UNIVERSITÀ DI PISA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Project of Security in Networked Computing System

P2P Secure Messenger

by Giovanni Pollina

Accademic year 2015/2016

Index

1 Target

Project 3

Let’s consider a Peer-to-Peer system of Android devices where each peer owns a public and a private key. Before starting the communication, two peers (A and B) establish a session key to encrypt their communications. Design and analyze a key exchange protocol which satisfies the following requirements:

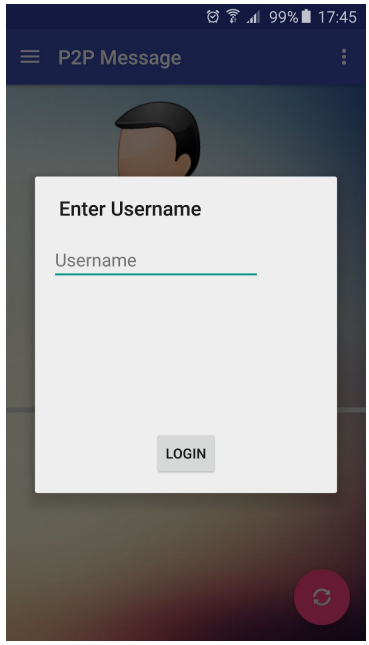
- At the end of the protocol a session key is established.
- At the end of the protocol A knows that B has received the session key.
- At the end of the protocol B knows that A has received the session key.

Develop a simple application which implements the designed protocol. After the session key is established A and B exchange a couple of messages encrypted and decrypted through the session key. The text of the messages, both sent and received, should be visible in an Activity.

2 User Interface

2.1 Main Activity

At the beginning the user have to insert his user name,than it will be used by NSD api in order to register it on the network:



In the MainActivity the user can see his IP address and the name selected previously, duringthe initial login and can see also the list of the available peers:

2.2 Chat Activity

When the user select the peer which want to connect, and the Key exchange protocol succeed, then a new Activity appears, the ChatActivity, it is used to send and receive messages.

3 Network Service Discovery (NSD)

In order to discover the available peer on the network is used the NSd framework provided by Android API.

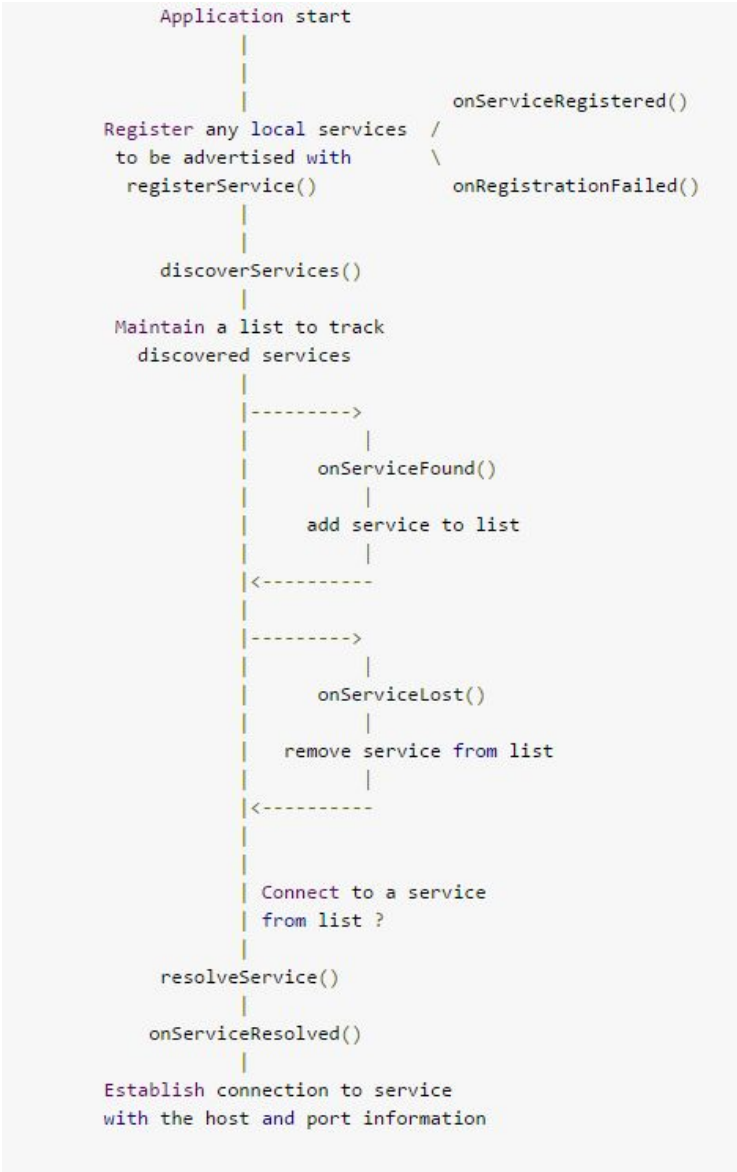
Network Service Discovery (NSD) allows to identify other devices on the local network that support the services my app requests.

