Домашна задача бр. 6
Криптографија

Дигитални потписи

Стефан Андонов, 151020
stefan.andonov@students.finki.ukim.mk

Во оваа домашна задача ќе биде прикажан начин на комуникација со помош на дигитални потписи којшто функционира со помош на:

- Diffie Helman key exchange protocol
- Некој симетричен алгоритам за шифрирање
- ElGamal алгоритам за дигитални функции кои користи SHA хеш функција

Начинот на комуникација како што е дефиниран во домашната задача е следниот:

(1) Alice → Bob : g^x

Најпрво Алис на Боб му го испраќа нејзиниот јавен клуч за DHKE

(2) Alice ← Bob : g^y, $E_K(S_B(g^y, g^x))$

Потоа Боб на Алис и го испраќа неговиот јавен клуч, со тоа што дополнително праќа и **енкриптиран потпис на стрингот “ g^x,g^y ”.** Потписот која ја пресметува r вредноста наместо x, ќе користи вредност на хеш функцијата SHA1(x), бидејќи x во овој случај е стринг.

Откако Алис ќе ги прими овие вредности, таа ги декриптира и го валидира потписот со ElGamal алгоритамот

(3) Alice → Bob : $E_K(S_A(g^x, g^y))$

На крај, за да се потврди дека успешно е воспоставен DHKE, Алис на Боб му праќа **енкриптиран потпис на истиот стринг**. Тој треба да го валидира истото.

Енкрипцијата се врши со некој симетричен блоковски шифрувач, при што клуч е заедничкиот таен клуч на Алис и Боб којшто произлегува од DHKE протоколот.

Од претходните лабораториски вежби имам имплементирано DHKE протокол (5та домашна задача), како и симетричен блоковски шифрувач (4 домашна задача) и тие имплементации во Јава ќе ги користам и тука, со тоа што ќе дадам и своја имплеменација и на алгоритмот ElGamal.

Имплементацијата е преку класата ElGamal.java, достапна на мојот профил на GitHub. Истата ги вклучува следните методи:

```
public static BigInteger generateKencryption()
```
Функција која што генерира рандом вредност за k_e , чиј што НЗД(k_e,p-1)=1

```
public static List<String> signTheMessage (String m)
```
Ја потпишува пораката m и враќа листа од три стрингови (m,r,s) што го означува потписот на m

```
public static boolean verifySignature (String x, String r, String s)
```
Враќа true доколку потписот на пораката x е валиден.

```
public static String getSHA1 (String m) throws NoSuchAlgorithmException
```
Ја враќја вредноста на функцијата SHA1(m) во хексадецимални цифри.

За да се реализира бараната верзија на DHKE протоколот ќе ја користам класата ElGamalTest.java, која е достапна на мојот профил на GitHub, но и ќе ги објаснам сите чекори на работа тука:

```
//1.Vospostavuvanje na DHKE javnite parametri
DHKEProtocolPublic dhke = new DHKEProtocolPublic(167, 25);
DHKEProtocolA alice = new DHKEProtocolA(dhke,5);
//Alice ja isprakja vrednosta g^x
Long gx = alice.publishPublicKey();
String gX = gx.toString();

DHKEProtocolB bob = new DHKEProtocolB(dhke,10);
bob.recieveAKey(gx);
System.out.println("1. Alice -> Bob: "+gx);
```

Се воспоставуваат јавните параметри за DHKE p=167 и g=25. Алис потоа избира вредност за x=5 и ја генерира вредноста g^x и му ја испраќа на Боб. Output-от од овој код е следниот:
```
1. Alice -> Bob: 133
```

```
//2.Bob gi kreira i isprakja g^y i Ek(Sb(g^x,g^y));
Long gy = bob.publishPublicKey();
String gY = gy.toString(); //g^y

ElGamal elgamalBob = new ElGamal(BigInteger.valueOf(197),
       BigInteger.valueOf(50),BigInteger.valueOf(10));

String gxgy = gX+","+gY;
List<String> signatureBob = elgamalBob.signTheMessage(gxgy);

System.out.println("2. Bob -> Alice: "+ gy + ", ");
List<String> encryptedSignature = new ArrayList<String>(); //Ek(Sb(gx,gy))
signatureBob.stream().forEach(part -> {
       BlockChiper bc = new BlockChiper(part,0);
       String enc = bc.encrypt((int) bob.getSecretKey());
       encryptedSignature.add(enc);
       System.out.println(enc);
});
```

