Politechnika Białostocka

Sprawozdanie z pracowni specjalistycznej

Bezpieczeństwo Sieci Komputerowych

Temat: **Generatory Liczb Pseudolosowych, Szyfry strumieniowe.**

Wykonujący ćwiczenie: Paweł Szapiel

Studia dzienne

Kierunek: **INFORMATYKA**

Semestr: **VI**

Grupa zajęciowa: **6**

Prowadzący ćwiczenie: mgr inż. Dariusz Jankowski

Data wykonania ćwiczenia: **4.04.2022**

ZADANIE 1.

1. Cel ćwiczenia

Celem zajęć jest kontynuacja zapoznawania się z podstawowymi metodami utajniania informacji. Zaimplementuj i zaprezentuj działanie poniższych algorytmów. Program powinien obsługiwać zarówno opcję szyfrowania, jak i deszyfrowania.

2. Treść zadań

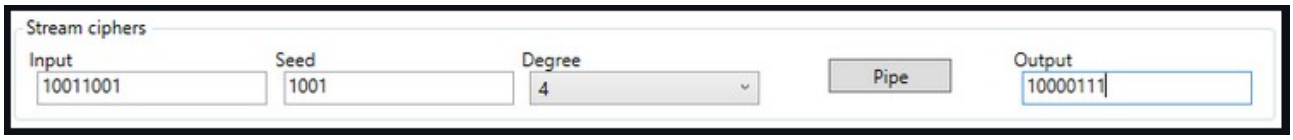
Zadania połączyłem, ponieważ szyfr strumieniowy korzysta z implementacji LFSR.

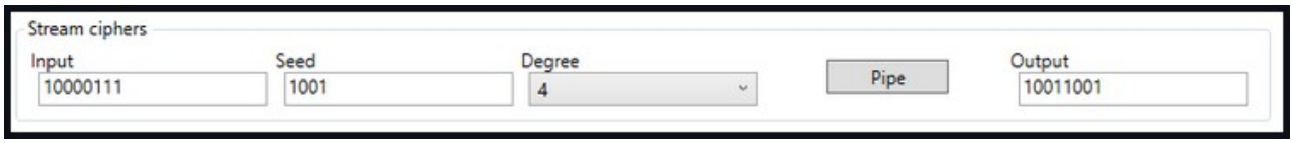
GENERATOR LICZB PSEUDOLOSOWYCH

Zaimplementuj generator liczb pseudolosowych bazujący na metodzie LFSR. Wielomian powinien być podawany przez użytkownika jako parametr. Program powinien generować bity „w nieskończoność” aż do momentu zatrzymania algorytmu przez użytkownika.

SZYFR STRUMIENIOWY

Zaimplementuj kryptosystem bazujący na schemacie Synchronous Stream Cipher dla podanego wielomianu. Do generowania klucza wykorzystaj metodę LFSR. Na potrzeby działania algorytmu informację jawną przekształć do postaci binarnej. Program powinien mieć możliwość obsługi plików różnego typu (szyfrowanie zarówno plików tekstowych, jak i co najmniej jednego innego formatu plików).





4. Instrukcja korzystania z GUI:

1. Wprowadź ciąg bitów o dowolnej długości w pole 'Input'
2. Wpisz ciąg bitów określający początkowe nastawienie generatora w pole 'seed' (długość od 1-9)
3. Ustal długość wielomianu szyfrującego (stopień odpowiada długości ciągu seed od 1 do 9) np. 10010 to 'degree' = 5
4. Kliknij 'Pipe' w celu szyfracji.
5. Skopiuj output i wklej w input. Ponownie kliknij Pipe w celu Deszyfracji

5. Zasada działania

LFSR wygeneruje ten sam ciąg bitów jeżeli seed będzie jednakowy. Dla każdego bitu z input i bitu wygenerowanego stosuję operację xor. Wynik dodaje do wyjścia. Dzięki odwrotności działania 'xor' ciąg zaszyfrowany można zdeszyfrować posiadając te same ustawienia szyfru.

