Université Libre de Bruxelles
INFO-F405 - Computer Security
Synchronized File Exchanger

Groupe 8:
Csori Pinto Constanza, Dauphin Guillaume, Deng Rhenzi, Picard Simon

December 1, 2014

Contents

1 Introduction **3**

2 Implementation **3**

3 Code description **3**

4 Threat model **3**
4.1 Information document 3
4.2 External dependencies 4
4.3 Implementation assumptions 4
4.4 External security notes 4
4.5 Internal security notes 4
4.6 Levels of trust 5
4.7 Entry points 5
4.8 Assets 6

5 Model analysis **6**
5.1 Data flow diagram 6
5.2 Potential threats 8

6 Conclusion **8**

1 Introduction

The prupose of the project is to implement a secure synchronized file exchanger between two computers with the possibility to send or receive a file. In this project we implemented a custom protocol and we used AES and RSA algorithm to encrypt and decrypt data in order to securely send informations. We also use SHA3 hash generate signature and thus verify the identity of the correspondant.

2 Implementation

The project is coded in Java.
The generation of the key was not to be handle by the project but using openssl.
Here are the commands that lead to the required files by the project :

```
openssl req -x509 -newkey rsa:2048 -keyout private_key.pem -out public_certificate.pem -days 365
openssl pkcs8 -topk8 -nocrypt -in private_key.pem -inform PEM -out private_key.der -outform DER
openssl x509 -in public_certificate.pem -inform PEM {out public_certificate.der -outform DER
```

First we generate the private key and the public certificate and the we convert them to DER files that Java can read.

AES and RSA implementation were handle by the Java native library javax.crypto.
SHA3 implementation were found on the internet.
RSA signature with SHA3 was not implemented so it is done as follow : generate SHA3 hash and then encrypt it with private key.
The verification wass to hash the same data, decrypt the signature usign the public key and compare the hashes.

The GUI was done using SWING.

3 Code description

There is two possible action send or receive a file. The receiver is the server and the sender is the client, once the socket are opened the protocol can begin. First the sender, called Alice, send his public certificate and the receiver, called Bob, do the same, after that the Alice generate a nonce by concatenating a 32 bits random number and a timestamp. Then Alice send a session key wich is a random 128 bits number prefixed by his IP and Bob's IP. This message is encrypted in RSA using Bob's public key. Once the session key is sent, Alice send the next message which is the file wrapped by the IPs and the nonce is encrypted in AES using the session key. Then Alice send her signature and Bob verify that it is correct usign Alice's public key. Finally Bob sent a hash to Alice as an acknowledge.

4 Threat model

4.1 Information document

Step of the development : Version 1.0
Director : Csori Pinto Constanza, Dauphin Guillaume, Deng Rhenzi, Picard Simon
Participants : Csori Pinto Constanza, Dauphin Guillaume, Deng Rhenzi, Picard Simon
Reviewer : Csori Pinto Constanza, Dauphin Guillaume, Deng Rhenzi, Picard Simon
Localisation : ULB
Description : The group of students from ULB has created an application for secure file exchange between two users. The file is encrypted and the system uses a encrypted secure protocol.

4.2 External dependencies

ID : 1
Description : The software runs on two computers. Because the code is in Java, it can be easily ported on smartphones or tablet.

ID : 2
Description : Both computers are connected to each other. The system works with cable connection or wifi.

ID : 3
Description : There is a firewall on each computer.

ID : 4
Description : The OS on the computers can be any OS.

4.3 Implementation assumptions

ID : 1
Description : The random numbers are considered as random even if they're not perfectly random.

ID : 2
Description : The implementations of hashing and encryption algorithms must be secure and they must never leak sensitive data through buffer overflows, data in shared memory, etc.

ID : 3
Description : Java handle de sockets gestion therefor their implementation must be secure

4.4 External security notes

ID : 1
Description : Firewall must be set up with open port for the socket in java.

ID : 2
Description : The sender and the receiver must provide a private key to the program, they should not share this key.

4.5 Internal security notes

ID : 1
Description : OpenSSL is used to generate RSA key pairs. Thus we rely on OpenSSL not leaking sensitive data (e.g. it must not make keys accessible through some system hack), and OpenSSL must generate correct and secure RSA certificates.

ID : 2
Description : The server and client programs are implemented in Java, using libraries that provide AES and RSA implementations

4.6 Levels of trust

ID : 1
Name : Application
Description : The application itself communicate to the same application on the second computer.

ID : 2
Name : Internet
Description : Online application and connection to the computer. No permission on the application.

ID : 3
Name : Local Network
Description : User or application on the local Network. No permission on the application.

ID : 4
Name : Bluetooth User
Description : User or application using bluetooth near the computer of the sender or the receiver. No permission on the application.

ID : 5
Name : User
Description : The User of the application (sender or receiver).

