Group Golf

4/29/19

Project 3 Report

Introduction

For this project we had to implement the Oblivious Transfer Protocol. We were given Alice's side of the code, and we had to implement Bob's based off what we were given for Alice. These implementations were just a rough draft where there was no consideration given for cheating. Once we had the basic implementations working, we then had to fix security errors in the code.

Methodology

First, we had to get the basic OTP working. Given Alice's code this was straight forward to do. Once we had that working, we were given questions we had to answer that would give use ideas of what we needed to defend against.

1. A : generates asymmetric keypairs $(K_I^{priv}, K_I^{pub}), (K_J^{priv}, K_J^{pub})$, message M
2. B : generates symmetric key K_B
3. $A \rightarrow B$: K_I^{pub}, K_J^{pub}
4. B : selects H from I, J at random
   $B \rightarrow A$: $\{K_B\}_{K_H^{pub}}$
5. A : selects G from I, J at random
   computes $K_A = \left\{\{K_B\}_{K_H^{pub}}\right\}_{K_G^{priv}}$
   Note: $G = H \implies K_A = K_B$
6. $A \rightarrow B$: $\{M\}_{K_A} \bullet G$
7. B : computes $M' = \left\{\{M\}_{K_A}\right\}_{K_B}$
   Note: $G = H \implies K_A = K_B \implies M = M' \implies B$ wins
   Note: $G \neq H \implies A$ wins
   $B \rightarrow A$: $M \bullet H$
8. $A \rightarrow B$: K_I^{priv}, K_J^{priv} for verification

Above is the protocol we implemented. The next section will cover the questions and forms of cheating we covered.

Protocol and Cheat Overview

6.1 Alice.Java

1. What mistake has Alice made in step 6, and how might Bob take advantage? How must Alice fix it?

Alice isn't sending a different message every time they do the protocol. If Bob were to realize this, he could return that decrypted message to Alice and say that he won. The way that Alice fixes this is by returning different messages every time. We have implemented this using a random number generator and seven different possible messages.

2. At what point does Alice learn whether she wins or loses?

Alice learns the outcome at the end of step seven, once Bob has sent her what he has as the decrypted message and what his choice of key was (H).

3. What might Alice do in step 8 to confuse Bob?

In step 8 Alice sends Bob both private keys corresponding to the public keys that he got in step 3. Alice could send Bob the same private key each time. She could also send Bob private keys that didn't correspond to the public keys.

To beat this, we have Bob checking that the public and private keys are unique and that the corresponding keys encrypt and decrypt.

6.2 How might Alice decryptKey() to take advantage of a likely mistake by Bob? Notice that Alice provided Bob an encryptKey() method. How should Bob change this to thwart Alice?

Alice could use decrypt key to find the key if Bob didn't initialize the array holding the bytes to be full of random bytes.

Bob should change this method to create an initial array of random bytes that way Alice can't know where they key is in the array.

6.3 csec2019 / CommutativeRSAKeyPairGenerator.Java

1. What does it mean for RSA keys to be commutative?

For keys to be commutative it means that they share a common modulus.

2. Why do you suppose Alice chose to use commutative RSA keys?

Using commutative keys is the main way for the OTP to be fair.

3. What advantage might she gain if she uses non-commutative RSA keys?

If she were to use non-commutative keys, then she would end up being the one who knows whether she wins or loses before Bob. She would be able to adjust her call to where she would always win.

6.4 What about Bob?

1. At what point does Bob learn whether he wins or loses?

Bob learns whether he wins or loses in step 7.

2. What checks should Bob perform after receiving Alice's private keys in step 8?

We have Bob checking that the public and private keys are unique and that the corresponding keys encrypt and decrypt.

Sample Inputs and Outputs

There are no inputs nor outputs it is all randomly generated at time of compilation.