Niezawodność i diagnostyka układów cyfrowych 2
projekt

„Unikanie utraty synchronizacji przy pomocy randomizacji (Scrambling)”

Janusz Długosz — 235746
Jakub Dorda — 235013
Marcin Kotas — 235098

Prowadzący:
Doc. dr inż. Jacek Jarnicki

Wrocław, 7 czerwca 2018

Spis treści

1 Wstęp

Projekt polegał na wymodelowaniu systemu transmisji danych cyfrowych używającego scramblerów oraz porównanie efektywności implementacji w wariancie addytywnym i multiplikacyjnym. Eksperyment miał na celu zbadanie która implementacja cechuje się mniejszym wskaźnikiem BER - Bit Error Rate.

2 Scramblery - teoria

2.1 Przeznaczenie

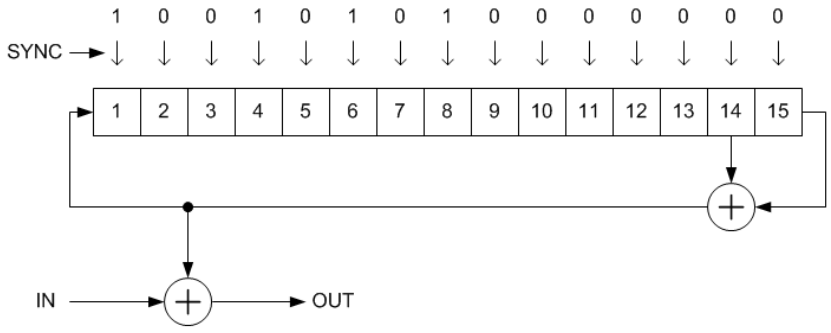
Scramblery są kluczowym elementem warstwy fizycznej systemu transmisji bezprzewodowej, poza zapewnieniem kodowania interwałowego oraz modulacji transmisji, powodami dla których używa się układów scaramblujących są:

- Dokładne odtworzenie czasu w układzie odbiorczym potrzebnego do synchronizacji transmisji, bez konieczności używania redundantnych znaków końca ramki. Rozwiązuje to problem potrzeby stosowania układu odtwarzającego taktowanie. Automatyzuje kontrolę wzmocnienia sygnału oraz pracę innych układów przetwarzających sygnał, dzięki eliminacji długich ciągów zer i jedynek.
- Zmniejszenie zakłóceń sygnału między falami nośnymi, bardziej zróżnicowane dane zapewniają rozproszenie widma sygnału co przekłada się na mniejszą wrażliwość na zakłócenia między sąsiednimi kanałami transmisji.
- Prosta metoda szyfrowania transmisji, stosowana dla sygnałów analogowych. Zarówno scrambler w urządzeniu nadawczym jak i descrambler w urządzeniu odbiorczym muszą posiadać ten sam klucz/ciąg znaków w celu poprawnego dekodowania danych.

2.2 Typy scramblerów

Addytywny (synchroniczny)

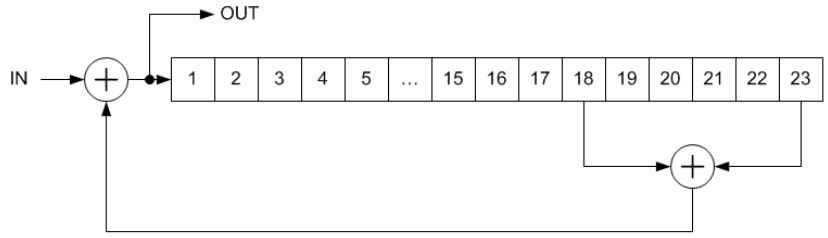
Scramblery synchroniczne przetwarzają dane wejściowe przez zastosowanie podwójnej operacji xor z dwoma najmłodszymi bitami pseudo losowego ciągu znaków oraz bitem wejścia. Po każdej operacji ciąg - klucz jest przesuwany o jedną pozycję w prawo, a wynik xor dwóch najmłodszych bitów zapisywany na pozycji pierwszej. Ta implementacja scramblera posiada wadę konieczności synchronizacji między scramblerem, a descramblerem przez zastosowanie zakodowanego słowa inicjalizującego przeładowanie ciągu znaków losowych do stanu początkowego, dzieje się to na początku każdej nowej ramki danych. Scrambler addytywny jest zarówno descramblerem bez konieczności dodatkowych modyfikacji.



