PODSTAWY KRYPTOGRAFII

Zadanie 1, zestaw V

Wykonali:

Michał Ferdzyn 242383

Artur Grzybek 242399

Cel zadania

Napisać program szyfrujący/deszyfrujący dane wprowadzone przez użytkownika lub z pliku wykorzystując algorytm trzykrotnego DESa.

3DES

Szyfr 3DES jest blokowym szyfrem symetrycznym, zbudowanym na bazie DES, który polega na trzykrotnym przetworzeniu wiadomości tym właśnie algorytmem. Blok 64 bitów tekstu jawnego pojawia się na jednym końcu algorytmu, a blok 64 bitów wychodzi na jego drugim końcu. Zarówno podczas szyfrowania jak i deszyfrowania wykorzystuje się ten sam algorytm (z wyjątkiem różnic w operowaniu kluczem).

Opis algorytmu

Generowanie 16 podkluczy:

1. Klucz ma długość 56 bitów (zwykle jest liczną zapisaną za pomocą 64 bitów, przy czym każdy co ósmy bit jest bitem parzystości, który jest pomijany).
2. Ów 64 bitowy klucz przechodzi permutacje względem tablicy PC1, tym samym powstaje zredukowany klucz 56 bitowy.

```
final byte[] PC1 =
        {
                57, 49, 41, 33, 25, 17, 9,
                1, 58, 50, 42, 34, 26, 18,
                10, 2, 59, 51, 43, 35, 27,
                19, 11, 3, 60, 52, 44, 36,
                63, 55, 47, 39, 31, 23, 15,
                7, 62, 54, 46, 38, 30, 22,
                14, 6, 61, 53, 45, 37, 29,
                21, 13, 5, 28, 20, 12, 4
        };
```

3. Otrzymany klucz dzielmy na dwie części, gdzie każda ma 28 bitów.
4. Połowy klucza są przesuwane w lewo o jeden bądź dwa bity, zależnie od numeru cyklu (zależność ta jest opisana w naszej implementacji jako tabela Shifts).

```
final byte[] SHIFTS =  { 1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1 };
```

Tym samym tworzymy 16 par bloków, każda para bloków jest tworzona z poprzedniego za pomocą wspomnianych przesunięć.

5. Po wykonaniu tego przesunięcia jest wybieranych 48 z 56 bitów. Ponieważ w operacji tej dokonuje się zmiany w porządku występowania bitów jest również wyboru podciągu bitów, nosi ona nazwę permutacji z kompresją. Operacja ta dostarcza 48-bitowego podciągu (permutacja z kompresją odbywa się według tablicy PC-2).

```
final byte[] PC2 =
        {
                14, 17, 11, 24, 1, 5,
                3, 28, 15, 6, 21, 10,
                23, 19, 12, 4, 26, 8,
                16, 7, 27, 20, 13, 2,
                41, 52, 31, 37, 47, 55,
                30, 40, 51, 45, 33, 48,
                44, 49, 39, 56, 34, 53,
                46, 42, 50, 36, 29, 32
        };
```

6. Tym samym za każdym razem generowany jest inny 48 bitowy podklucz dla każdego z 16 cykli algorytmu DES.

Algorytm kodowania:

1. Dane są dzielone na 64 bitowe bloki, w razie potrzeby końcowy zostaje dopełniony zerami.
2. Następnie zostaje przeprowadzona permutacja początkowa na wszystkich blokach. Wykorzystana do tego zostaje tablica bajtów:

```
final byte[] IP = new byte[] {
            58, 50, 42, 34, 26, 18, 10, 2,
            60, 52, 44, 36, 28, 20, 12, 4,
            62, 54, 46, 38, 30, 22, 14, 6,
            64, 56, 48, 40, 32, 24, 16, 8,
            57, 49, 41, 33, 25, 17, 9, 1,
            59, 51, 43, 35, 27, 19, 11, 3,
            61, 53, 45, 37, 29, 21, 13, 5,
            63, 55, 47, 39, 31, 23, 15, 7};
```

3. Następnie blok wejściowy jest dzielony na lewą połowę i prawą połowę, każda o długości 32 bity.
4. Kolejno jest wykonywane 16 cykli jednakowych operacji, nazwanych funkcjami Feistela, w czasie których dane są łączone z kluczem (implementacje funkcji Feistela przedstawimy w osobnym punkcie).
5. Po szesnastym cyklu lewa i prawa strona są zamieniane miejscami i łączone. Otrzymujemy zaszyfrowany 64 bitowy blok.
6. Otrzymany blok przechodzi permutacje końcową, wykonaną według tablicy:

```
    final byte[] IPplus = new byte[] {
            40, 8, 48, 16, 56, 24, 64, 32,
            39, 7, 47, 15, 55, 23, 63, 31,
            38, 6, 46, 14, 54, 22, 62, 30,
            37, 5, 45, 13, 53, 21, 61, 29,
            36, 4, 44, 12, 52, 20, 60, 28,
            35, 3, 43, 11, 51, 19, 59, 27,
            34, 2, 42, 10, 50, 18, 58, 26,
            33, 1, 41, 9, 49, 17, 57, 25};
```