(WiFi printers support this protocol for example.)

Nsd work with Multicast DNS protocol, for this reason the service is limited to Local Network.

The Nsd works with 4 fases:

1. -Register a Nsd service on the Local Network
2. -Discover other Nsd service on the Local Network
3. -Connect to the service available on the Local Network
4. -Eventually Unregister at the end my service from the Local Network



However there are three main operations the API supports - registration, discovery and resolution:

In order to use the Nsd framework on the app, what is used is NsdHelper class provide to all function that i need.

At the beginning the app control if the WiFi is enabled on the smartphone, it is possible by using the WifiManager class which provides the necessary function to handle WiFi interface on smartphone and other Android devices.

For instance in the app is used this simple piece of code to see if the WiFi is enabled or not, and in this case i will enable it:

```
WifiManager wifiManager = (WifiManager) getSystemService(Context.WIFI_SERVICE);
    if (!wifiManager.isWifiEnabled()) {


        /*Check is wifi is enabled*/

        Toast.makeText(MainActivity.this, "Activating WiFi..", Toast.LENGTH_LONG).show();


        //I activate the wifi

        wifiManager.setWifiEnabled(true);

    } else {

        /*WiFi is already active*/

        Toast.makeText(MainActivity.this, "WiFi active..", Toast.LENGTH_LONG).show();


    }
```

And with the class `WifiInfo` i will be able to obtain information about the WiFi interface, such as Ip address assigned by the router and other information.

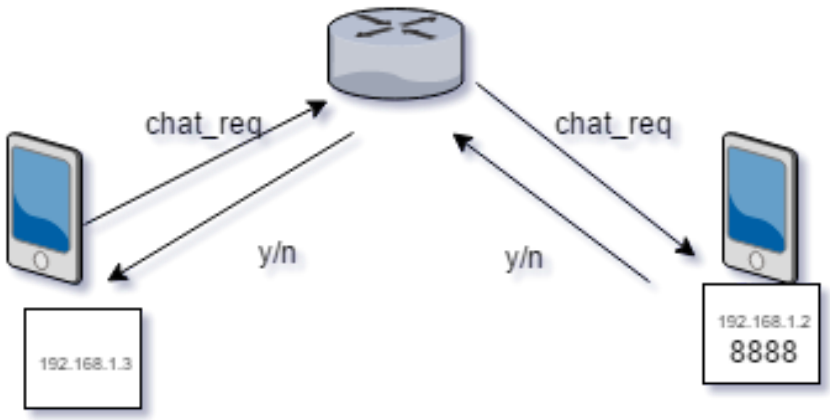
4 Network Operation

Sockets are used in order to perform network operations and permit exchange information between devices.

In particular are used 2 socket for send a chat_request and receive it.

This 2 sockets are handled by 2 Async Task:

- ServerAsyncTask, wich are issued at the start by the MainActiviti in onCreate() function and handle a ServerSocket wich listen always on port 8888 an incoming chat request.
- ClientAsyncTask, is activated every time i click on an available peer, it is launched by the listener `onItemClick(..)` used on the ListView which show in the MainActivity the list of the available peer.