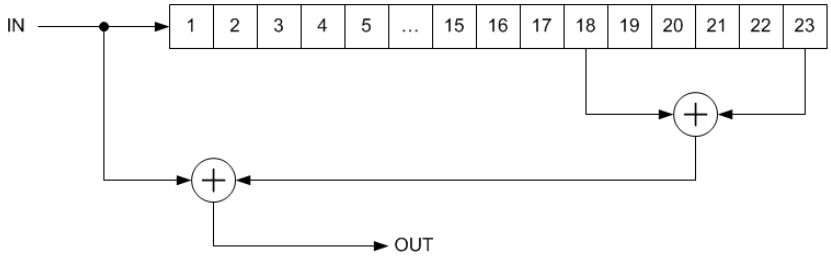
Rys. 1: Scrambler addytywny
źródło: wikipedia.org

Multiplikacyjny (samo-synchronizujący)

Scramblery multyplikacyjne, nazywane są tak ponieważ wykonują mnożenie sygnału wejściowego z funkcją przejścia ciągu znaków scramblera. W implementacji zgodnej ze standardem V.34 dokonuje operacji xor na 18 i 23 bicie klucza, jest to tzw. funkca przejścia scramblera. Wynikiem jest pierwszy bit ciągu klucza, po operacji xor z bitem wejściowym oraz wynikiem funkcji przejścia. Scrambler multiplikacyjny jest rekurencyjny, a descrambler nie, przez co ich konstrukcje różnią się. Scramblery multiplikacyjne, w przeciwieństwie do addytywnych, nie wymagają synchronizacji ciągu znaków z descramblerami.



Rys. 2: Scrambler multiplikacyjny V.34
źródło: wikipedia.org



Rys. 3: Descrambler multiplikacyjny V.34
źródło: wikipedia.org

3 Opis symulatora

3.1 Założenia

Napisany program miał za zadanie wytworzyć ciąg zer, ciąg jedynek oraz losowy ciąg bitów (wszystkie o długości 1000000). Następnie symulowane było scramblowanie dla każdego z przypadków za pomocą scramblera addytywnego DVB oraz multiplikacyjnego V34. Tak przetworzone dane przesyłane są przez kanał, który dla zadanego prawdopodobieństwa przekłamuje bity w ciągu danych. Po tej operacji wynik jest descramblowany i dla prawdopodobieństwa z przedziału $[0, 0.1]$ liczony jest stopa błędów BER, czyli stosunek ilości przekłamanych do wszystkich bitów.

3.2 Opis programów

Generacja losowego ciągu bitów

Generuje ona najpierw ciąg losowych liczb rzeczywistych z przedziału [0,1]. Liczby były losowane rozkładem równomiernym. Następnie liczby powyżej zadanego progu (w czasie eksperymentów próg = 0.5) zamieniane są na bity 1, a liczby poniżej progu zamieniane są na 0.

Scrambler DVB

Bufor scramblera ma rozmiar 15 bitów. Do funkcji przekazywana jest tablica zawierająca bity

oraz zadane jest słowo synchronizujące `[ 1 0 0 1 0 1 0 1 0 0 0 0 0 0 0 ]`. Pętla w funkcji wykonuje się tyle razy ile mamy bitów w danych wejściowych. Za każdym razem jest wykonywana operacja xor bitów nr 14 i 15, a następnie na wyniku tego działania jest wykonywana operacja xor z aktualnym bitem danych, wynikiem jest bit wyjściowy. Dodatkowo słowo synchronizujące jest przesuwane o jeden bit w prawo, a jego bitowi nr 1 przypisywana jest wcześniej obliczony xor bitów nr 14 i 15.