4.7 Entry points

ID : 1
Name : Ethernet port
Description : Both computers are connected via a ethernet cable to each other.
Level of trust : (1) Application

ID : 2
Name : Wifi
Description : Both computers are connected via wifi to each other.
Level of trust : (1) Application, (2) Internet, (3) Local Network

ID : 3
Name : Bluetooth
Description : Both computers are connected via bluetooth to each other.
Level of trust : (1) Application, (4) Bluetooth User

ID : 4
Name : File
Description : The application is used to exchange files. At a given moment, the user must give a file to the application.
Level of trust : (5) User

ID : 5
Name : Keys
Description : The keys are used for the encryption. A code injection can come through that + buffer overflow.
Level of trust : (1) Application

4.8 Assets

ID : 1
Name : IP Address
Description : The IP adress is used as identifier for the users in the program. It will be encrypted during the process.
Level of trust : (1) Application, (5) User

ID : 2
Name : Session key
Description : The session key is used to encrypt the file that Alice wants to send to Bob.
Level of trust : (1) Application, (5) User

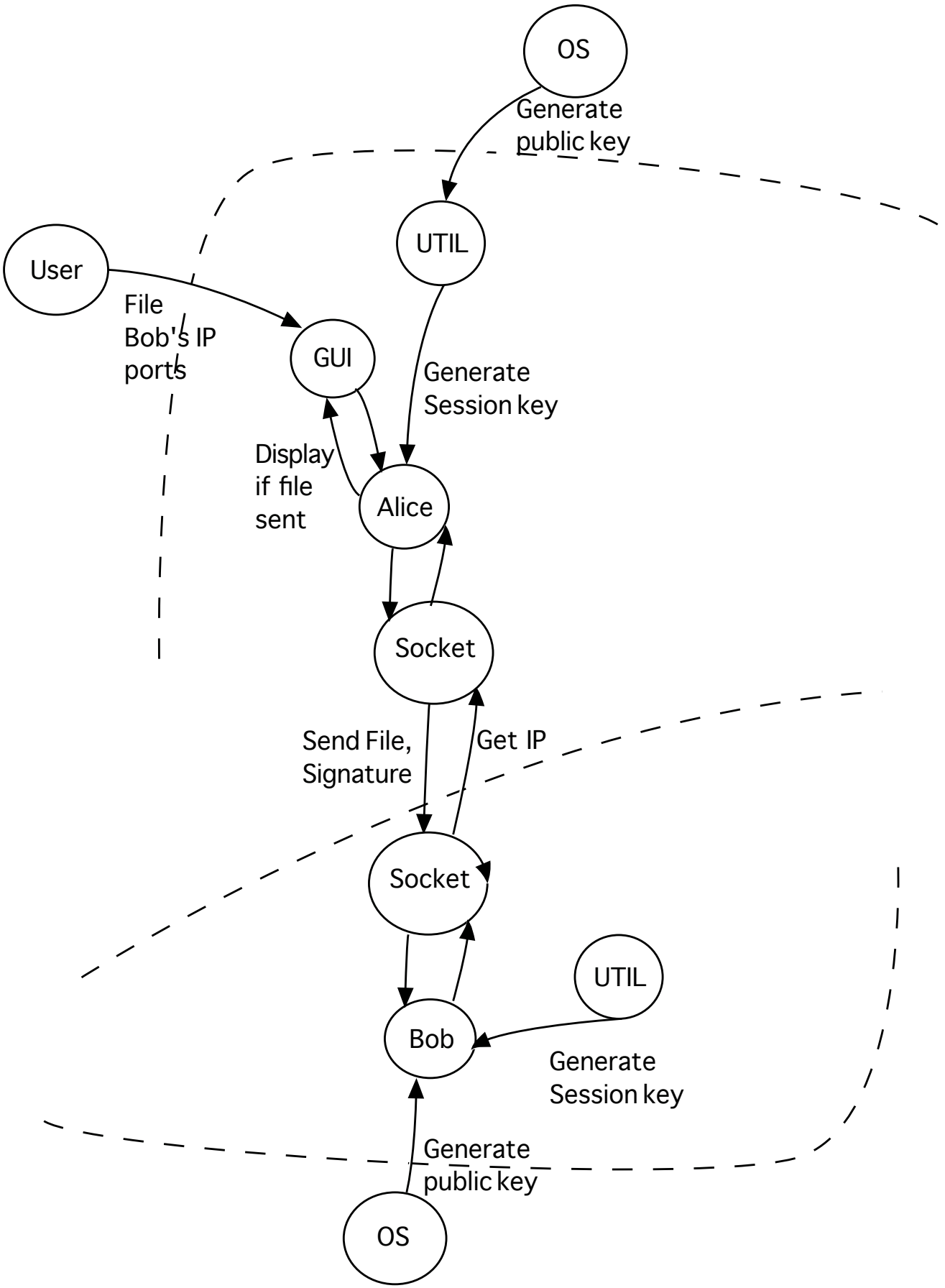
ID : 3
Name : Private key
Description : Used to create the signature.
Level of trust : (1) Application, (5) User

ID : 4
Name : Signature
Description : The signature is used to confirm the personnality of the sender.
Level of trust : (1) Application, (5) User

ID : 5
Name : File
Description : The file sent by the user.
Level of trust : (1) Application, (5) User

5 Model analysis

5.1 Data flow diagram


OS
Generate
public key
UTIL
User
File
Bob's IP
ports
GUI
Generate
Session key
Display
if file
sent
Alice
Socket
Send File,
Signature
Get IP
Socket
UTIL
Bob
Generate
Session key
Generate
public key
OS

5.2 Potential threats

6 Conclusion

The goal of the project was to send a file over the network in a secure way, using AES and RSA encryption and SHA3 hash, following a given protocol. First the sender distribute an AES key that is send encrypted by RSA and then the file is encrypted using AES. The design of the exchange prevent anyone but the receiver to decrypt the file if someone is able to intercept the messages.
Symmetric encryption and fecryption is often mor efficent and fits better for larger volume of date than assymetrics ciphers but a shared key is mandotory, so first a random session key is sent on the network encrypted using assymetric encryptoin, RSA, and then, for the large message, the file, it is encrypted using symmetric encryption, AES.