INSTITUTO SUPERIOR TÉCNICO
Departamento de Engenharia Informática
Security in Computers Networks and Systems
MEIC, 2016-2017 – 1st Semester
Taguspark Campus

Secure payments using SMS

Group T 07

68158
João Sousa

77958
Tiago Rosado

77965
Duarte Botelho

1 Problem

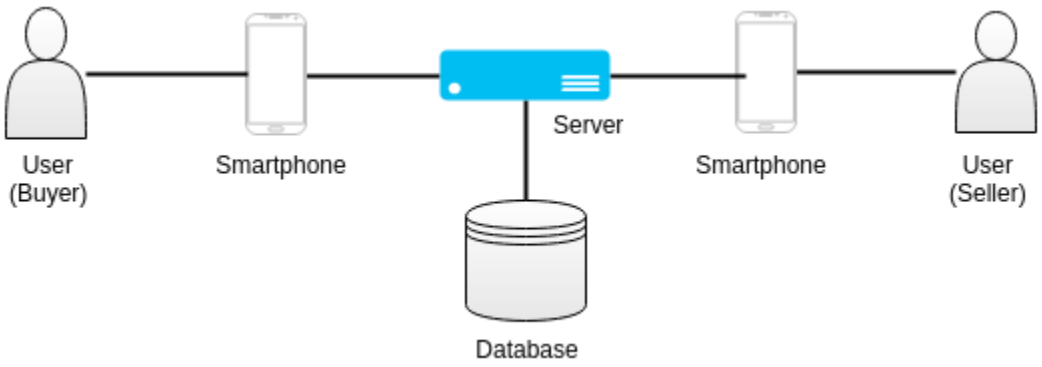
In today's world smartphones are part of everyone's life. In 2014 there were 1.5 billion[1] smartphone users. By the end of 2016 this number is expected to be almost 2.1 billion and will still be growing for more years. These numbers are a huge advantage to developers, because with a simple app they can change the lives of the population worldwide. So the best way to bring a service to the population is by making a small phone app that people of all ages can use easily and without mistakes.
Given this, what if a simple app that every user can download to their smartphone and configure in a few easy steps could do banking transactions even more securely and easier than the available traditional ways? This project is the solution to that problem.

2 Requirements

This solution will have the main following requirements:

- **Integrity:**
  The message cannot be modified by a man-in-the-middle. The message sent by the Android application needs to be the same that the server receives.
- **Confidentiality:**
  For example if the user Bob sends 4000€ to the user Alice, no one else except Bob and Alice and the server should know that this transaction took place.
- **Authenticity**
  Before a transaction from Bob to Alice takes place, Bob must prove that is really him that is trying to make the transaction. A man in the middle cannot impersonate Bob to send for example money for himself account.
- **Non-repudiation**
  If Bob sends a transaction to Alice, Bob cannot state that he didn't do it.
- **Double Spending**
  The same message cannot be sent twice. There must be something different in every message from the same user, even if the user wants to send the same amount of money to other user several times. This way, man-in-the-middle attacks are prevented.

3 Solution



[1] Numbers from https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

In order to achieve the proposed requirements we will be implementing a solution based on a personal smartphone that communicates with a main server to transfer money from one bank account to another. As UDP packets do not grant proper delivery and order we stand before an unreliable channel of communication. Given these conditions, a security mechanism will be built in attempt to secure this channel against attacks and network failures. We are going to use a framework to handle DB transactions with the server so our main goal is to establish a secure connection between clients and the server.

Basic Version

On this stage we will be creating a simple UDP message transmission protocol with max length of **120 bytes** to simulate SMS messages and implement bank transfers. Each message will have:

- (Destination + Origin) IBANs - 27 bytes
- Transaction value - 4 bytes
- Challenge-response to confirm the payment order

Intermediate Version

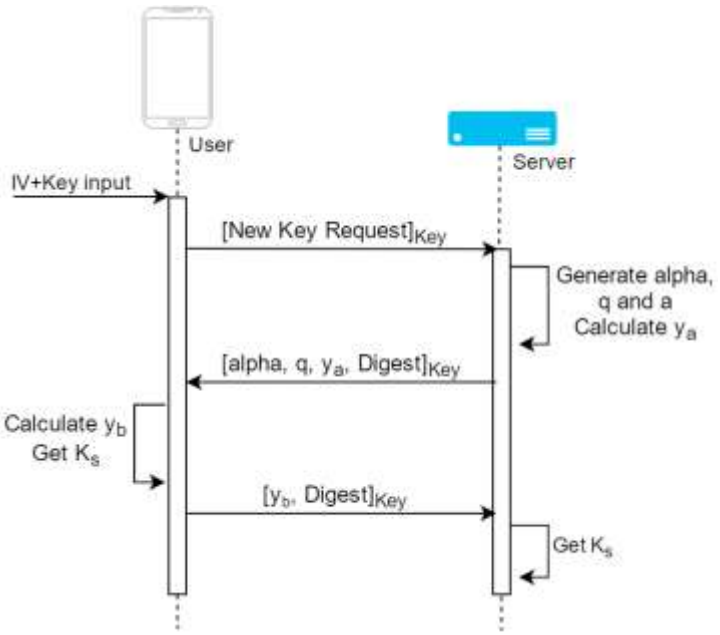
We will be implementing

- Message digest truncated SHA2 (to assure integrity) - 128 bit (16 bytes)
- Nounce UUID v4 (freshness) - 128 bit (16 bytes)

Advanced Version

- AES ciphered communication
- Money transfer limit within a certain time frame to prevent abusive usage

By the end of this project, each message will take 63 bytes. As we are using AES , the overall size of messages after ciphering is calculated according to: $(\frac{63}{16} + 1) * 16 = \mathbf{79\ bytes}$



The initial communication with the server will be done as the exemplified on the diagram shown at the right side of the page.