Descrambler DVB

Descrambler DVB działa tak samo jak scrambler. Wynika to z tego, że jest addytywny. Aby zdeszyfrować poprawnie dane należy użyć tego samego słowa synchronizującego co w scramblerze.

Scrambler V34

Bufor scramblera ma rozmiar 23 bitów. Do funkcji przekazywana jest tablica zawierająca bity. W przeciwieństwie do scramblera DVB nie używa zadanego słowa synchronizującego. Na początku bufor wypełniony jest zerami. Pętla w funkcji wykonuje się tyle razy ile mamy bitów w danych wejściowych. Za każdą iteracją pętli obliczany jest xor bitów nr 18 i 23. Następnie bufor przesuwany jest w prawo o jeden bit. Bit nr 1 będzie miał wartość działania xor aktualnych danych wejściowych oraz wcześniej obliczonego xor bitów nr 18 i 23. Taką samą wartość będzie miał bit wyjściowy.

Descrambler V34

Bufor descramblera ma rozmiar 23 bitów. Do funkcji przekazywana jest tablica zawierająca bity. Bufor na początku wypełniony jest zerami. Pętla w funkcji wykonuje się tyle razy ile mamy bitów w danych wejściowych. W pętli obliczany jest xor 18 i 23 bitu. Następnie bufor przesuwany jest o jeden bit w prawo. Jego bitem nr 1 będzie aktualny bit danych wejściowych. Bitem wynikowym jest xor aktualnego bitu danych wejściowych oraz wcześniej obliczonego xor bitów 18 i 23.

Kanał transmisyjny

Do funkcji przekazywana jest tablica bitów oraz prawdopodobieństwo z jakim pojedynczy bit może zostać przekłamany. Dla każdego bitu z danych wejściowych losowana jest liczba rzeczywista z przedziału [0,1], jeśli będzie ona mniejsza od zadanego prawdopodobieństwa to na bicie z danych wejściowych zostanie przeprowadzona negacja, jeśli będzie większa to bit zostanie zostawiony bez zmian.

4 Eksperymenty symulacyjne

4.1 Plan eksperymentów

Eksperymenty przeprowadzone są dla każdego zestawu danych — ciągi zer, jedynek oraz losowych bitów. Rozmiar danych dla którego przedstawiono wykresy wynosił 1000000 elementów.

W pierwszej kolejności ciągi danych są przekazywane do obu scramblerów. Funkcja monitor generuje wykres przedstawiający liczbę wystąpień oraz długości ciągów zer i jedynek w danych.

W celu wyznaczenia wskaźnika Bit Error Rate dane zescramblowane przepuszczane są przez kanał. Kanał z zadanym prawdopodobieństwem zamienia kolejne bity danych na przeciwne. W ramach eksperymentu prawdopodobieństwo przekłamania zmieniane jest od 0% do 10% co 0,5%.

Po wyjściu z kanału dane są descramblowane, a następnie porównywane z oryginalnym zestawem danych. Liczba błędnych bitów dzielona jest przez rozmiar danych, aby uzyskać wskaźnik BER.

4.2 Wyniki eksperymentów

Wykresy 4, 5 oraz 10, 11 przedstawiają ciąg zer po scramblingu odpowiednio DVB oraz V34. W przypadku scramblera V34 żadne bity nie zostały zamienione, ponieważ na początku w buforze scramblera znajdują się same zera. Operacja $0 \oplus 0$ zawsze da w wyniku 0, więc stan bufora nigdy się nie zmieni. Scrambler DVB zawsze zaczyna od słowa SYNC w buforze, co eliminuje ten problem.

