2019

[SIEĆ FEISTELA]

Algorytmy i mechanizmy kryptograficzne

SSINF2017

Studia stacjonarne I stopnia – inżynierskie, Wydział Bezpieczeństwa Wewnętrznego,
kierunek Informatyka, specjalność Informatyka w Bezpieczeństwie..
Szczytno, 2019 rok.

Spis treści

1 Osoby biorące udział w projekcie:

- st. asp. Przemysław Jezutek – KGP Warszawa

- mł. asp. Marcin Miszkurka – KWP zs. w Radomiu

- mł.asp. Dariusz Soborski – KSP Warszawa

2 Wstęp:

Projekt został przeprowadzony w ramach zajęć Algorytmy i mechanizmy kryptograficzne .

Wykładowca prof. Mirosław Kurkowski.

Program szyfruje i deszyfruje podany 16 bitowy tekst jawny kluczem 8 bitowym przy użyciu algorytmu sieci Feistela. Tekst jawny i klucz możemy wpisać w odpowiednie pole lub używając przycisków wygenerować losowy ciąg bitów.

Aplikacja napisana jest w języku Python 3.5.3 użyto zintegrowanego środowiska programistycznego firmy JetBrains PyCharm 2018.3.5 (Community Edition) oraz graficznego interfejsu uzytkownika tkinter 8.6

3 Podstawowe definicje:

Kryptologia jest to wiedza naukowa obejmująca kryptografię i kryptoanalizę.

Kryptografia to dziedzina obejmująca kwestię związane z utajnieniem danych (w kontekście przesyłania i zabezpieczenia dostępu do danych) przed nie uprawnionym dostępem. Jako utajnienie należy rozumieć takie działanie, które powoduje że wiadomość jest trudna do odczytania(rozszyfrowania) przez osoby nie znające tzw. klucza rozszyfrującego, wiadomość będzie niezrozumiałym ciągiem znaków.

Kryptoanaliza natomiast to dziedzina kryptologii zajmująca się łamaniem szyfrów, czyli odczytywaniem zaszyfrowanych danych bez posiadania kluczy rozszyfrowujących. Dane, które poddawane będą operacjom ochrony kryptograficznej nazywamy tekstem jawnym lub wiadomością czytelną.

Kryptogramem (szyfrogramem) będziemy nazywali zaszyfrowaną postać wiadomości czytelnej.

Klucz szyfrowania to ciąg danych służących do szyfrowania wiadomości czytelnej w kryptogram za pomocą algorytmu szyfrowania. Klucz ten jest odpowiednio ustalany w fazie szyfrowania.

4 Sieć Feistela:

4.1 Horst Feistel:

Horst Feistel urodził się w 1915 roku w Niemczech w Berlinie, w 1934 roku wyemigrował się do Stanów Zjednoczonych. Ukończył Massachusetts Institute of Technology (MIT) w Cambridge na wydziale fizyki.

Podczas II wojny światowej został umieszczony w areszcie domowym. 31 stycznia 1944 r. uzyskał obywatelstwo amerykańskie. Następnego dnia otrzymał poświadczenie bezpieczeństwa i rozpoczął pracę w Centrum Badawczym Sił Powietrznych USA w Cambridge (AFCRC) na urządzeniach identyfikacyjnych Friend or Foe (IFF) . Później został zatrudniony w Laboratorium Lincolna MIT, a następnie w korporacji MITRE. W końcu przeniósł się do IBM, gdzie wspólnie z Walterem Tuchmanem uczestniczył w programie badawczym o nazwie Lucifer. Efektem pracy tego zespołu była tzw. sieć Feistela.

Wiele z obecnie stosowanych algorytmów symetrycznych stosuje pewne modyfikacje sieci Feistela. Szyfr Lucifer operował na sieci Feistla i S-box, posiadał klucz 128 bitów, cały cykl szyfrowania trwał 16 rund. Używany był w bankowości elektronicznej. Nastepcą był DES z kluczem 64 bitowym. Inne algorytmy korzystające z sieci Feistela to 3DES, Twofish, FEAL.

