Kryptografia i kryptoanaliza

Laboratorium 4

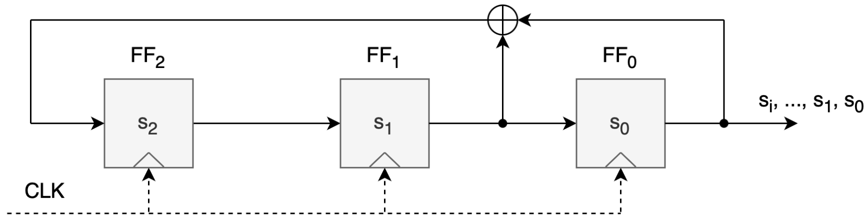
Michał Łaskawski

Zadanie 1

Dokonaj implementacji rejestru przesuwnego z liniowym sprzężeniem zwrotnym `LFSR`. Rejestr ten powinien:

- Przyjmować parametry:
  - sekwencję zer i jedynek, inicjującą stan początkowy rejestru, wartość `m`, która określa stopień rejestru a tym samym ilość jego stanów,
  - sekwencję liczb całkowitych określających połączenia sprzężenia zwrotnego rejestru,
  - dodatnią liczbę całkowitą określającą długość wyjściowego strumienia bitów (zer i jedynek).
- Zwracać wyjściowy strumień bitów.

Przykład. Rejestr przesuwny z liniowym sprzężeniem zwrotnym stopnia m = 3.



Rysunek 1: Liniowy rejestr przesuwający ze sprzężeniem zwrotnym stopnia 3

Powyższy rejestr zdefiniowany może być przy pomocy wielomianu (w porządku rosnącym): $C(x) = p_0x^0 + p_1x + p_2x^2 + x^3$ gdzie p_0, p_1, p_2 przyjmują odpowiednio wartości 1, 1, 0, co daje: $C(x) = 1 + x + x^3$.

Powyższy rejestr, którego stan początkowy określa sekwencja $s_0 = 0$, $s_1 = 0$ oraz $s_2 = 1$, wygeneruje sekwencję wyjściową w postaci: `[0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1]`, przyjmując że żądana długość strumienia wyjściowego wynosi 14.

Zadanie 2

Wykorzystaj program z zadania pierwszego do utworzenia rejestrów `LFSR` opisanych wielomianami:

1. $C(x) = 1 + x^2 + x^5$
2. $C(x) = 1 + x + x^3 + x^5$

Przyjmując, że:

- sekwencja inicjująca ma postać: `[1,0,0,1,0]`, (uwaga, kolejność bitów w odniesieniu do stanów: s_0, s_1, s_2, s_3, s_4),
- oraz że żądana długość sekwencji wyjściowej wynosi 25,

to dla zdefiniowanych kolejno rejestrów, sekwencja wyjściowa powinna być następująca:

1. `[1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1]`
2. `[1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1]`

Zadanie 3

Na podstawie otrzymanych w poprzednich zadaniach, sekwencji bitów generatora. Zidentyfikuj parametry rejestru `LFSR`, który został użyty do wygenerowania tych sekwencji. Parametrami podlegającymi identyfikacji są:

- sekwencja początkowa (`seed`),
- wektor określający sprzężenia zwrotne, które wchodzą do obliczeń `XOR` (`taps`).

W tym celu wykorzystaj algorytm `Berlekamp - Massey`, który pracuje nad polem $GF(2)$. Algorytm ten jest efektywnym narzędziem służącym do znalezienia minimalnego wielomianu połączeń dla danej sekwencji nad ciałem skończonym $GF(q)$ (w ogólnym przypadku).

Algorytm ten korzystając z danej sekwencji bitów, powinien wygenerować informacje o złożoności liniowej L oraz wielomian połączeń C, czyli:

- *Złożoność liniowa* L: to najmniejsza liczba poprzednich elementów sekwencji potrzebna do wyznaczenia kolejnego elementu w sposób liniowy.
- *Wielomian połączeń* $C(x)$: Wielomian opisujący zależności między elementami sekwencji.

Algorytm iteracyjnie analizuje sekwencję, obliczając rozbieżność d między przewidywaną a rzeczywistą wartością sekwencji, następnie aktualizuje wielomian połączeń $C(x)$ oraz złożoność liniową L, gdy jest to konieczne.