W przypadku ciągu jedynek (wykresy 6, 7 oraz 12, 13) oba scramblery zachowują się już poprawnie. Liczba wystąpień poszczególnych długości ciągów jest podobna dla obu rodzajów scramblera. Scrambler V34 wygenerował jednak najdłuższy ciąg długości 18, natomiast DVB długości 14.

Dla danych generowanych losowo (wykresy 8, 9 oraz 14, 15) oba scramblery dały podobne wyniki.

Bit Error Rate scramblera V34 (wykres 17) w zmierzonym zakresie jest znacznie gorszy od scramblera DVB (wykres 16). Rośnie on około 2 razy szybciej.

4.3 Wnioski

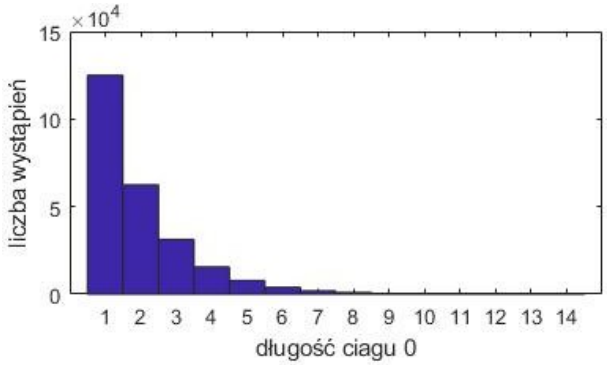
Scramblery działają poprawnie, dane wyjściowe składają się głównie z krótkich ciągów tych samych bitów. Bieżąca implementacja kanału nie jest wystarczająco dopracowana na potrzeby symulacji. Przekłamania bitów powinny odbywać się z prawdopodobieństwem, które rośnie wraz z długością ciągu tych samych bitów w danych. Dopiero wtedy widoczne stałyby się zyski wynikające z użycia scramblerów.

Obecnie z eksperymentu wynika, iż użycie scramblera skutkuje w najlepszym wypadku niepogorszeniem współczynnika BER (dla DVB BER = prawdopodobieństwu przekłamania), a w najgorszym przypadku dwukrotnym wzroście współczynnika BER (dla V34). Wzrost BER dla V34 wynika z faktu, że bity danych ładowane są do bufora, a więc przekłamany bit może skutkować późniejszym dodatkowym przekłamaniem.

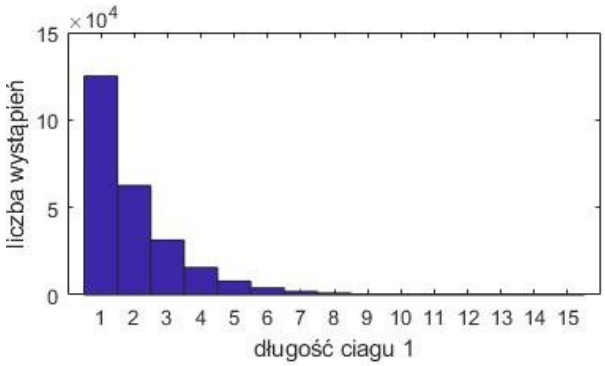
Eksperyment można więc uznać za nieudany, ponieważ użyty symulator nie uwzględnia istotnych zjawisk zachodzących podczas transmisji.

