Politechnika Wrocławska
Katedra Informatyki Technicznej
Zakład Architektury Komputerów

Bezpieczeństwo usług i systemów informatycznych 2
Sprawozdanie z laboratorium nr 1

Komunikator z szyfrowaniem

Autor:

Radosław Lis 241385

prowadzący - mgr inż. Przemysław Świercz

Wrocław, dn. 26 października 2020

Spis treści

1 Cel zadania

Celem zadania było stworzenia komunikatora w architekturze klient-serwer i zaimplementowanie protokołu Diffiego-Hellmana oraz prostym kodowaniu wiadomości w oparciu o wyliczony sekret (k).

2 Sposób wykonania zadania

2.1 Języki i technologie

Zarówno aplikacja kliencka jak i serwera została wykonana w języku Java, a GUI zostało utworzone za pomocą technologii JavaFX.

2.2 Zrealizowanie połączenia TCP

W celu reprezentacji połączenia między klientem a serwerem zostały użyte klasy z pakietu *java.net*, tj. klasa *ServerSocket* oraz klasa *Socket*, odpowiednio dla aplikacji serwera oraz klienta.

2.3 Logika komunikatora

Protokół został zaimplementowany zgodnie z informacjami zawartymi tutaj.

Po uruchomieniu aplikacji serwera i naciśnięciu przycisku **Run server** tworzy się obiekt klasy *ServerSocket* działający na porcie 6666. Czeka nas przychodzące klienckie połączenia i tworzy nowe wątki odpowiadające za sesje (obiekty klasy *Session*), co umożliwia wsparcie wielu klientów jednocześnie (zgodnie z sekcją *Supporting multiple clients* tutaj). Po utworzeniu takiego obiektu tworzy się obiekt klasy *Generator*, który ustala wartości parametrów p oraz g (inne dla każdej sesji).

Następnie po uruchomieniu aplikacji klienckich i naciśnięciu przycisku **Start connection** inicjalizowane jest połączenie do serwera i tworzony jest nowy obiekt klasy *Socket* oraz klient rozpoczyna nasłuchiwanie na odbiór parametru B od serwera. Gdy zostanie zainicjowane połączenie i zostanie utworzony nowe obiekt klasy *Session*, próbuje on od razu wysłać do klienta obliczoną wartość parametru B, która pojawia się w graficznym UI klienta w odpowiednim polu.

Następny krok należy do klienta, który wysyła prośbę o parametry p i g do serwera. Po ich otrzymaniu musi wylosować parametr a i obliczyć za jego pomocą A. Następnie może albo wysłać to A serwerowi albo obliczyć sekret (bo ma już i p i g oraz B). Bez różnicy co zrobi najpierw. Po wyliczeniu sekretu tworzy się nowy obiekt klasy *Conversation*, który obsługuję rozmowę z serwerem i odkodowuje wiadomości. Już teraz może zacząć wysyłać wiadomości do serwera, ale bez szyfrowania, które jest domyślnie ustawione na **NONE**. W dowolnym momencie konwersacji może zmienić rodzaj szyfrowania albo na XOR albo na szyfr Cezara, w których użyty jest najmłodszy bajt sekretu.

3 Szczegóły implementacyjne

W celu zapewnienia bezpiecznej wymiany informacji przy tworzeniu parametrów i ich wyliczaniu została wykorzystana klasa *BigInteger* dostarczona przez Javę. Typ *long* niestety nie spełniał wymagań, gdyż pozwala przechowywać w pamięci tylko 64-bitową liczbę. Dodatkowo pomocną okazała się metoda $powMod$, która pozwala na wykonanie potrzebnych obliczeń w zaledwie jednej linii kodu. U mnie wielkości parametrów zostały ustalone następująco:

- p - 1024 bity,
- g - 256 bitów,
- a - 512 bitów,
- b - 512 bitów

Parametr p został wyliczony za pomocą statycznej metody klasy *BigInteger* nazywającej się *probablePrime* (informacja tu), która przyjmuje liczbę bitów ile ma mieć liczba pierwsza oraz obiekt klasy *Random* (w moim przypadku trochę bezpieczniejsze *SecureRandom*) i zwraca liczbę pierwszą.

Kodowanie *base*64 odbywa się za pomocą klasy *Base*64 z pakietu *java.util*, która udostępnia potrzebne nam metody *decode* oraz *encode*. Użycie tej gotowej klasy było po prostu wygodne i zaoszczędziło dużo zasobów w porównaniu do sytuacji gdybym miał sam dokonać implementacji.

4 Wnioski

Gotowe klasy Javy (np. *BigInteger*, *Socket*) znacząco przyspieszają implementację protokołu Diffiego-Hellmana. Ważny jest odpowiedni dobór wielkości parametrów, tak żeby ryzyko przechwytu wiadomości było jak najmniejsze. Program jest łatwo rozszerzalny, można dodać inne metody szyfrowania, np. poprzez klucze publiczne.