5 Handle the peers

In order to handle the peer it was created a specific class called peer, this simple class record the information about an available peer, below the code:

```
import java.net.Socket;
import java.security.Key;


/**
* Created by gianni on 22/07/16.
*/
public class Peer {

   String address;
   String Name;
   int id;
   Socket sk; //if active then indicates the socket used to communicate with this specific peer
   Key publicKey;
   Key sessionKey;

   public Peer (String address,String Name){
      this.address=address;
      this.Name=Name;
   }
   public void setAddres(String address){
      this.address=address;
   }
   public void setName(String name){
      this.Name=name;
   }
   public void setId(int id){
      this.id=id;
   }
.....
```

and each of this peer is contained in a hashTable which maintain the list of peers:

```
import java.util.Hashtable;

/**
* Created by gianni on 24/07/16.
*/

public class ListOfPeer {

   private static Hashtable<String,Peer> peerList;
   int i;

   public ListOfPeer(){
      peerList=new Hashtable<String, Peer>(); //Allocate a new hashTable
   }

   public boolean lookup(String addr){
     return peerList.containsKey(addr);//Tests if the specified object is a key in this hashtable.
   }

 public static void insert(String addr,Peer p){

      peerList.put(addr,p);
   }

   public static Peer getPeer(String addr){
      return peerList.get(addr);
   }
}
```

However to permit exchange of information, relative to peers, between MainActivity and ChatActivity is used an apposite Singleton class called SingletonListOfPeer.

Then by using this code: `ListOfPeer mListOfPeer=new SingletonListOfPeer().getSingleton();`

the same reference, to the univoque HashTable, return.

6 Security

P2P secure Messenger allows people to exchange messages.

This app implements a protocol which guarantee the end-to-end encryption.

This end-to-end encryption protocol is designed to prevent third parties from having plaintext access to messages.

Terms

Public Key Types

- Identity Key Pair – A long-term RSA_1024 key pair, generated at install time (Public and private key).

Session Key Types

- Message Key – An 32-byte value that is used to encrypt message contents:
  - AES-256 key

6.1 Setup

At the setup time, the first launch a pair of identity key is created and stored in the memory of smartphone, the two keys are putted in two files: pub.key and pr.key respectively (in the P2P_key directory).

If the keys are not initializated a new pair are created.

```
//creating public_private key pair
    File sdCard = Environment.getExternalStorageDirectory();
    File dir = new File (sdCard.getAbsolutePath() + "/P2P_key");
    dir.mkdirs();
    File pub_file = new File(dir, "pub.key");
    File pr_file = new File(dir, "pr.key");
```

Exchanging Messages

Once a session has been established, clients exchange messages which are protected with the Message Key using AES256 in ECB mode for encryption.

This encryption avoid that a third-party can see the message content in plaintext.

6.2 Design and Implementation of the Key Exchange protocol with BAN logic

Burrows-Abadi-Needham logic, which is also known as BAN logic is a collections of rules and postulates for modelling and analysing the information security protocol. This logic is also a logic of beliefs, since it analyses the authentication protocols by deriving the beliefs from a series of logical deductions. It helps users to analyze and discover whether the protocol designed is secure against eavesdropping, or trustworthy, or both.

BAN logic is able to help us to answer the following questions :

- What is the aim of this protocol or what does this protocol to achieve?
- Are more assumptions needed for this protocol than for another?
- Is there any unnecessary part that could be left out without weakening this protocol?
- Is there any encrypted information that could be sent in clear text without weakening it?

6.2.1 Analysis Procedures

From a practical point of view, there are four main procedures for the analysis this logic. They are listed as follows: • Idealize the protocol from the original one.

- Set the assumptions about the initial state of proposed protocol.
- Logical formulas are added to the each statement of the protocol
- Logical postulates are applied to the assumptions and assertions. And the purpose of it is to discover beliefs in this protocol that the parties had.

These procedures may be repeated when the idealized protocol is refined or new assumptions are found as necessary. Step by step, this logic points the direction of protocol analysis from the initial assumptions to the final beliefs.

6.2.2 Basic Notation

• P| ≡ X: P believes X, or P is entitled to act as though X is true.