4.2 Mechanizm działania:

Sieć Feistel jest algorytmem blokowym, który przyjmuje postać bloków wejściowych. Dany blok podzielony jest na połówki, lewą i prawą, oznaczone odpowiednio: L i P.

L = lewa połowa bloku wiadomości jawnej

P = prawa połowa bloku wiadomości jawnej

W pierwszym kroku zastępujemy lewą połowę wartością prawej połowy(wiadomości jawnej).

Drugi kroku, zastosujemy funkcję f S na prawej połowie (wiadomości jawnej) i kluczu. Otrzymana wartość i lewa połowa(wiadomości jawnej) są przetwarzane przez operację XOR.

1. L‘ = P.

2. P‘ = L XOR f S (P,K).

Właściwości funkcji XOR:

1. 1 XOR 1 = 0 ; 0 XOR 0 = 0

2. 1 XOR 0 = 1 ; 0 XOR 1 = 1

Łatwo zauważyć, że szyfrogram otrzymany za pomocą sieci Feistel można bez trudu odszyfrować, stosując funkcję f S , która jest również używana do szyfrowanie. Mamy

L = P‘ XOR f S (K, L‘).

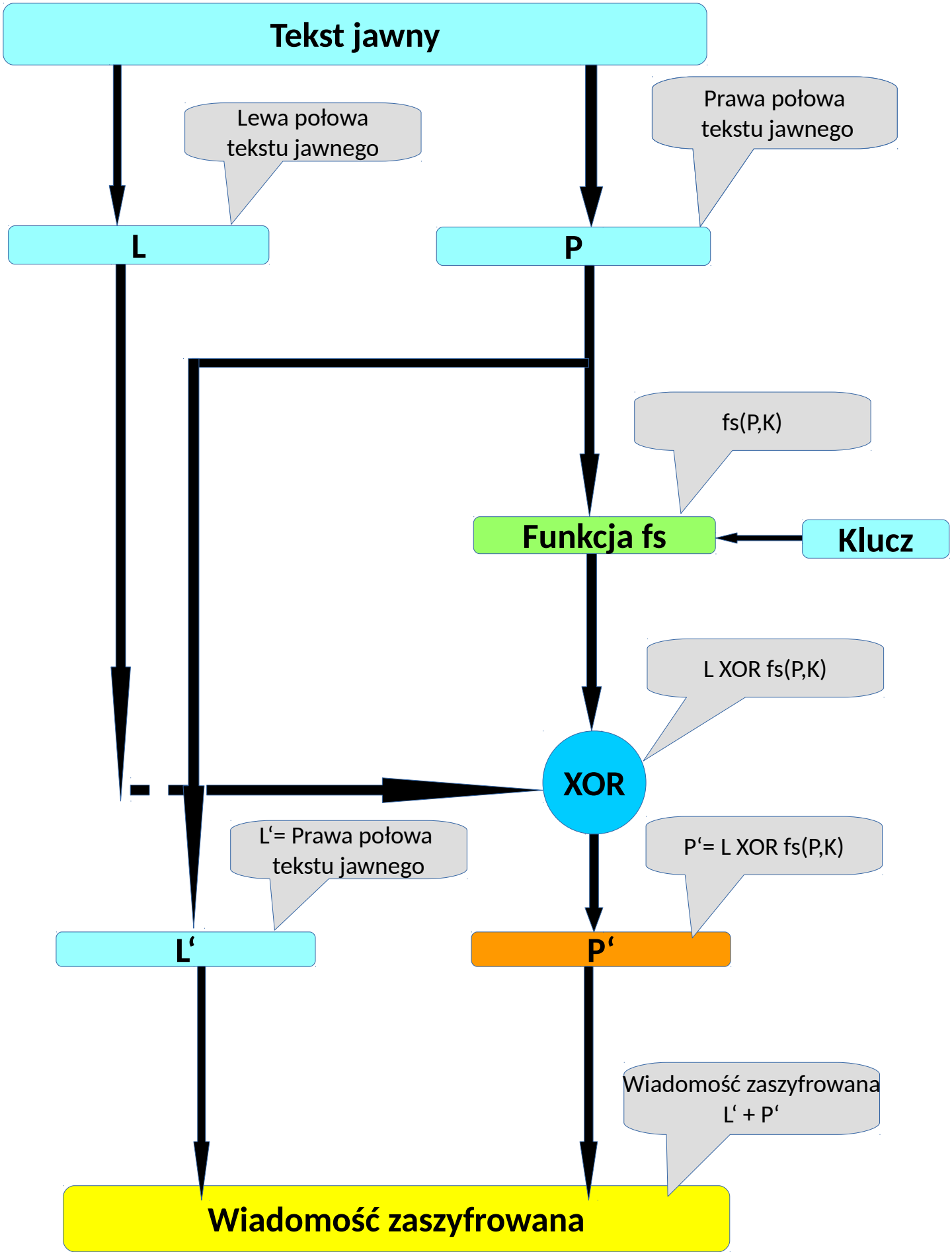
P = L‘.

