Bezpieczeństwo Systemów Komputerowych

Apliikacja szyfrująca pliki

Dokumentacja Techniczna

Maciej Nawrocki

Filip Plombon

# Opis projektu

Program napisany został w języku Java. Pliki podawane na wejście programu szyfrowane są algorytmem *Blowfish*, natomiast klucze sesyjne każdego użytkownika szyfrowane są algorytmem *RSA*. Do powyższych algorytmów został dodany provider *Bouncy Castle.*

## Sczegóły

|  |  |
| --- | --- |
| Język | Java 8 |
| Algorytm | Blowfish |
| Długość klucza sesyjnego | 128 bit |
| Tryby pracy | ECB / CBC / CFB / OFB |
| Wielkości podbloków(CFB / OFB) | 8 / 16 / 24 / 32 / 48 bit |
| Generator wartości losowych | SecureRandom(Java 8) |

# Opis działania programu

## Dodawanie użytkowników

Dodawanie nonwego użytkownika przeprowadzane jest poprzez wybranie *Program->Dodaj użytkownika.* Program prosi o podanie adresu e-mail oraz hasła. Po podaniu tych parametrów tworzone są dwa pliki o nazie odpowiadającej podanemu adresowi e-mail, zawierająe klucz prywatny i plubliczny, znajdujące się odpowiednio w katalogach *privateKey* oraz *publicKey*. Klucz publiczny zapisany został w postaci zaszyfrowanej przy pomocy algorytmu *AES* w trzybie *ECB*(AES/ECB/PKCD5Padding), gdzie do zaszyfrowania jako klucza użyto skrótu *SHA-256* hasła podanego przez użytkownika.

## Szyfrowanie plików

Szyfrowanie plików przeprowadzane jest po wskazaniu przez użytkownika pliku źródłowego oraz wynikowego i wybraniu *Szyfruj.* Szyfrowanie pliku przeprowadzane jest w oddzielnym wątku.

Kiedy użytkownik rozpocznie szyfrowanie pliku, najpierw zostaje wygenerowany unikalny klucz sesyjny (*javax.crypto.SecredKey*) o długości 128 bit. Do generowania kluczy została użyta klasa *KeyGenerator(javax.crypto.KeyGenerator)*, a do zapewnienia wartości losowych została użyta klasa *SecureRandom(java.security.SecureRandom),* zainicjowana za pomocą metody z *Java 8 – SecureRandom.getInstanceStrong().*

Następnie zostaje stworzony *FileHeader*, w formacie *XML,* który zawiera informacje na temat metody szyfrowania pliku oraz listę użytkowników, uprawnionych do odczytania pliku. Dla każdego użytkownika, klucz sesyjny zostaje zaszyfrowany jego kluczem publicznym.

Plik jest zapisywany do wcześniej wskazanej lokalizacji za pomocą szyfrującego strumienia wyjścia(*javax.crypto.CipherOutputStream)*. Strumień jest tworzony przy użyciu instancji klasy *Cipher(javax.crypto.Cipher),* która zostaje zainicjowana dla algorytmu *Blowfish,* trybu wybranego przez użytkownika(*ECB / CBC / CFB / OFB*), i w przypadku trybów *CFB oraz OFB –* długością bloku(8 / 16 / 24 / 32 / 48 bit). Dla każdego trybu, za wyjątkiem trybu *ECB,* generowany jest wektor początkowy *IV*. Wektor generowany jest jako 8-elementowa tablica bajtów, przy pomocy funcji *nextBytes(),* pochodzącej z klasy *SecureRandom(SecureRandom.getInstanceStrong.nextBytes(byte[] array)*). Zapisywanie pliku przeprowadzane jest w częściach po 1024 bajty.

## Deszyfrowanie pliku

Aby odszyfrować plik, użytkownik musi wybrać jego lokalizację, oraz lokalizację pliku wynikowego. Po wybraniu zaszyfrowanego pliku odcytywany jest nagłówek, a z niego lista upoważnionych odbiorców. Użytkownik musi wybrać z listy siebie (zaznaczenie poprzez jednokrotne kliknięcie na adres e-mail), a następnie wpisać hasło i wybrać *Odszyfruj.* Deszyfracja pliku przebiega w oddzielnym wątku.

Po rozpoczęciu odszyfrowywania pliku, uprawnieni użytkownicy ładowani są do listy, z której następnie wybierany jest zaznaczony użytkownik.

Pobierany jest zaszyfrowany klucz prywatny, który zostaje odsyfrowany za pomocą skrótu hasła podanego przez użytkownika. Jeśli hasło podane przez użytkownika jest nieprawidłowe, zostaje wygenerowany losowy klucz prywatny.

Za pomocą posiadanego klucza prywatnego, odszyfrowany zostaje klucz sesyjny. Jeżeli klucz prywatny nie jest prawidłowy, to generowany jest losowy klucz sesyjny.

Za pomocą posiadanego klucza sesyjnego, odszyfrowany zostaje plik. Jeżeli klucz sesyjny nie był prawidłowy, to plik nie będzie możliwy do odczytania.

# Struktura pliku wynikowego

## Przykładowy plik wynikowy

