same wartości zarówno do szyfrowania, jak i odszyfrowywania, musi być on zdeterminowany. Sek-



Rys. 1. Szyfr strumieniowy z kluczami. Fig. 1. Stream cipher with keys

wencje wyjściowe mogą powtarzać się, ponieważ jest on zbudowany z wykorzystaniem maszyny ze skończoną liczbą stanów (tj. komputera). Takie generatory strumieni klucza noszą nazwę okresowych. Z wyjątkiem przypadku wytwarzania kluczy jednorazowych wszystkie generatory strumieni klucza są okresowe. Bardzo ważne jest uzyskanie długiego okresu dla generatora strumienia klucza i to znacznie dłuższego od liczby bitów, które generator wytworzy w czasie pomiędzy zmianami kluczy. Okres generatora strumienia klucza musi być o wiele rzędów wielkości większy niż podana wartość.

Głównym elementem, w oparciu o który można zbudować algorytm strumieniowy jest tzw. LFSR (*Linear Feedback Shift Register* – Rejestr Przesuwny ze Sprzężeniem Zwrotnym).



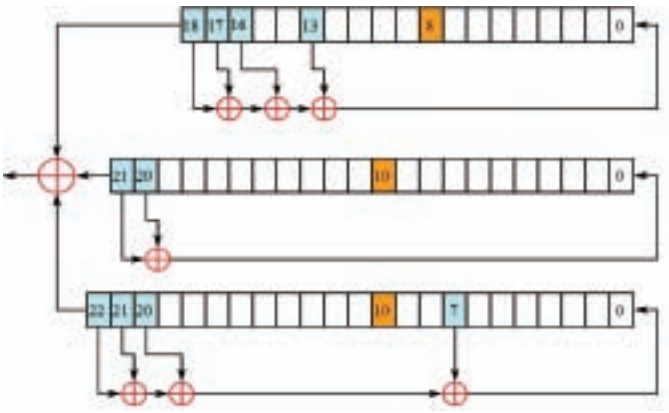
Rys. 2. Liniowy rejestr przesuwny ze sprzężeniem zwrotnym
Fig. 2. Linear Feedback Shift Register

Składa się on z ciągu bitów oraz odczepów, które są wejściem do funkcji XOR generującej bit wchodzący do rejestru na pozycji najbardziej znaczącego bitu. Wyjściem LFSRa jest bit najmniej znaczący (*rys. 2*). Oczywiście nic nie stoi na przeszkodzie, aby do budowy szyfru strumieniowego użyć więcej niż jednego LFSRa. Podejście takie jest często stosowane i zwykle powoduje wzrost bezpieczeństwa algorytmu. Aby osiągnąć maksymalny okres dla układu kilku rejestrów ich długości muszą być względnie pierwsze, a wielomiany charakterystyczne, utworzone z elementów ciągu odczepów muszą być wielomianami pierwotnymi.

Algorytm A5/1

Jednak bezpieczeństwo kryptograficzne algorytmu opartego tylko o LFSR jest bardzo słabe. Aby je wzmocnić rozbudowuje się algorytm o dodatkowe LFSRy oraz dodaje funkcjonalność nieliniowego ich taktowania. Jednym z przykładów takiego algorytmu jest A5/1 używany do szyfrowania transmisji GSM. Czas trwania ramki GSM wynosi 4,615 milisekundy i wynosi 114 bitów. Stanem początkowym algorytmu są same zera, natomiast 64-bitowy klucz tajny razem z jawnym 22-bitowym numerem ramki służą do inicjalizacji algorytmu. A5/1 oparty jest na trzech rejestrach przesuwnych ze sprzężeniem zwrotnym R1, R2, R3, o długości 19, 22, 23 bitów. Wielomiany charakterystyczne wynoszą odpowiednio $x^{19}+x^{18}+x^{17}+x^{14}+1$, $x^{22}+x^{21}+1$ oraz $x^{23}+x^{22}+x^{21}+x^{8}+1$, a numery bitów służące do sterowania pracą rejestrów 8, 10, 10 (*rys. 3*). Wyjściem algorytmu jest suma algebraiczna modulo 2 wyjść wszystkich LFSR.

Do taktowania rejestrów wykorzystuje po jednym bicie z każdego rejestru. W każdym cyklu analizowana jest zawartość tych bitów. Jeśli co najmniej dwa mają wartość 1 to taktowane są tylko rejestry, w których bity te miały wartość 1. W przeciwnym przypadku taktowane są te rejestry, które miały wartość 0. Jak widać w każdym takcie pracy algorytmu taktowane są przynajmniej 2 rejestry, każdy z prawdopodobieństwem 0,75. Schemat ten można zapisać jak w *tabeli*.