Możemy zauważyć, że

$P = L'$.

Pozostawienie połowy tekstu jawnego bez zmian może budzić pewne wątpliwości. Jednakże, bezpieczeństwo algorytmu można zwiększyć przez powtórzenie całej procedury dowolną liczbę razy. Algorytmy symetryczne stosowane w praktyce zazwyczaj stosują kilka rund.

4.3 Schemat sieci Feistela:



5 Kod źródłowy(python3):

5.1 Ścieżka interpretera, kodowanie.

```
#!/usr/bin/python3
#(Shebang)ścieżka do interpretera który ma zostać użyty w celu wykonania skryptu
# -*- coding: utf-8 -*-
# ustawiamy kodowanie znaków całego pliku czyli polskie litery
```

5.2 Import modułów.

```
import random#pobranie modułu random generator liczb pseudolosowych
from tkinter import *#pobranie biblioteki  tkinter umożliwia tworzenie interfejsu
graficznego
from tkinter import messagebox as msb#pobranie modułu z funkcjami do obsługi okien
dialogowych
```

5.3 Definiowanie okna głównego.

```
okno=Tk()#tworzenie okna głównego programu
okno.title("Sieć Feistela")#ustawienie tytułu(nazwy) okna głównego
```

5.4 Funkcja losuj tekst.

```
def losuj_tekst():# definiujemy nową funkcje
    a=[]# definiujemy pustą lite
    b=""# definiujemy pusty string
    for i in range(16):# powtarzamy blok poniżej x 16
        a.append(str(random.randint(0,1)))#dodanie do listy stringów 0 lub 1
        # funkcja randint(losowanie tylko integer z zakresu(0,1)) z modułu random
    b="".join(a)#zamiana listy na ciąg znaków bez spacji
    pobierz_tekst_jawny.delete(0,END)#czyszczenie zawartości pola
    pobierz_tekst_jawny.insert(END,(b))# wysłanie losowej wiadomości do pola
```

5.5 Funkcja losuj klucz.

```
def losuj_klucz():# definiujemy nową funkcje
    c = []# definiujemy pustą lite
    d = ""# definiujemy pusty string
    for i in range(8):# powtarzamy blok poniżej x 8
        c.append(str(random.randint(0, 1)))  #dodanie do listy stringów 0 lub 1
        # funkcja randint(losowanie tylko integer z zakresu(0,1)) z modułu random
    d="".join(c)  # zamiana listy na ciąg znaków bez spacji
    pobierz_klucz.delete(0,END)#czyszczenie zawartości danego pola
    pobierz_klucz.insert(END,(d))# wysłanie losowego klucza do pola
```

