Kryptografia i kryptoanaliza

Laboratorium 2

Michał Łaskawski

Zadanie 1

Napisz program (w języku: `C++`, `RUST`, `Python`) implementujący algorytm szyfru przesuwnego (szyfr Cezara).

1. Tekst jawny powinien być importowany do programu z pliku tekstowego, którego nazwa określona powinna być po zdefiniowanym argumencie / fladze: `-i`.
2. Wynik pracy programu powinien być eksportowany do pliku tekstowego, którego nazwa określona powinna być po zdefiniowanym argumencie / fladze: `-o`.
3. Klucz powinien być określany za pomocą parametru / flagi `-k`.
4. Tryb pracy programu powinien być określony poprzez flagi: `-e` dla procesu szyfrowania, `-d` dla procesu deszyfrowania.

Przykład wywołania programu w celu zaszyfrowania tekstu:

```
./program -e -k klucz.txt -i tekst_jawny.txt -o szyfrogram.txt
```

Przykład wywołania programu w celu odszyfrowania tekstu:

```
./program -d -k klucz.txt -i szyfrogram.txt -o tekst_odszyfrowany.txt
```

Uwagi:

- Kolejność argumentów powinna być dowolna.
- Odczytany tekst jawny, przed dalszym przetwarzaniem, powinien być zamieniony do postaci składającej się tylko z dużych liter. Ponadto z tekstu powinny być usunięte wszystkie znaki, które nie są literami, np: odstępy, przecinki, kropki itp.

Zadanie 2

Rozbuduj program z poprzedniego zadania poprzez implementację ataku typu `brute-force` na szyfrogram wygenerowany przy pomocy algorytmu przesuwnego.

1. Algorytm powinien być wyzwalany po użyciu flagi `-a` z parametrem `bf`.

   Przykład wywołania programu:

   ```
   ./program -a bf -i szyfrogram -o tekst_odszyfrowany
   ```

Uwagi:

- Program w celu klasyfikacji wyniku działania algorytmu, powinien wykorzystywać test χ^2 na poziomie istotności 0.05 (patrz ostatnie zadanie z poprzedniej instrukcji).
- Do wyznaczenia wartości krytycznej, decydującej o odrzuceniu hipotezy zerowej (odszyfrowany tekst jest tekstem w języku angielskim), należy użyć funkcji `gsl_cdf_chisq_Pinv(p, df)` z biblioteki `gsl` (`C++`).
  - Funkcja ta oblicza dystrybuantę (`CDF`) rozkładu χ^2 a następnie zwraca wartość zmiennej losowej χ^2, dla której dystrybuanta ta przyjmuje podaną wartość prawdopodobieństwa (pierwszy argument funkcji).
    - Przykład, jeżeli podane prawdopodobieństwo $p = 0.95$, to funkcja zwróci wartość charakterystyki χ^2, dla której 95% obszaru pod krzywą rozkładu znajduje się na lewo od tej wartości.
  - Drugim argumentem funkcji jest liczba stopni swobody. Jest to wartość, która odnosi się do liczby niezależnych zmiennych, które mogą swobodnie przyjmować wartości w określonym zbiorze danych (liczba wartości w próbie, które mogą się zmieniać bez narzucania ograniczeń przez inne zmienne).
    - W kontekście odszyfrowywania tekstu zaszyfrowanego szyfrem Cezara z wykorzystaniem metody `b-f`, liczba stopni swobody odnosi się do liczby możliwych przesunięć klucza. Ponieważ szyfr Cezara, polega na przesunięci każdej litery o stałą wartość, to liczba tych możliwych przesunięć zależy od ilości liter w alfabecie. Dla alfabetu o n znakach, liczba stopni swobody będzie wynosiła $n - 1$ ponieważ pomijane jest przesunięcie o 0 znaków.

    * W kontekście odszyfrowywania tekstu zaszyfrowanego szyfrem afinicznym z wykorzystaniem metody `b-f`, liczba stopni swobody, również określona jest przez liczbę możliwych wartości klucza. Jednakże tym razem liczba ta jest wynikiem iloczynu wartości dwóch komponentów klucza, to jest $a \times b = 12 \times 26 = 312$.

- Korzystając z języka `RUST`, do wyznaczenia wartości krytycznej, należy użyć obiektu utworzonego przy pomocy konstruktora rozkładu `ChiSquared`, wywołując metodę `new(df)`, gdzie `df` to liczba stopni swobody. Następnie na tak utworzonym obiekcie, należy wywołać metodę `inverse_cdf(p)`, gdzie `p` to wartość prawdopodobieństwa. Narzędzia te dostępne sa po dołączeniu do projektu biblioteki `statrs` i modułu `distribution`.
- Korzystając z język `Python`, wartość krytyczną można wyznaczyć używając obiektu `chi2` z modułu `stats` biblioteki `scipi`. Na obiekcie tym należy wywołać metodę `ppf(p, df)`.

Zadanie 3

Napisz program analogiczny do programu z zadania 1, który tym razem implementuje szyfr afiniczny.

Zadanie 4

Rozbuduj program z poprzedniego zadania poprzez implementację ataku typu `brute-force` na szyfrogram zaimplementowany przy pomocy szyfru afinicznego. Sposób pracy z programem powinien być analogiczny do pracy z programem z zadania 2.