Bit Compromise Report

Bit Compromise Operation

Bit Compromise scheme works as follows:

a) *B* generates a random value *r* and sends it to *A*.
b) *A* uses a cipher function *E( )* with a key k to cipher the value *b* (compromised value) and the random value *r*. Then sends to *B* the value of $C(b) = E_k(b, r)$.

To open the compromise

a) *A* sends its key *k* to *B*.
b) B decipher the compromise *C (b)* using the received key and check that *b* is with its random value *r*.

Implementation

There are two classes which correspond to A and B, they are called Anna and Bernat, they connect each other through a socket connection.

Inside their classes the protocol run as follows for Anna class:

1. Anna waits for the connection of B, the Bit which she has chosen is shown in the text box labeled “Bit”.
2. Anna waits for the message “Random” from Bernat, once he is connected. At the time, she generates a key and then cipher the text though AES encryption.

```
byte bytes[] = (byte[])in.readObject();
key = generateKey();
cipheredText = aes.encrypt(bytes, key);
```

   According to the Cipher Java Documentation, AES algorithms in CBC mode expect a 16-byte parameter value for the initial vector(IV) in bArray.
3. Then she sends ciphered text to Bernat.
4. Once Bernat has tossed the coin and she receives it, she is able to send the key to Bernat.

The protocol run as follows for Bernat:

1. Anna is already waiting and he connects, he generates a random number, and then tells Anna he is going to send it.

```
SecureRandom rand;
rand = SecureRandom.getInstance("SHA1PRNG", "SUN");
randomBytes = new byte[15];
rand.nextBytes(randomBytes);
sendMessage("Random");
sendMessage(randomBytes);
```

2. Then he waits for the box preceded with a label “Box” with his ciphered random number and the compromised bit, when he receives it, he is now able to toss the coin and send it.

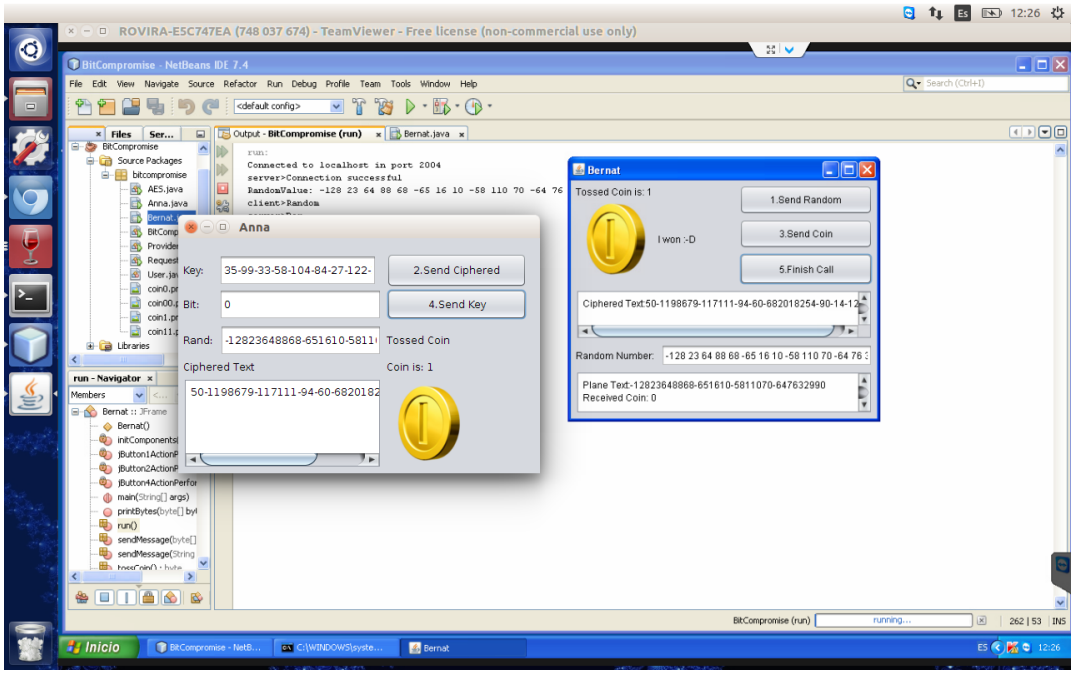
```
coin = tossCoin();
sendMessage("Coin is: "+Byte.toString(coin));
```

3. Then he waits for the key, to open the box he received from Anna and check who won.

To show how the protocol works, it has been put a number in the buttons to follow the sequence properly. In the image below is shown how they communicate remotely. I one side Anna is running in a Linux system, and in the other side Bernat is running on a remote Windows system, it has been done to make visible the different sides in which they are running.

The use of the interface is as follows:

1. Run Anna, bit 0 is already chosen but it may be changed in the box.
2. Run Bernat, if it is necessary it can be observed the console in which are informed the events of the programs, for example that the connection has been made successfully.
3. Click on Button “1. Send Random”.
4. Anna receives the random number, click on Button “2. Send Ciphered”.
5. Bernat receives the Ciphered Text so he tosses the coin by clicking in Button “3.Send Coin”.
6. Anna receives the coin and she already knows who won, but Bernat does not. So she sends the key by clicking “4. Send Key”.
7. Bernat receives the Key and at the moment he knows who won, so he finish the call by clicking on “5. Finish the call” and with this announcing who won.



Notes:

- It has been used AES encryption to cipher the random key plus the compromised bit.
- The key length was 16 bits.
- They can run in the same computer as the sockets are set to localhost.