5.6 Funkcja szyfrująca.

```
def szyfruj():# definiujemy nową funkcje
    a = (pobierz_tekst_jawny.get())#  a i b aby skrócić poniższe zapisy
    b = (pobierz_klucz.get())
    try:#obsługa błędów
        if len(a) == 16 and int(a, 2) <= (2 ** 16):# sprawdzenie długości tekstu
jawnego
            if len(b) == 8 and int(b, 2) <= (2 ** 8):# sprawdzenie długości klucza
                msb.showinfo("ok","Podano poprawny format\n wiadomość i klucza ")#
okno informacyjne
                lewa = int(a[:8], 2)#pierwsza połowa stringu a, od początku 0 do 8
pozycji (bez 8 strażnik)
                #2 oznacz zamianę powstałego stringu na liczbę binarną a int
zamienia ją od razu na liczbę dziesiętną
                prawa = int(a[8:], 2)#jw
                klucz = int(b, 2)# cały  string zamiana na liczbę binarną a później
                                 #na dziesiętną
                lewa_zaszyfr = (prawa)
                prawa_zaszyfr = lewa ^ (prawa ^ klucz)#xor na liczbach dziesiętnych
                lewa_zaszyfr = bin(lewa_zaszyfr)#konwersja liczby dziesiętnej na
binarną wynik to znów string
                prawa_zaszyfr = bin(prawa_zaszyfr)# jw
                lewa_zaszyfr = lewa_zaszyfr[2:]#od 2 pozycji bo pierwsze 2 to
oznaczenie 0b ze jest liczba binarna
                if len(lewa_zaszyfr) < 8: # jeśli mniej niż 8 znaków uzupełniamy od
lewej 0 aby było 8 znaków
                    lewa_zaszyfr = "0" * (8 - int(len(lewa_zaszyfr))) +
(lewa_zaszyfr)
                prawa_zaszyfr = prawa_zaszyfr[2:]# jw
                if len(prawa_zaszyfr) < 8: #mniej niż 8 znaków
                    prawa_zaszyfr = "0" * (8 - int(len(prawa_zaszyfr))) +
(prawa_zaszyfr)
                wynik = lewa_zaszyfr + prawa_zaszyfr# konkatenacja stringów
                wynik = "wiadomość po zaszyfrowaniu \n" + wynik
                pokaz_tekst_zaszyfrowany["text"]=""# pusty ciąg znaków
                pokaz_tekst_zaszyfrowany["text"] = wynik#przekierowujemy string
wynik jako text do okna
            else:
                msb.showerror("Błąd","Nieodpowiednia ilość \nznakóœ klucza. Proszę
poprawić")
                #okno z komunikatem o błędzie podanego klucza
        else:
            msb.showerror("Błąd","Nieodpowiednia ilość \nznaków wiadomości jawnej.
\nProszę poprawić")
            # okno z komunikatem o błędzie podanej wiadomości
    except ValueError:
        msb.showerror("Błąd","Podajemy tylko 0 lub 1 nie litery.\nProszę poprawic")
```