Rys. 3. Schemat budowy A5/1 (źródło wikipedia)
Fig. 3. Scheme of A5/1 (Skurce wkikipedia)

Taktowanie rejestrów. Register Control

Wartość R1	Wartość R2	Wartość R3	Takt R1	Takt R2	Takt R3
0	0	0	Tak	Tak	Tak
0	0	1	Tak	Tak	Nie
0	1	0	Tak	Nie	Tak
0	1	1	Nie	Tak	Tak
1	0	0	Nie	Tak	Tak
1	0	1	Tak	Nie	Tak
1	1	0	Tak	Tak	Nie
1	1	1	Tak	Tak	Tak

Praca algorytmu zaczyna się od procesu jego inicjalizacji. Najpierw, z pominięciem sterowania taktowaniem rejestrów, wprowadzany jest 64-bitowy klucz, następnie 22-bitowy numer ramki. Obie te operacje trwają odpowiednio 64 i 22 takty. Kolejną operacją jest mieszanie, które trwa 100 cykli zegarowych. W czasie tej operacji sterowanie rejestrami odbywa się już w normalny sposób, natomiast ignorowane są wyjścia z algorytmu. Właściwe szyfrowanie trwa 228 rund w czasie których generowanych jest 228 bitów strumienia szyfrującego po 114 bitów na wysłanie i odebranie ramki.

Zaletą algorytmów opartych na LFSR jest wysoka szybkość działania i stosunkowo łatwa implementacja w rozwiązaniach sprzętowych. Jeden takt pracy zegara może odpowiadać jednemu taktowi pracy rejestru dzięki czemu, łatwo jest dostosować pracę takiego algorytmu do pracy innych urządzeń. Jednak algorytmy strumieniowe nie muszą być oparte o rejestr przesuwny ze sprzężeniem zwrotnym.

Algorytm Rabbit

W latach 2004–2008 organizacja ECRYPT przeprowadziła konkurs eSTREAM mający na celu zaprezentowanie nowych technologii w algorytmach strumieniowych. Jednym z algorytmów biorących udział w konkursie był zaprezentowany już

w 2003 roku na konferencji Fast Software Encryption, algorytm Rabbit. Został on jednym z czterech laureatów konkursu w kategorii algorytmów przeznaczonych do implementacji programowej. Zgłaszany był także w kategorii algorytmów przeznaczonych do implementacji sprzętowej, jednak nie odniósł tam większych sukcesów. Jego budowa jest zdecydowanie bardziej skomplikowana niż A5/1. Parametrami wejściowymi Rabbita są klucz o długości 128 bitów oraz wektor inicjalizujący IV o długości 64 bitów. Wektor IV jest dodatkową wartością służącą do inicjalizacji algorytmów kryptograficznych, użycie jego nie jest obowiązkowe, a przesyłany jest jawnie, dołączony do szyfrogramu. Rabbit generuje strumień klucza w 128-bitowych blokach.

Budowa algorytmu Rabbit nie jest oparta na rejestrze przesuwnym ze sprzężeniem zwrotnym. Jego stan wewnętrzny składa się z 513 bitów podzielonych na dwie grupy po osiem 32 bitowych rejestrów ($C_0...C_7$, $X_0...X_7$) i jeden bit 'b' służący za wskaźnik przeniesienia. Cykl pracy algorytmu podzielić można na poszczególne części takie jak funkcja licznika, funkcja następnego stanu oraz ekstrakcja strumienia. Na główną rundę składa się kolejne przejście przez wszystkie trzech części algorytmu.

Praca algorytmu rozpoczyna się od inicjalizacji, czyli zapełnienia stanu wewnętrznego bitami klucza oraz wektora IV. Klucz dzielony jest na osiem podkluczy K_j dla j = 0..7. Następnie wykonywana jest operacja wg schematu, dla parzystych j: $X_j = k_{(j+1 \bmod 8)} \,||\, k_j$, $C_j = k_{(j+4 \bmod 8)} \,||\, k_{(j+5 \bmod 8)}$ dla nieparzystych j: $X_j = k_{(j+5 \bmod 8)} \,||\, k_{(j+4 \bmod 8)}$, $C_j = k_j \,||\, K_{(j+1 \bmod 8)}$. Operacja || oznacza konkatenację, czyli złączenie dwóch wartości. Po zapełnieniu wszystkich X oraz C następuje wykonanie czterech iteracji par funkcji licznika i następnego stanu. Ostatnim elementem inicjalizacji klucza jest uaktualnienie stanów $C_j = C_j \oplus X_{(j+4 \bmod 8)}$

Następnie wykonywana jest inicjalizacja wektora IV. Polega ona na dodaniu wartości bitów o numerach IV [63..0] do stanów C:

$C_0 = C_0 \oplus IV\ [31..0]$
$C_1 = C_1 \oplus (IV\ [63..48] \,||\, IV\ [31..16])$
$C_2 = C_2 \oplus IV\ [63..32]$
$C_3 = C_3 \oplus (IV\ [47..32] \,||\, IV\ [15..0])$
$C_4 = C_4 \oplus IV\ [31..0]$
$C_5 = C_5 \oplus (IV\ [63..48] \,||\, IV\ [31..16])$
$C_6 = C_6 \oplus IV\ [63..32]$
$C_7 = C_7 \oplus (IV\ [47..32] \,||\, IV\ [15..0])$

Następnie wykonywana jest czterokrotna iteracja pary funkcji licznika i następnego stanu. Tak przygotowany algorytm jest już gotowy do działania. Na jego rundę składają się trzy, kolejno wykonywane po sobie funkcje. Funkcja licznika, funkcja następnego stanu oraz ekstrakcja strumienia. Funkcja licznika służy do uaktualniania stanu C.