5 Wykresy

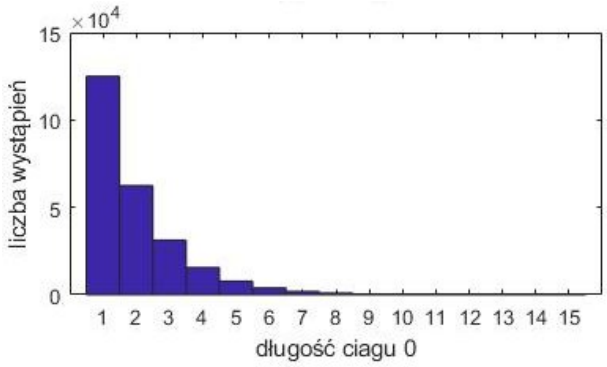
5.1 DVB



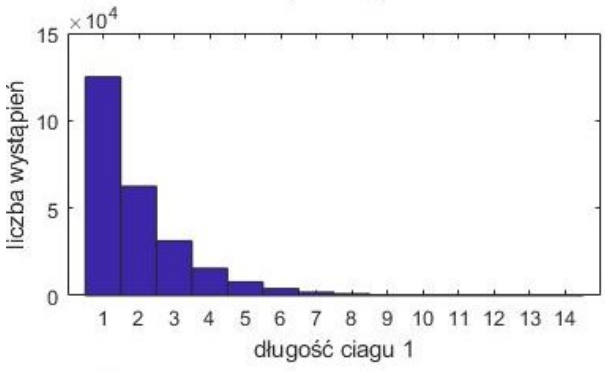
Rys. 4: Dane wejściowe: ciąg zer



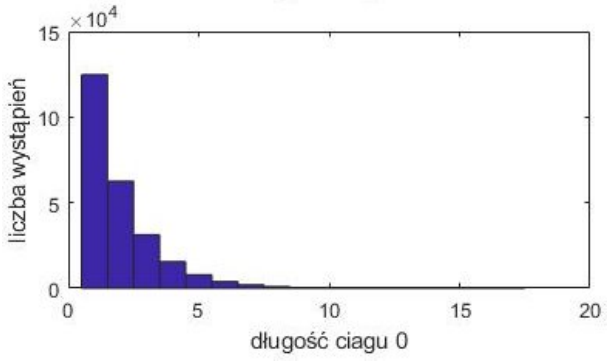
Rys. 5: Dane wejściowe: ciąg zer



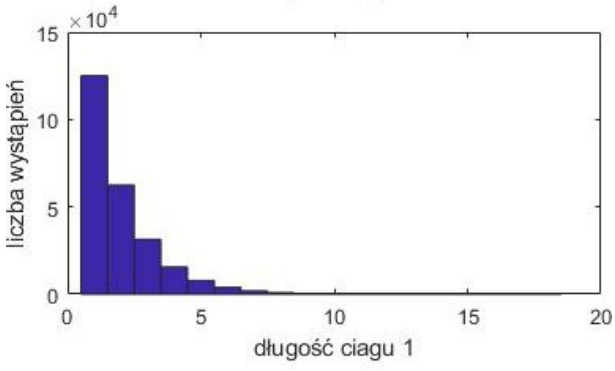
Rys. 6: Dane wejściowe: ciąg jedynek



Rys. 7: Dane wejściowe: ciąg jedynek

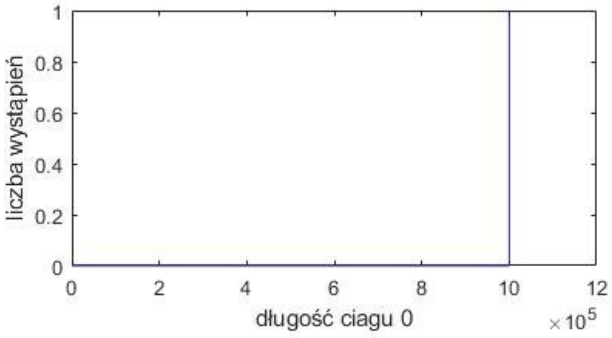


Rys. 8: Dane wejściowe: ciąg losowych bitów



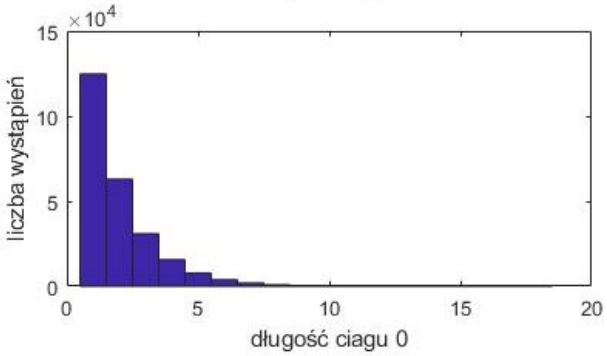
Rys. 9: Dane wejściowe: ciąg losowych bitów