6. Szyfracja – kod.

Kod źródłowy

```
        /// <summary>
        /// Tworzy generator LFSR o danym stopniu wielomianu 'degree', następnie dodaje
bramki xor
        /// przy bitach podanych w parametrze 'polymonial' (1001 -> wstaw xor przy bicie
pierwszy i czwartym)
        /// Początkowe ustawienie bufora określa tablica 'seed'
        /// </summary>
        /// <param name="degree">stopień szyfrującego wielomianu LFSR</param>
        /// <param name="seed">początkowe ustawienie bufora</param>
        /// <param name="polymonial">określa wielomian szyfrujący</param>
        /// <returns>Zwraca wygenerowany ciąg bitów pseudolosowych</returns>
        private int[] GenerateLFSR(int degree, int[] seed, int[] polymonial)
        {

            // Przypisanie seeda do bufora
            int[] buffer = new int[degree];
            for (int i = 0; i < degree; i++)
            {
                buffer[i] = seed[i];
            }

            // Policz jedynki na xor-ach w celu wygenerowania nowego bitu
            int sum = 0;
            for (int i = 0; i < degree; i++)
            {
                if (polymonial[i] == 1 && buffer[i] == 1)
                    sum += 1;
            }

            // shift
            for (int i = degree - 1; i > 0; i--)
            {
                buffer[i] = buffer[i - 1];
            }
            // wpisanie nowego bitu na wejście bufora
            buffer[0] = sum % 2 == 1 ? 1 : 0;

            return buffer;
        }

        /// <summary>
        /// Szyfruje inputStream za pomocą pseudolosowego generatora LFSR
        /// o długości = 'degree' zaczynając od ustawienia 'seed'
        /// </summary>
        /// <param name="inputStream">ciąg do zaszyfrowania</param>
        /// <param name="seed">pozycja początkowa bufora</param>
        /// <param name="degree">stopień wielomianu szyfrującego</param>
        /// <returns>Zwraca zaszyfrowany ciąg bitów tj. tablicę intów 0 lub 1 </returns>
```

```
        public int[]? StreamCipher(int[] inputStream, int[] seed, int degree)
        {
            // Wczytanie tablicy których stopni wielomianu użyć
            // do generowania liczb jeżeli stopień nieobecny w słowniku
            // lub błąd otwarcia json'a kończy funkcję
            Dictionary<string, int[]>? degreesList = ReadJsonFile<Dictionary<string,
int[]>?>();
            if (degreesList == null) return null;

            int[]? degreesToTake = null;
            degreesList?.TryGetValue(degree.ToString(), out degreesToTake);
            if (degreesToTake == null)
            {
                MessageBox.Show("Degree not present in dictionary");
                return null;
            }


            int[] polymonial = new int[degree];
            // Ustawienie jedynek na odpowiednie stopnie wielomianu
            // Odwzorowuje dodanie xor-ów
            for (int i = 0; i < degreesToTake.Length; i++)
            {
                polymonial[degreesToTake[i] - 1] = 1;
            }
            int[] outputStream = new int[inputStream.Length];

            // pobranie pierwszej wartości bufora
            int[] buffer = GenerateLFSR(degree, seed, polymonial);
            for (int i = 0; i < inputStream.Length; i++)
            {
                // wpisanie na wyjście wyniku operacji xor na wartości z input'a i
pierwszej liczby w buforze LSFR
                outputStream[i] = inputStream[i] ^ buffer[0];
                buffer = GenerateLFSR(degree, buffer, polymonial);
            }

            return outputStream;
        }
```

Plik konfiguracyjny paru pierwszych stopni wielomianu:

```
{
  "1": [ 1 ],
  "2": [ 1, 2 ],
  "3": [ 1, 3 ],
  "4": [ 1, 4 ],
  "5": [ 2, 5 ],
  "6": [ 1, 6 ],
  "7": [ 1, 7 ],
  "8": [ 1, 5, 6, 8 ],
  "9": [ 4, 9]
}
```

np. „4”: [1, 4] => 1001 (x^3 + 1)

7. Wnioski

Stopień wielomianu generującego bity w funkcji LFSR, zależy od długości seed’a, dlatego podczas pracy nad algorytmem okazało się, że parametryzacja ‘degree’ nie jest do końca potrzebna. Natomiast nie usunąłem tego parametru w celach edukacyjnych. (Gdy ich długość będzie różna aplikacja powiadomi o błędzie użytkownika)

Rozwijając algorytm, można zaimplementować ręczne wprowadzanie wielomianu określającego pozycję bramek xor w rejestrze, a nie jak to ma miejsce obecnie – jeden wielomian do jednej wielkości rejestru.

Link:
Github Repository