5.7 Funkcja deszyfrująca.

```
def deszyfruj():# definiujemy nową funkcje
    a = (pobierz_szyfr.get())# a i b aby skrócić poniższe zapisy
    b = (pobierz_klucz2.get())
    try:#obsługa błędów
        if len(a) == 16 and int(a, 2) <= (2 ** 16):#sprawdzenie długości szyfru
            if len(b) == 8 and int(b, 2) <= (2 ** 8):# sprawdzenie długości klucza
                msb.showinfo("ok","Podano poprawny format\n wiadomość i klucza ")#
okno informacyjne
                lewa = int(a[:8], 2)#pierwsza połowa stringu a, od początku 0 do 8
pozycji (bez 8 strażnik)
                #2 oznacz zamianę powstałego stringu na liczbę binarną a int
zamienia ją od razu na liczbę dziesiętną
                prawa = int(a[8:], 2)# jw
                klucz = int(b, 2)# cały klucz string zamiana na liczbę binarną a
później na dziesiętną
                prawa_odszyfr = (lewa)
                lewa_odszyfr = prawa ^ (lewa ^ klucz)#xor na liczbach dziesiętnych
                lewa_odszyfr = bin(lewa_odszyfr)#konwersja liczby dziesiętnej na
binarną wynik to znów string
                prawa_odszyfr = bin(prawa_odszyfr)# jw
                lewa_odszyfr = lewa_odszyfr[2:]#od 2 pozycji bo pierwsze 2 to
oznaczenie 0b ze jest liczba binarna
                if len(lewa_odszyfr) < 8:  # mniej niż 8 znaków dodajemy tyle 0 ile
znaków brakuje
                    lewa_odszyfr = "0" * (8 - int(len(lewa_odszyfr))) +
(lewa_odszyfr)
                prawa_odszyfr = prawa_odszyfr[2:]
                if len(prawa_odszyfr) < 8:# mniej niż 8 znaków dodajemy tyle 0 ile
znaków brakuje
                    prawa_odszyfr = "0" * (8 - int(len(prawa_odszyfr))) +
(prawa_odszyfr)
                wynik = lewa_odszyfr + prawa_odszyfr# konkatenacja stringów
                wynik = "wiadomość po odszyfrowaniu \n" + wynik
                pokaz_tekst_odszyfrowany["text"]=""# pusty ciąg znaków
                pokaz_tekst_odszyfrowany["text"] = wynik#przekierowujemy string
wynik jako text do okna
            else:
                msb.showerror("Błąd","Nieodpowiednia ilość \nznaków klucza. Proszę
poprawić")
                #okno z komunikatem o błędzie podanego klucza
        else:
            msb.showerror("Błąd","Nieodpowiednia ilość \nznaków wiadomości
zaszyfrowanej. \nProszę poprawić")
            # okno z komunikatem o błędzie podanej ilości znaków
    except ValueError:# okno z komunikatem o wpisanym błędnym typie
        msb.showerror("Błąd","Podajemy tylko 0 lub 1 nie litery.\nProszę poprawic")
```

5.8 Funkcja zamykająca program.

```
def koniec():# definiujemy nową funkcje
    if msb.askokcancel("Pytanie", "Czy na pewno kończymy pracę"):
        # okno dialogowe z przyciskami ok i anuluj - zwraca prawdę, gdy ok jest
wciśnięte
        msb.showinfo("Info","Koniec, żegnam")
        okno.destroy()#zamyka nasze okno i cały program
    else:
        msb.showinfo("Info","Ok, popracujmy dalej")#powrót do okna programu
```

5.9 Funkcja czyszcząca pola.

```
def kasowanie():# czyszczenie zawartości poszczególnych pól i okien
    pobierz_tekst_jawny.delete(0, END)
    pobierz_klucz.delete(0,END)
    pobierz_szyfr.delete(0,END)
    pobierz_klucz2.delete(0,END)
    pokaz_tekst_zaszyfrowany["text"]=""
    pokaz_tekst_odszyfrowany["text"] = ""
```

5.10 Definiowanie widgetów.

5.10.1 Definiowanie etykiet.

```
#definiowanie etykiet nazwy kolory
tekst_jawny = Label(okno, text ="Wpisz wiadomość 16 bitów",bg="silver")
klucz = Label(okno, text = "Wpisz klucz,8 bitów",bg="silver")
pokaz_tekst_zaszyfrowany = Label(okno)
tekst_zaszyfrowany = Label(okno,text = "Wiadomość zaszyfrowana",bg="silver")
szyfr = Label(okno, text ="Wpisz wiadomość zaszyfrowaną",bg="silver")
klucz2 = Label(okno, text = "Wpisz klucz,8 bitów",bg="silver")
pokaz_tekst_odszyfrowany = Label(okno)
```

5.10.2 Definiowanie pól do wpisywania.

```
#definiowanie pola do wpisywania długość kolory
pobierz_tekst_jawny = Entry(okno,width=16,bg="yellow")
pobierz_klucz = Entry(okno,width=8,bg="yellow")
pobierz_szyfr = Entry(okno,width=16,bg="yellow")
pobierz_klucz2 = Entry(okno,width=8,bg="yellow")
```