5.2 V34

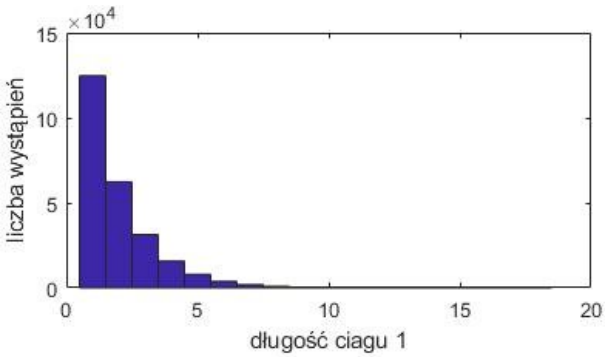


Rys. 10: Dane wejściowe: ciąg zer

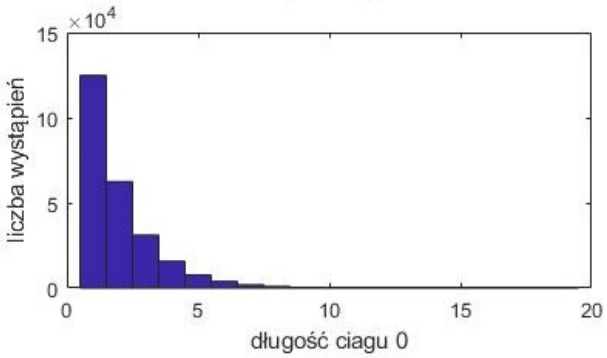
Rys. 11: Dane wejściowe: ciąg zer



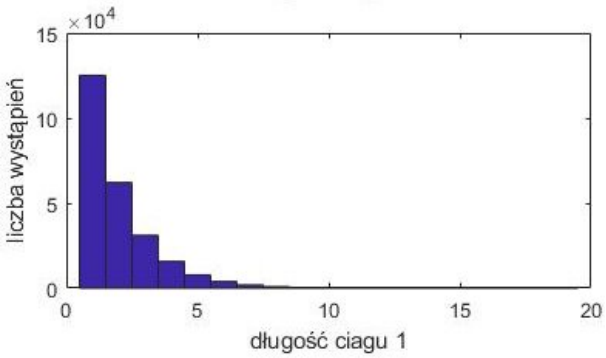
Rys. 12: Dane wejściowe: ciąg jedynek