$C_0 = C_0 + A_0 + b_{(j-1)}$ modulo 2^{32}
$C_1 = C_1 + A_1 + b_0$ modulo 2^{32}
$C_2 = C_2 + A_2 + b_1$ modulo 2^{32}
$C_3 = C_3 + A_3 + b_2$ modulo 2^{32}
$C_4 = C_4 + A_4 + b_3$ modulo 2^{32}
$C_5 = C_5 + A_5 + b_4$ modulo 2^{32}
$C_6 = C_6 + A_6 + b_5$ modulo 2^{32}
$C_7 = C_7 + A_7 + b_6$ modulo 2^{32}

Natomiast bit przeniesienia przyjmuje wartości: bj = 1 jeżeli $C_0 + A_0 + b'_7$ modulo 2^{32} dla j = 0, 1 jeżeli $C_j + A_j + b_{(j-1)}$ modulo 2^{32} dla j > 0, 0 w przeciwnym przypadku. Operacja + oznacza dodawanie algebraiczne. Stałe mają wartości A0 = 0x4D34D34D, A1 = 0xD34D34D3, A2 = 0x34D34D34, A3 = 0x4D34D34D, A4 = 0xD34D34D3, A5 = 0x34D34D34, A6 = 0x4D34D34D, A7 = 0xD34D34D3.

Głównym elementem algorytmu Rabbit jest funkcja następnego stanu i to przede wszystkim na niej opiera się bezpieczeństwo całego algorytmu. Jej podstawowym działaniem jest funkcja G opisana wzorem: $G_j = (X_j + C_j) \oplus ((X_j + C_j) <<32) \bmod 2^{32}$, gdzie $<<i$ oznacza rotację w lewo o *i* bitów. Drugim elementem funkcji następnego stanu jest uaktualnienie rejestru X wg opisu:

$X_0 = G_0 + (G_7 <<< 16) + (G_6 <<< 16)$
$X_1 = G_1 + (G_0 <<< 8) + G_7$
$X_2 = G_2 + (G_1 <<< 16) + (G_0 <<< 16)$
$X_3 = G_3 + (G_2 <<< 8) + G_1$
$X_4 = G_4 + (G_3 <<< 16) + (G_2 <<< 16)$
$X_5 = G_5 + (G_4 <<< 8) + G_3$
$X_6 = G_6 + (G_5 <<< 16) + (G_4 <<< 16)$
$X_7 = G_7 + (G_6 <<< 8) + G_5$

Ostatnią czynnością wykonywaną w każdej rundzie jest ekstrakcja strumienia szyfrującego. Przebiega ona wg schematu: $S[15..0] = X_0[15..0] \oplus X_5[31..16]$, $S[31..16] = X_0[31..16] \oplus X_3[15..0]$, $S[47..32] = X_2[15..0] \oplus X_7[31..16]$, $S[63..48] = X_2[31..16] \oplus X_5[15..0]$, $S[79..64] = X_4[15..0] \oplus X_1[31..16]$, $S[95..80] = X_4[31..16] \oplus X_7[15..0]$, $S[111..96] = X_6[15..0] \oplus X_3[31..16]$

Z wyjścia S pobierane są 128 bitowe bloki strumieni szyfrującego. Jest to całkiem odmienne podejście niż przy poprzednim algorytmie. W A5/1 w jednym takcie zegara generowanybył jeden bit strumienia szyfrującego. Jednak zaznaczyć trzeba, że Rabbit jest algorytmem ogólnego przeznaczenia projektowanym przede wszystkim do zastosowania w implementacjach programowych. A5/1 jest algorytmem projektowanym do konkretnego zastosowania jakim jest szyfrowanie transmisji w sieciach GSM i jest implementowany w sprzęcie.

Podsumowanie

Na przykładzie algorytmów A5/1 oraz Rabbit widać wyraźnie jak odmienne może być podejście do projektowania szyfrów strumieniowych. W pierwszym przypadku zastosowano trzy rejestry liniowe ze sprzężeniem zwrotnym, sterowane w nieregularny sposób, a wyjście z każdego z nich bierze udział w generowaniu strumienia szyfrującego. Takie rozwiązanie stawia przede wszystkim na wydajność i łatwość implementacji. Algorytm Rabbit został zorientowany przede wszystkim na bezpieczeństwo. W celu jego zapewnienia została stworzona funkcja G. Niestety, ukierunkowanie na wysokie bezpieczeństwo pociągnęło za sobą zauważalny, choć nie dramatyczny spadek wydajności, co spowodowane jest użyciem operacji potęgowania. Są dwa odmienne podejścia do projektowania i budowy szyfrów strumieniowych. W ostatnim czasie pojawia się coraz więcej algorytmów podobnych do Rabbita, tworzonych bez użycia LFSRów.