5.10.3 Aktywacja etykiet i pól.

```
#aktywacja położenia etykiet i pól wpisywania tekstu wiersze i kolumny(metoda grid)
tekst_jawny.grid(row = 0)
klucz.grid(row = 1)
pobierz_tekst_jawny.grid(row = 0, column = 1)
pobierz_klucz.grid(row = 1,column = 1)
pokaz_tekst_zaszyfrowany.grid(row=4, column=1)
szyfr.grid(row = 6)
klucz2.grid(row = 7)
pobierz_szyfr.grid(row = 6, column = 1)
pobierz_klucz2.grid(row = 7, column = 1)
pokaz_tekst_odszyfrowany.grid(row=8, column=1)
```

5.10.4 Definiowanie przycisków

```
#definiowanie przycisków nazwy,kolory, położenie,przypisanie funkcji
przycisk1 = Button(okno,text = "Szyfruj wiadomość",bg="blue", command = szyfruj)
przycisk1.grid(row = 4, column = 0)
przycisk2 = Button(okno, text = "Deszyfruj widomość",bg="blue", command =
deszyfruj)
przycisk2.grid(row = 8, column = 0)
przycisk3 = Button(okno, text = "Losowanie wiadomości",bg="green", command =
losuj_tekst)
przycisk3.grid(row = 2, column = 0)
przycisk4 = Button(okno, text = "Losowanie klucza",bg="green", command =
losuj_klucz)
przycisk4.grid(row = 2, column = 1)
przycisk5 = Button(okno,text = "Zamknij program",bg="red", command = koniec)
przycisk5.grid(row = 9, column = 1)
przycisk6 = Button(okno,text = "Wyczyść wszystkie pola",bg="orange", command =
kasowanie)
przycisk6.grid(row = 9)
```

5.10.5 Uruchomienie pętli okna programu

```
okno.mainloop()#włączamy pętle okna głównego"
```

6 Obsługa programu.

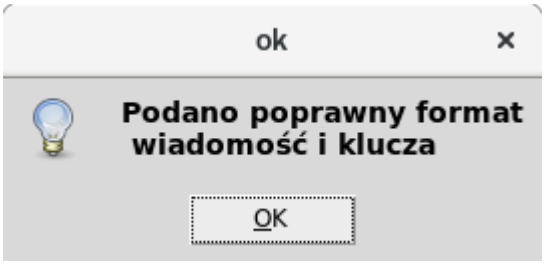
6.1 Puste okno programu:

- Po uruchomieniu programu na ekranie zobaczymy okno

6.2 Szyfrowanie wiadomości:

- Wpisujemy tekst jawny(16 bitów) i klucz(8 bitów), możemy użyć przycisków „Losowanie wiadomości“ i „Losowanie klucza“:

- Po naciśnięciu przycisku „Szyfruj wiadomość“, program weryfikuje podane dane. Jeśli dane są wpisane poprawnie, wyświetlane jest następujące okno:

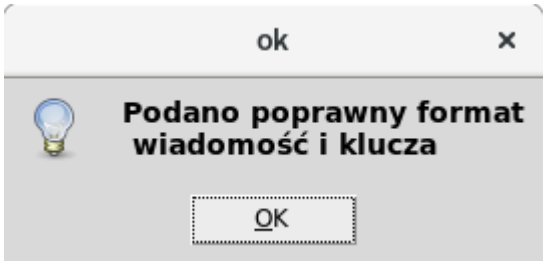


- Wybranie „ok“, wyświetlony zostaje tekst zaszyfrowany:

6.3 Odszyfrowanie:

- Wpisujemy wiadomość zaszyfrowaną i klucz:

- Po naciśnięciu przycisku „Deszyfruj wiadomość“, program weryfikuje podane dane. Jeśli dane są wpisane poprawnie, wyświetlane jest następujące okno:

