Kamil Petk nr 176648

**Opis modelu ataku man-in-the-middle na anonimowy protokół Diffiego-Hellmana**

Atak man-in-the-middle polega na włączeniu się napastnika pomiędzy dwie komunikujące się ze sobą strony. Zajmuje on potajemne miejsce między nimi i zaczyna pośredniczyć w wymianie wiadomości podszywając się pod obie strony tak, aby pozostać niezauważonym.

Protokół Diifiego-Hellmana pozwala uzgadniać klucze szyfrujące między stronami. Taki klucz może zostać wykorzystany np. do szyfrowania komunikacji. Jest on jednak podatny na ataki man-in-the-middle.

Opis kolejnych kroków ataku (na przykładzie Alicja, Bob, Ewa):

1. Alicja wraz z Bobem uzgadnia dużą liczbę pierwszą p oraz liczbę g.
2. Alicja wybiera losowy wykładnik a < p – 1
3. Następnie Alicja oblicza A = (g^a) mod p i wysyła je do Boba.
4. Ewa będąca ukrytym napastnikiem w komunikacji między Alicją a Bobem odrzuca A.
5. Następnie Ewa wybiera losowy wykładnik c < p – 1
6. Ewa oblicza C = (g^c) mod p i wysyła je do Boba.
7. Bob odbiera C myśląc, że otrzymał ją od Alicji.
8. Bob wybiera losowy wykładnik b < p - 1
9. Oblicza Key = (C^b) mod p
10. Bob ustawia B = (g^b) mod p i wysyła do Alicji.
11. Ewa odrzuca B.
12. Następnie Ewa oblicza Key = (B^c) mod p.
13. Ewa wybiera losowy wykładnik d.
14. Oblicza drugi Key = (A^d) mod p
15. Ewa ustawia D = (g^d) mod p i wysyła do Alicji
16. Alicja odbiera D myśląc, że otrzymała je od Boba.
17. Następnie Alicja oblicza Key = (D^a) mod p

Efektem tych kroków jest posiadanie przez Ewę obu kluczy do komunikacji z Alicją i Bobem.

Key = (C^b) mod p = (B^c) mod p

Key = (D^a) mod p = (A^d) mod p  
Jest ona teraz w stanie podsłuchiwać i odszyfrowywać przesyłane między nimi wiadomości pozostając nadal w ukryciu, gdyż po każdym odszyfrowaniu jest w stanie zaszyfrować ponownie wiadomość tak, aby jej odbiorca mógł także ją odszyfrować za pomocą swojego klucza.

Utworzona przeze mnie symulacja w języku C++ oraz Python obrazuje atak. Niestety dla języka C++ użycie dużych liczb p, a, b, c, d powodowało problemy z obliczeniami uzyskując bardzo duże liczby (nawet przy użyciu long long int) skutkując błędnymi wynikami dla operacji mod. W języku Python problem ten nie występuje, co pozwoliło zastosować dużą liczbę pierwszą p oraz losowość dla a, b, c, d.