• P ◁ X: P sees X. Someone has sent a message containing X so that he can read

• P| ~ X: P once said X at some time, e.g. a used key K. Or A uttered a message containing X.

• P| ⇒ X: P has jurisdiction over X. P maybe an authority over X and can be trusted on X.

• #(X): X is fresh. It indicates that X has not been sent at any time before.

• P $\leftarrow^{K}\rightarrow$ Q: P and Q share key K. P and Q can use key K to communicate. The key is unknown to anyone else.

• $\xrightarrow{k}$P: P has public key K. P has a public key K, and the corresponding private key is K^{-1}

• P $\Leftarrow^{k}\Rightarrow$ Q: P and A share secret X. X is a secret only known by P, Q, or principals trusted by them. P and Q may use X to prove their identities to one another, e.g. X is a password for them.

• $\{X\}_k$: This denotes the formula X encrypted under the key K.
Formally, $\{X\}_k$is short for $\{X\}_k$*from* P.

• < X > $_k$: This devotes X combined with the formula Y. Y is intended to be secret and its presence proves the identity of whoever utters < X > $_k$. In implementations, X can simply be concatenated using the password Y. Y plays a special role as a proof of origin for X.

6.2.3 Logical Postulates

Message meaning rule

$$\frac{P| \equiv Q \overset{k}{\Longleftrightarrow} P, P \lhd \{x\}_k}{P| \equiv Q| \sim X} \qquad (1)$$

$$\frac{P| \equiv \overset{k}{\rightarrow} Q, P \lhd \{x\}_{k^{-1}}}{P| \equiv Q| \sim X} \qquad (2)$$

$$\frac{P| \equiv Q \overset{k}{\leftrightarrows} P, < P >_k}{P| \equiv Q| \sim X} \quad (3)$$

Nonce verification rule

$$\frac{P| \equiv \#(X), P| \equiv Q| \sim X}{P| \equiv Q| \equiv X} \quad (4)$$

Jurisdiction rule

$$\frac{P| \equiv Q| \equiv X, P| \equiv Q \Rightarrow X}{P| \equiv X} \quad (5)$$

Other rules

$$\frac{P| \equiv \#(X)}{P| \equiv \#(X,Y)} \quad (6)$$

$$\frac{P| \equiv (X,Y)}{P| \equiv X, P| \equiv Y} \quad (7)$$

$$\frac{P| \equiv Q| \equiv (X,Y)}{P| \equiv Q| \equiv X} \quad (8)$$

$$\frac{P| \equiv Q| \sim (X,Y)}{P| \equiv Q| \sim X} \quad (9)$$

6.2.4 Security Protocol Proposed

First, the protocol is introduced below in a simplified form. A denotes the initiator peer, B the other peer. K_A, K_B are the public keys of A and B, respectively.

K_A^{-1} K_B^{-1} are the secret keys that matches K_A and K_B.

The peer B generates K_{AB}, which becomes the session key between A and B.

N_a ,$N_{a'}$ N_b ,$N_{b'}$ are all the nonces that were generated by A and B respectively.

Real Security Protocol Proposed

1. $A \rightarrow B$: A , N_a (Chat request to B with my id)
2. $B \rightarrow A$: B, K_b, $E_{Kb-1}(A , N_a , N_b)$
3. $A \rightarrow B$: A , $E_{Ka-1}(A, K_a , N_b)$, $E_{Kb}(A, K_a , N_{a'})$
4. $B \rightarrow A$: B , $E_{Ka}(A, K_{ab} , N_{a'}, N_{b'})$
5. $A \rightarrow B$: $K_{ab}(N_{b'} - 1)$

The messages 2 and 3 are used for echanges public key, with message 3 i share the $N_{a'}$ secret which permit me to exchange K_{ab} in a safe way.