Rys. 13: Dane wejściowe: ciąg jedynek

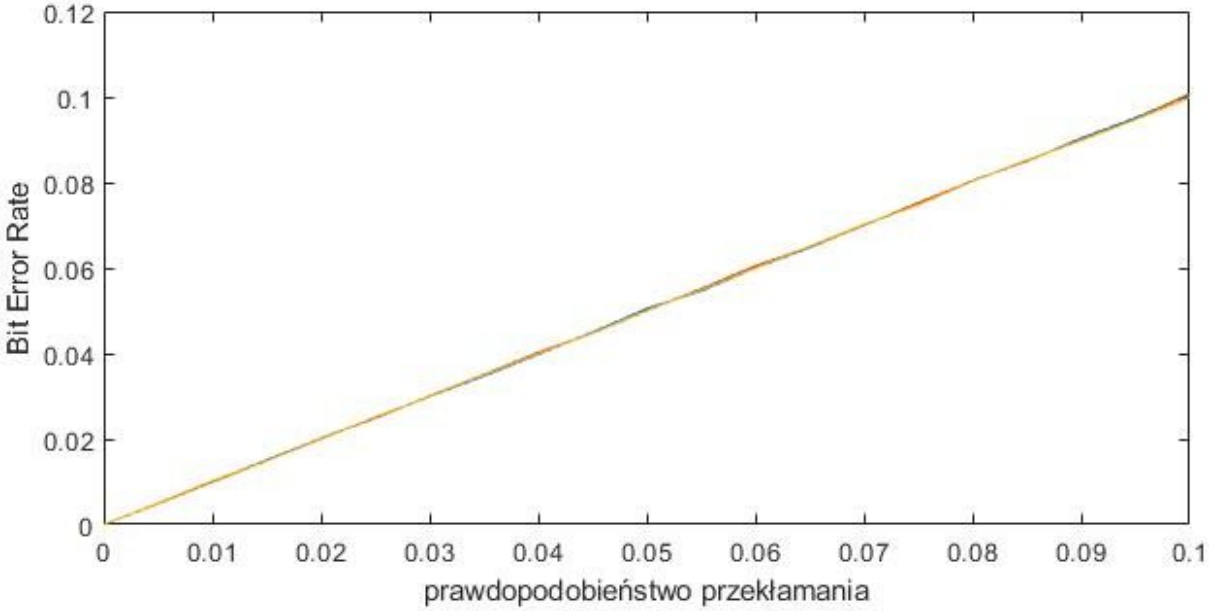


Rys. 14: Dane wejściowe: ciąg losowych bitów

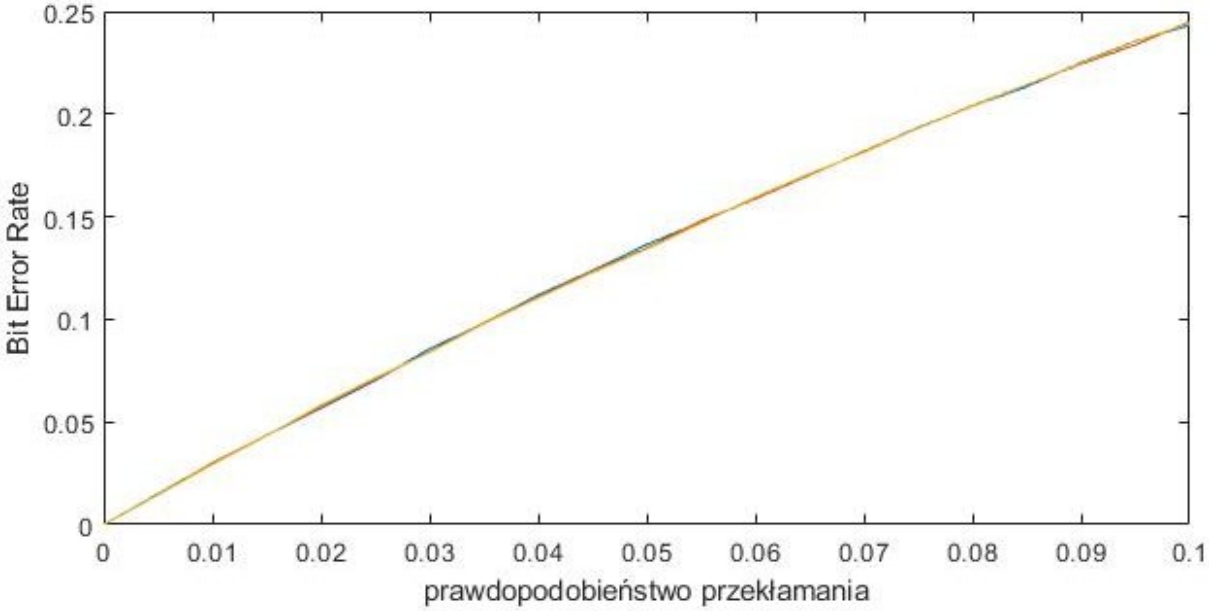


Rys. 15: Dane wejściowe: ciąg losowych bitów

5.3 Bit Error Rate



Rys. 16: BER dla DVB



Rys. 17: BER dla V34