Adam Wudziński 212466

# Raport z laboratorium nr 1 (Protokół DH)

## 1. Cel

Celem laboratorium było stworzenie prostej aplikacji klient-serwer do przesyłania wiadomości. Komunikacja miała odbywać się z wykorzystaniem TCP, a do ustanowienia połączenia pomiędzy klientem a serwerem miał zostać wykorzystany protokół Diffiego-Hellmana. Aplikacja mogła powstać w dowolnym języku i technologii.

## 2. Technologie

Do wykonania zadania użyłem języka Java w wersji 7, jest to też wersja wymagana do uruchomienia projektu. Do zarządzania zależnościami i budowania projektu skorzystałem z Mavena 4.0.0. Wybrałem język Java ponieważ mam w nim najwięcej doświadczenia.

## 3. Sposób wykonania

Projekt ma dwie główne klasy: Client i Server. Obie mają po jednej głównej metodzie, która rozpoczyna ich działanie.

W przypadku serwera jest to metoda startListening(), która uruchamia server i nasłuchuje połączeń na określonym w konstruktorze porcie. Gdy dostanie połączenie zaczyna zgodnie z protokołem DH, wymianę informacji. Jeżeli w między czasie format wiadomości, którą dostaje od klienta będzie nie poprawny aplikacja wyrzuci błąd. Niestety nie jest on obsługiwany.

Jeżeli uda się nawiązać połączenie, serwer będzie wyświetlał wiadomości które dostaje od klienta.

Klasa Client do rozpoczęcia swojego działania ma metode connectToServer(), która łączy się z serwerem, którego adres i port określony jest w konstruktorze. Jest tam również określany sposób szyfrowania.

Serwer nie obsługuje wielu klientów, jedynie jednego, oraz nie można dynamicznie zmieniać sposobu szyfrowania.

Obsługa nawiązywania połączenia(obsługi protokołu) jest delegowana do klas ClientDhProtocol oraz ServeerDhProtocol. One natomiast korzystają z klas MathUtils do wyliczania sekretów.

Samo połączenie jak już zostanie nawiązane jest wykonywane poprzez klasę Connection, która posiada referencje do socketu, streamu wejściowego i wyjściowego oraz strategii kryptowania. Pozwala wysłać oraz odebrać wiadomość, która zostanie odpowiednio zaszyfrowana lub zdeszyfrowana, w zależności od algorytmu ustalonego podczas nawiązywania połączenia.

Wybór odpowiedniej strategii do zaszyfrowania przesyłanej wiadomości delegowany jest do klasy CryptionStrategy, która pamięta jaki sposób szyfrowania został wybrany i na tej podstawie wywołuje odpowiednie algorytmy szyfrujące, których implementacja znajduje się w klasie CryptorUtils. Implementacje algorytmów szyfrujących znalazłem na stronie stackoverflow.com

Do obsługi Jsona wykorzystałem klase JSONObject z pakietu com.google.json-simple jsonorg.json.simple od Google, ponieważ jest łatwa w obsłudze. Do przesyłu danych wykorzystałem klasy ObjectInputStream i ObjectOutputStream z pakietu java.io, ponieważ dzięki nim przesłanie obiektów jsona jest trywialnie proste.

W projekcie znajduje się również plik README.md, w którym opisałem jak zbudować i uruchomić projekt.

## 4. Wnioski i przemyślenia

Było to dla mnie ciekawe doświadczanie, ponieważ pierwszy raz implementowałem jakikolwiek protokół pomiędzy serwerem a klientem. Bardzo ciekawy jest sam algorytm, że tak naprawdę wykorzystując dość proste narzędzia i zależności pozwala zapewnić bardzo satysfakcjonujący poziom bezpieczeństwa.