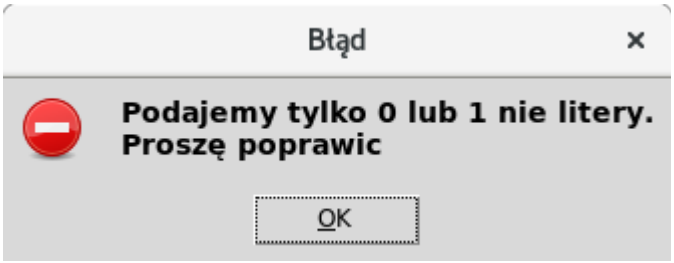


- Wybranie „ok“, wyświetlona zostaje wiadomość odszyfrowana :

6.4 Komunikaty o błędach.

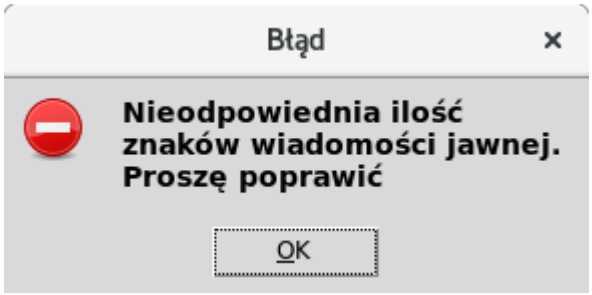
6.4.1 Błąd typu wpisanych danych.

- Wpisanie w oknie tekstu jawnego, wiadomości zaszyfrowanej lub klucz innego znaku niż 0 1 spowoduje wyświetlenie okna z komunikatem:

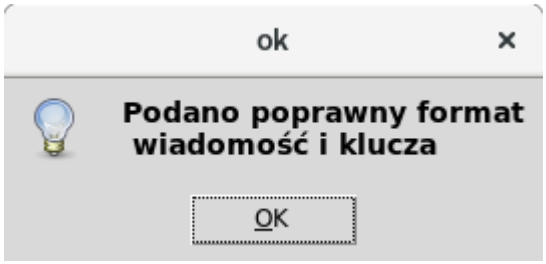


6.4.2 Błąd ilości wpisanych danych.

- wpisanie nieodpowiedniej liczby 0,1 spowoduje wyświetlenie komunikatu:



- Po poprawieniu błędnych danych, wyświetlone zostanie komunikat:



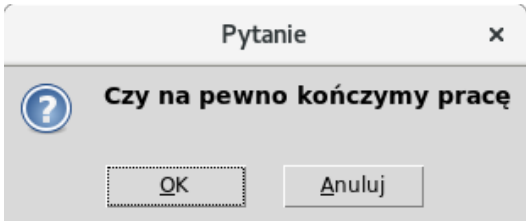
6.5 Czyszczenie okien i wyjście z programu.

6.5.1 Czyszczenie okien:

- Naciśnięcie przycisku „Wyczyść wszystkie pola“, usunie zawartość wszystkich okien:

6.5.2 Wyjście z programu:

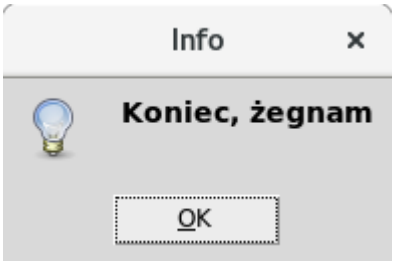
- Naciśnięcie przycisku „Zamknij program“ wyświetli okno. Mamy 2 możliwości wyboru :



1. Naciśnięcie przycisku „Anuluj“, wyświetli komunikat:

- Wybranie „ok“, wrócimy do programu.