Objectives

$Key\ authentication$

$A| \equiv A \overset{K_a b}{\Longleftrightarrow} B$

$B| \equiv A \overset{K_a b}{\Longleftrightarrow} B$

$Key\ confirmation$

$A| \equiv B| \equiv A \overset{K_a b}{\Longleftrightarrow} B$

$B| \equiv A| \equiv A \overset{K_a b}{\Longleftrightarrow} B$

Assumption

$A| \equiv \overset{K_b}{\rightarrow} B \qquad A| \equiv \overset{K_a}{\rightarrow} A$

$B| \equiv \overset{K_a}{\rightarrow} A \qquad B| \equiv \overset{K_b}{\rightarrow} B$

$A| \equiv \#N_a \qquad B| \equiv \#N_b$

$B| \equiv B \Rightarrow A \overset{K_a b}{\leftrightarrow} B$

$A| \equiv B \Rightarrow A \overset{K_a b}{\leftrightarrow} B$

$B| \equiv A \overset{K_a b}{\leftrightarrow} B$

$A| \equiv B \Rightarrow \overset{K_b}{\rightarrow} B$

Idealized protocol

Now we transform the protocol. The idealized protocol is as follows:

1. A→B : Omitted since it does not contribute to the logical properties of the protocol
2. B→A : $\overset{K_b}{\rightarrow} B$, {A , N_a , N_b }$_{Kb-1}$
3. A→B : { A , $\overset{K_a}{\rightarrow} A$, $N_{a'}$, N_s }$_{Ka-1}$, {A , $\overset{K_a}{\rightarrow} A$, $N_{a'}$, $A \overset{N'_a}{\rightleftharpoons} B$}
4. B→A : {A , $A \overset{K_a}{\leftrightarrow} B$, $\#(A \overset{K_a}{\leftrightarrow} B)$, $N_{a'}$, $N_{s'}$}$_{Ka}$
5. A→B : {$N_{s'}$, $A \overset{K_a}{\leftrightarrow} B$}$_{Kas}$

Reasoning process for protocol

1. Message 1 is omitted, because it has no contribution to the logical properties of the protocol
2. From the message meaning postulate (2) we have $A| \equiv B| \sim \{A, N_a, N_b\}$ and since $A| \equiv \#N_a$ then we have from (6) postulate $A| \equiv \#\{A, N_a, N_b\}$ and form nonce verification postulate (4) we have $A| \equiv B| \equiv \{A, N_a, N_b\}$.
3. In this message we can add another assumption: $A| \equiv A \overset{N'_a}{\rightleftharpoons} B$ the A believes that $N_{a'}$ is a shared secret between A and B because A created $N_{a'}$ and only B can decrypt it because it is the only who could know his private key.

   From the message meaning postulate (2) we can find : $B| \equiv A| \sim \{\overset{K_a}{\rightarrow} A, A, N_b\}$ and

since $B| \equiv \#N_b$ then we obtain from (6) $B| \equiv \#\{\overset{K_a}{\rightarrow} A, A, N_b\}$.

Instead from nonce verification rule (4) we obtain $B| \equiv A| \equiv \{\overset{K_a}{\rightarrow} A, A, N_b\}$ and finally $B| \equiv A| \equiv \overset{K_a}{\rightarrow} A$ from rule (8).

4. From postulate (3) since $N_{a'}$ is a shared secret between A and B we obtain $A| \equiv B| \sim \{A, K_{ab}, N'_b\}$ and since $A| \equiv \#N_{a'}$ then for (6) we have $A| \equiv \#\{A \overset{K_{ab}}{\leftrightarrow} B\}$ and then for (9) and for nonce verification (4) we have $A| \equiv B| \equiv A \overset{K_{ab}}{\leftrightarrow} B$ and for jurisdiction rule (5) we have $A| \equiv A \overset{K_{ab}}{\leftrightarrow} B$.

5. Since $B| \equiv A \overset{K_{ab}}{\leftrightarrow} B$ (by assumption) we obtain by message meaning postulate (1) $B| \equiv A| \sim (N'_b, A \overset{K_{ab}}{\leftrightarrow} B)$ and from (6) we have $B| \equiv \#(N'_b, A \overset{K_{ab}}{\leftrightarrow} B)$ since $B| \equiv \#N'_b$, and finally from nonce verification (4) we obtain $B| \equiv A| \equiv (N'_b, A \overset{K_{ab}}{\leftrightarrow} B)$ and then form (8) we obtain $B| \equiv A| \equiv A \overset{K_{ab}}{\leftrightarrow} B$.