7. Algorytm deszyfrowania działa analogicznie, jedyna różnica to odwrotna kolejność używanych podkluczy (od 16 do 1).

Funkcja Feistela:

1. Lewą stronę bloku oznaczymy jako L, prawą jako R, n będzie numerem danej rundy (n należy do <1,16>), K kolejnym podkluczem, natomiast f jako funkcje Feistela.
2. Dla kolejnych 16 iteracji korzystamy ze wzorów: $L_n = R_{n-1}$ oraz $R_n = L_{n-1} XOR\ f(R_{n-1}, K_n)$.
3. Rozszerzamy blok R_{n-1} do 48 bitów za pomocą funkcji rozszerzBlok (). Blok rozszerzony zwracany jest jako tablica bajtów.
4. Wykonujemy operacje XOR przez K_n na otrzymanej tablicy.
5. Otrzymaną tablicę dzielimy na 8 części, w każdej z nich pierwszy i ostatni bit kodują numer kolumny, a środkowe cztery numer wiersza w tablicy podstawienieBox.
6. Odczytujemy wartości z wyznaczonych pól w tablicy podstawienieBox i zamieniamy je na postać binarną umieszczając je po kolei w tablicy 64-bitowej.
7. Dokonujemy permutacji otrzymanej tablicy zgodnie z tablicą permutacja_pBloku:

```
final byte[] permutacja_pBloku =
            {
                    16, 7, 20, 21,
                    29, 12, 28, 17,
                    1, 15, 23, 26,
                    5, 18, 31, 10,
                    2, 8, 24, 14,
                    32, 27, 3, 9,
                    19, 13, 30, 6,
                    22, 11, 4, 25
            };
```

Szyfr 3DES powstaje według powyższego algorytmu w taki sposób:

1. Szyfrowanie przy pomocy pierwszego klucza
2. Deszyfrowanie przy pomocy drugiego klucza.
3. Szyfrowanie przy pomocy trzeciego klucza.

Deszyfrowanie przebiega analogicznie:

1. Deszyfrowanie przy pomocy trzeciego klucza.
2. Szyfrowanie przy pomocy drugiego klucza.
3. Deszyfrowanie przy pomocy pierwszego klucza.

Implementacja

Program został przygotowany w języku Java, natomiast interfejs graficzny został przygotowany przy pomocy JavaFX. Kod został podzielony na 6 plików, z których dwa służą do implementacji GUI, pozwalają nam na komunikacja z użytkownikiem. Klasa DES kolejno przedstawia algorytm, który wyżej został przez nas opisany. Klasa TripleDES natomiast dziedziczy z klasy DES i zawiera właściwą implementacje szyfrowania oraz deszyfrowania algorytmu trzykrotnego DESa. Wyodrębniliśmy również klasę FileOperations, która zawiera funkcje wczytywania oraz zapisywania związanymi z plikiem tekstowym. Klasa BitOperations natomiast zawiera metody, z których korzystaliśmy w przypadku pewnych przesunięć oraz

operacji na bitach. Użytkownik poprzez stworzoną przez nas aplikacje okienkową może samodzielnie wpisać klucze, z których chce skorzystać lub losowo je wygenerować. Umożliwione zostało samodzielne wprowadzanie tekstu jawnego, który ma zostać zaszyfrowany lub też wczytanie owego tekstu z wybranego pliku (analogicznie można wpisać szyfr, który chcemy odszyfrować bądź też wczytać go z pliku). Zarówno szyfr jak i odszyfrowany tekst wyświetlany jest w polach tekstowych, ale jest też możliwość zapisania owej zawartości do pliku. Zapewniona została również funkcjonalność szyfrowania oraz odszyfrowywania plików binarnych, takich jak .pdf oraz .jpg

Wnioski

- Dzięki zastosowanej kolejności szyfrowania (szyfrowanie, deszyfrowanie, szyfrowanie) algorytm 3DES jest kompatybilny wstecz z algorytmem DES. Zastosowanie tego samego klucza do 3 operacji pozwala odszyfrować dane zaszyfrowane starszą wersją algorytmu.
- Algorytm 3DES jest podatny na ataki typu meet in the middle, jako że używa wielokrotnie tego samego algorytmu szyfrującego.
- Algorytm 3DES zwiększa odporność na ataki brute force, w porównaniu do DES.

Bibliografia

1. http://www.crypto-it.net/pl/symetryczne/des.html#Maths
2. https://ftims.edu.p.lodz.pl/course/view.php?id=1801 ,,WYKŁAD KRYPTOGRAFIA SYMETRYCZNA”
3. https://www.tutorialspoint.com/cryptography/data_encryption_standard.htm