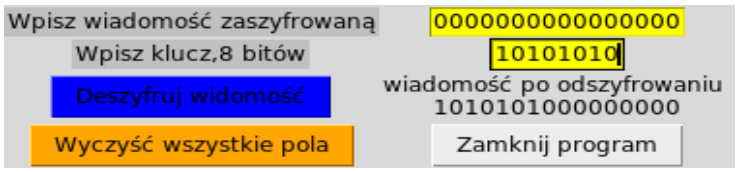
2. Naciśnięcie przycisku „OK“ w oknie Pytanie czy na pewno kończymy pracę, wyświetli okno:

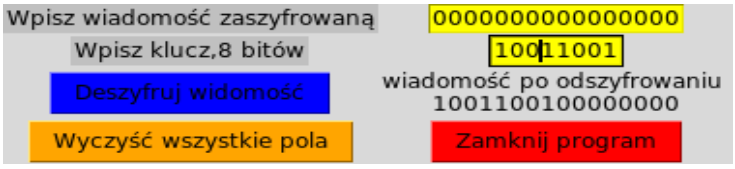


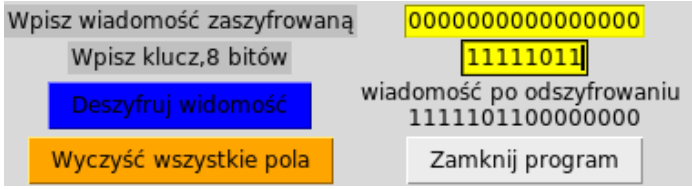
- Wybranie „ok“ zamyka program.

7 Sprawdzenie niestandardowych szyfrogramów.

7.1 Szyfrogram 0000000000000000:

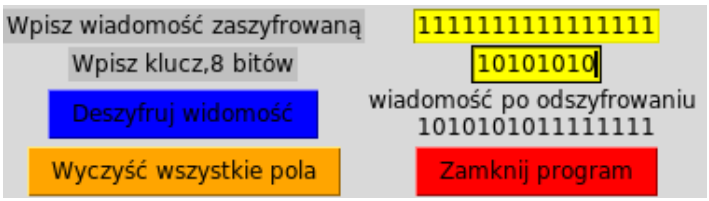






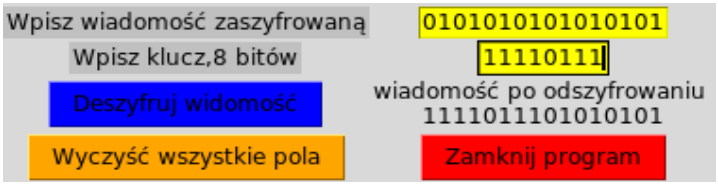
Jeśli szyfrogram zawiera same 0 wynikiem będzie pierwsze 8 znaków to klucz jakiego użyliśmy do deszyfracji a druga połowa to oczywiście osiem 0.

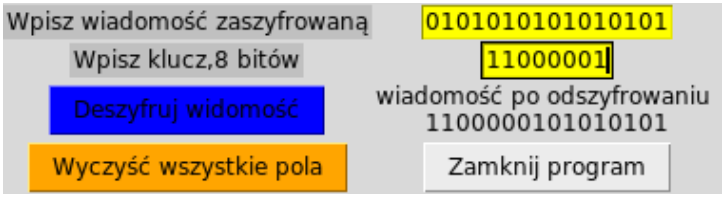
7.2 Szyfrogram 1111111111111111:



Jeśli szyfrogram zawiera same 1 wynikiem również będzie pierwsze 8 znaków to klucz jakiego użyliśmy do deszyfracji a druga połowa to oczywiście osiem 1.

7.3 Szyfrogram 0101010101010101:





Jeśli szyfrogram składa się, z ciągu: 0101010101010101 wynikiem również będzie pierwsze 8 znaków to klucz jakiego użyliśmy do deszyfracji a druga połowa to 01010101.

8 Literatura

http://wazniak.mimuw.edu.pl

http://encyklopedia.naukowy.pl

https://en.wikipedia.org

http://www.obliczeniowo.com.pl

http://effbot.org/tkinterbook/

https://pythonspot.com/tk-message-box/

C. Kościelny M. Kurkowski M. Srebrny „Modern Cryptography Primer“