The bank will provide to the client a code containing an **IV** and a **Key**. The app will ask the user for this code and in the end of the setup process the smartphone sends a new key request to the server. The server calculates the **Diffie Hellman** parameters and sends multiple UDP messages to the user's smartphone containing all the information needed to perform the rest of the algorithm. Finally the user's smartphone sends the information that the server needs to conclude the algorithm and both share a new Key. **All the messages sent and received are encrypted by the initial key with AES to assure integrity**.

4 Results

Basic version implemented completely.

Intermediate version:

- The message digest algorithm used was SHA-256 however given the limitations regarding message size each hash generated is reduced to half of its size .
- Implemented nounces in Transaction UID's. Transaction UID's are generated through Java UUID library granting uniqueness and are verified through MySQL database properties. We implemented a table on database where every transaction is stored and its primary key is the Transaction ID. Given this setup if a replay attack on a transaction is executed, the MySQL database will throw an exception which will be handled by the server and a replay attack will be detected.

Advanced version:

- AES cipher communication implemented partially. In the server side (using a local client non-android), the cipher mechanism works as expected, nevertheless in the client side the behaviour is not as expected. We suspect this deviation appears from the utilization of different Base64 libraries, but the error sustains in the version submitted.
- It was implemented a 400€ wage limit daily based.

5 Evaluation

We will guide our solution discussion based on the requirements provided above:

- ➔ **Integrity:**
  Every packet sent and received has a digest which is verified in order to assure its integrity.
- ➔ **Confidentiality:**
  All messages changed between clients and the server are encrypted using AES with a symmetric key. Our intention was to use the Diffie-Hellman algorithm to obtain the key (the communication needed in this algorithm was encrypted by a secret, KEK), nevertheless, due to errors in the implementation the key used was the secret, not assuring perfect forward secrecy.
- ➔ **Authenticity**
  The messages sent and received from both the user and the server are encrypted with the key that only they know. No one else can encrypt the message with this key so the server knows that every message encrypted with a certain key can only be sent from the only client possessing this key.
- ➔ **Non-repudiation**
  The shared secret is only known by that client, ensuring that messages encrypted with that key are sent from that client.
- ➔ **Double Spending**
  The transactions identifiers used are unique.

Furthermore, all communication between the server and the database is **protected against SQL injection** (using jdbc with prepared statements).

6 Conclusion and Future Work

Our proposal was to use the life enhancement possibilities of mobile phones in day to day tasks, empowered by the technological evolution, such as doing bank transactions simpler and even in a more secure way. To do so, we have implemented a new smartphone application putting into practice the theoretical knowledge acquired in the course.
The main goal was achieved, even though with some bumps along the way. Our main frustration was being close to accomplish the implementation of the Diffie-Hellman algorithm and not be able to reflect it in our application. However, we think the evolution of the application and the final result offsets the dedicated effort in all the phases of the project.

7 Tool references

Spark http://sparkjava.com/(REST API) - Spark framework is a micro framework for creating web applications in Java 8 with minimal effort.
JDBC https://docs.oracle.com/javase/tutorial/jdbc/basics/gettingstarted.html - Java Database Connectivity (JDBC) is an API for the programming language Java, which defines how a client may access a database. It will be used to assure the communication between the server and the database.
MySQL http://www.mysql.com/ - MySQL is an open-source relational database management system (RDBMS). It will be used to create the entities necessary to maintain the application running.
Android API https://developer.android.com/reference/packages.html - Contains resource classes used by applications included in the platform and defines application permissions for system features.
ZXing https://github.com/zxing/zxing (QR Code) - Open-source, multi-format 1D/2D barcode image processing library implemented in Java, with ports to other languages. It will be used to generate and read QR Code images of clients IBAN.
Maven https://maven.apache.org/ - Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

8 . Attachments

8.1 - Client letter from the bank.

Caro cliente com o número de telemóvel 911174628,

Agradecemos desde já que tenha escolhido o serviço **Secure Payment** para fazer as suas transferências. Neste momento já contamos com 4 clientes, sendo que você (sim, você!) é o nosso 4º cliente! Siga os passos abaixo indicados para configurar a aplicação. Caso tenha alguma dúvida não hesite em contactar-nos através do nosso contacto telefónico ou em qualquer um dos nossos balcões espalhados pelo país.

1. Descarregue a aplicação para o sistema operativo Android através do link
   [LINK]

2. Abra a aplicação e leia atentamente os passos lá indicados para a correcta configuração do sistema.
   Caso prefira, pode usar o seguinte QR Code com o seu IBAN

3. Quando a aplicação pedir para ler o código secreto, aponte a câmara do dispositivo para o QR Code abaixo indicado para que este proceda à sua leitura. Se tiver problemas pode escrevê-lo manualmente

4. Já está, já pode usar e desfrutar deste fantástico sistema

Esperemos que a sua experiência de utilização seja excelente e que nos ajude a crescer.

Com os melhores cumprimentos,
O Seu Banco