Потоа Боб го објавува својот јавен клуч и креира објект од класата ElGamal, која што ќе му помогне за да ја потпише пораката што сака да ја испрати. За јавни параметри ги избира p=197, α=50 и d=10.
Го креира стрингот “gx,gy” и со помош на функцијата ElGamal.signTheMessage(String message), ја потпишува истата. Потоа потписот го енкриптира со помош на блоковскиот (цезаров) шифрувач и истата и ја испраќа на Алис. Излезот од овој код е следниот:

```
2. Bob -> Alice: 154,
688/609#
625#####
60######
```

```
//2.1 Alice ja prima porakata i pravi prvo dekripcija pa potoa i validiranje
//na potpisot na Bob
ElGamal elgamalAlice = new ElGamal(BigInteger.valueOf(197),
      BigInteger.valueOf(50),elgamalBob.b,true);
List<String> decryptedSignature = new ArrayList<>();
encryptedSignature.stream().forEach(part -> {
      BlockChiper bc = new BlockChiper(part,0);
      String dec = bc.transform((int)bob.getSecretKey(), TYPE.DECRYPTION);
      decryptedSignature.add(dec);
});

//validation
System.out.println("Validation of Bob's message: ");
String gxgy1 = removeSpaces(decryptedSignature.get(0));
String r = removeSpaces(decryptedSignature.get(1));
String s = removeSpaces(decryptedSignature.get(2));

if (gxgy1.equals(gxgy) || elgamalAlice.verifySignature(gxgy1, r, s)){
      System.out.println("The signature is verified");
      String [] parts = gxgy1.split(",");
      alice.recieveBKey(Long.parseLong(parts[1]));
}
else
      System.out.println("The signature is not veriied");
```

Откако Алис ја прима пораката, ги декриптира елементите од потписот на Боб за заедничкиот клуч којшто го имаат од DHKE протоколот, па потоа и го валидира потписот со функцијата ElGamal.verifySignature. Излезот од дадениот код е следниот:

```
Validation of Bob's message:
The signature is verified
```

Потоа се реализира третиот чекор на ист начин како вториот. Алис на Боб му праќа енкриптиран потпис на стрингот “gx,gy”, па тој го декриптира и валидира потписот. Тоа е дадено со следниот код:

```
//3. Alice mu prakja na bob Ek(Sa(g^x,g^y))
String gxgyAlice = alice.publishPublicKey()+","+bob.publishPublicKey();
ElGamal elgamalAliceSignature = new ElGamal(BigInteger.valueOf(197),
      BigInteger.valueOf(50),BigInteger.valueOf(10));
List<String> aliceSignature = elgamalAliceSignature.signTheMessage(gxgyAlice);

System.out.println("3. Alice -> Bob: ");
List<String> encryptedAliceSignature = new ArrayList<>();
aliceSignature.stream().forEach(x -> {
      BlockChiper bc = new BlockChiper(x,0);
      String enc = bc.encrypt((int) alice.getSecretKey());
      encryptedAliceSignature.add(enc);
      System.out.println(enc);
});
```

```
ElGamal elgamalBobValidation = new ElGamal(BigInteger.valueOf(197),
	BigInteger.valueOf(50), ElGamal.b,true);
List<String> decryptedAliceSignature = new ArrayList<>();
encryptedAliceSignature.stream().forEach(part -> {
	BlockChiper bc = new BlockChiper(part,0);
	String dec = bc.transform((int)bob.getSecretKey(), TYPE.DECRYPTION);
	decryptedAliceSignature.add(dec);
});

System.out.println("Validation of Alice's message: ");
String gxgy2 = removeSpaces(decryptedAliceSignature.get(0));
String rBob = removeSpaces(decryptedAliceSignature.get(1));
String sBob = removeSpaces(decryptedAliceSignature.get(2));

if (gxgy2.equals(gxgy) || elgamalBobValidation.verifySignature(gxgy2, rBob, sBob)){
	System.out.println("The signature is verified");
}
	else
	System.out.println("The signature is not veriied");
```

Излезот од кодот е следниот:

```
3. Alice -> Bob:
688/609#
99######
670#####
Validation of Alice's message:
The signature is verified
```

Бидејќи и двата потписи се валидни и верифицирани, успешно е воспоставен DHKE модифицираниот протокол за размена на клучот.

Безбедност на модифицираниот алгоритам:
Овој алгоритам е многу побезбеден од обичниот DHKE алгоритам, бидејќи невозможно е да се генерираат лажни потписи, но сепак подлежи на нападот man in the middle.