Algorithm 1 Algorytm Berlekamp-Massey

Require: Sekwencja $s = [s_0, s_1, \dots, s_{n-1}]$ nad $GF(2)$
Ensure: Wielomian połączeń $C(x)$ oraz złożoność liniowa L

$n \leftarrow$ długość sekwencji s
$C(x) \leftarrow [1]$ ▷ Inicjalizacja wielomianu połączeń
$B(x) \leftarrow [1]$ ▷ Kopia poprzedniego wielomianu
$L \leftarrow 0$ ▷ Złożoność liniowa
$m \leftarrow -1$ ▷ Indeks ostatniej aktualizacji
for $N \leftarrow 0$ **do** $n - 1$ **do**
  $d \leftarrow s_N$
  for $i \leftarrow 1$ **do** L **do**
    $d \leftarrow d \oplus (C_i \cdot s_{N-i})$ ▷ Oblicz rozbieżność d
  end for
  if $d = 0$ **then**
    continue ▷ Brak potrzeby aktualizacji
  end if
  $T(x) \leftarrow C(x)$ ▷ Kopia aktualnego wielomianu $C(x)$
  $\delta \leftarrow N - m$
  for $i \leftarrow 0$ **do** $|B(x)| - 1$ **do**
    $C_{\delta+i} \leftarrow C_{\delta+i} \oplus B_i$ ▷ Aktualizacja $C(x)$
  end for
  if $2L \leq N$ **then**
    $L \leftarrow N + 1 - L$
    $B(x) \leftarrow T(x)$
    $m \leftarrow N$
  end if
end for
return $C(x), L$

Objaśnienie algorytmu:

1. **Inicjalizacja:**
   - $C(x)$: ustawiamy wielomian połączeń na [1] (współczynnik przy x^0).

- $B(x)$: kopia pomocnicza wielomianu połączeń, również [1].
- L: złożoność liniowa sekwencji, początkowo 0.
- m: indeks ostatniej aktualizacji wielomianu połączeń, początkowo -1.

2. **Pętla główna (dla każdego elementu sekwencji):**
   - **Obliczenie rozbieżności** d:

$$d = s_N \oplus \left(\bigoplus_{i=1}^{L} C_i \cdot s_{N-i} \right)$$

     - Rozbieżność d określa, czy aktualny wielomian połączeń poprawnie przewiduje wartość s_N.
   - **Jeśli** $d = 0$:
     - Aktualny wielomian $C(x)$ poprawnie przewiduje s_N, to przechodzimy do następnej iteracji.
   - **Jeśli** $d \neq 0$:
     - **Kopia** $T(x)$: zachowujemy aktualny wielomian w $T(x)$.
     - **Aktualizacja** $C(x)$:

$$C(x) = C(x) \oplus x^{\delta} \cdot B(x)$$

       gdzie $\delta = N - m$.
     - **Aktualizacja parametrów, jeśli** $2L \leq N$:
       * $L = N + 1 - L$: aktualizujemy złożoność liniową.
       * $B(x) = T(x)$: ustawiamy $B(x)$ na poprzedni $C(x)$.
       * $m = N$: aktualizujemy indeks ostatniej zmiany.
3. **Zakończenie:**
   - Po przetworzeniu całej sekwencji zwracamy wielomian połączeń $C(x)$ i złożoność liniową L.

Uwagi

- Wszystkie operacje wykonywane są w ciele $GF(2)$, czyli `modulo 2`.
- Wielomiany reprezentowane są jako listy współczynników, gdzie C_i to współczynnik przy x^i.
- Operator $\oplus$ reprezentuje dodawanie `modulo 2` (operację `XOR`).
- Symbol $\bigoplus$ oznacza sumę `modulo 2` wielu składników.
- Algorytm wygeneruje wektor współczynników wielomianu C w porządku malejącym.

Przykład obliczeń

Załóżmy, że:

- Złożoność liniowa $L = 3$.
- Współczynniki wielomianu połączeń:
  - $C_1 = 1$
  - $C_2 = 0$
  - $C_3 = 1$
- Elementy sekwencji:
  - $s_N = 0$

- $s_{N-1} = 1$
- $s_{N-2} = 0$
- $s_{N-3} = 1$

Obliczenie rozbieżności d:

$$\begin{aligned} d &= s_N \oplus \left(\bigoplus_{i=1}^{3} C_i \cdot s_{N-i} \right) \\ &= 0 \oplus ((1 \cdot 1) \oplus (0 \cdot 0) \oplus (1 \cdot 1)) \\ &= 0 \oplus (1 \oplus 0 \oplus 1) \\ &= 0 \oplus 0 \\ &= 0 \end{aligned}$$

Zadanie 4

Dokonaj implementacji kryptosystemu strumieniowego, który wykorzystuje pojedynczy `LFSR` jako strumień klucza. Przetestuj zaimplementowany kryptosystem dokonując operacji szyfrowania i deszyfrowania zadanej wiadomości w postaci tekstu w języku angielskim.