Institution : JCT

Faculty : Computer Science

פרויקט סביבה בטוחה

Secure Banking

TEE

Instructor:

Barak Einav

Students:

Noam Semhoun 1721743

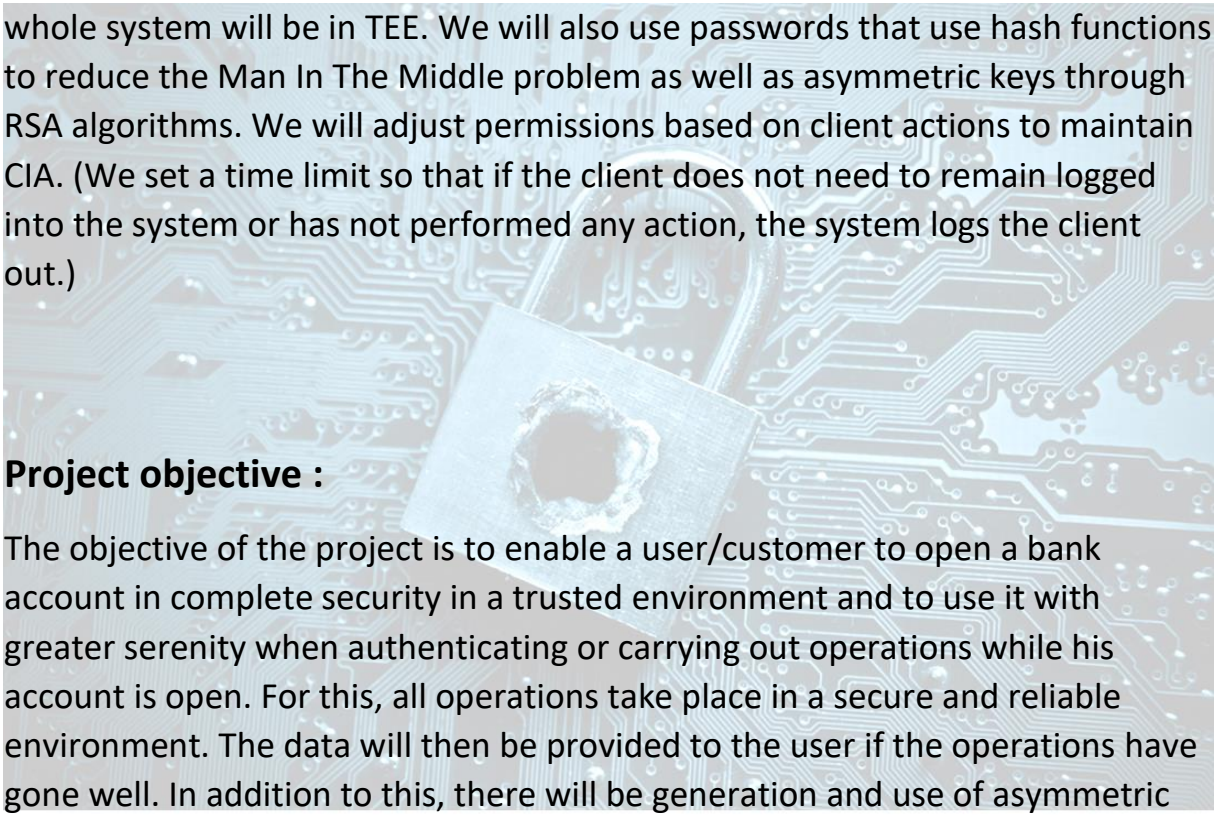
Yoël Obadia 1413422

Table of contents

Introduction:

Today, security is a challenge that requires constant and hard work against attackers in any computer system. Companies, hospitals, personal computers, organizations, banks are all examples of the daily life of malicious people in the field of security. Over time, all kinds of tools have been created to prevent as much as possible the attacks against computer system.

We will try to create, using several security tools, a system that prevents attacks on the computer systems of banks as much as possible. For this, the whole system will be in TEE. We will also use passwords that use hash functions to reduce the Man In The Middle problem as well as asymmetric keys through RSA algorithms. We will adjust permissions based on client actions to maintain CIA. (We set a time limit so that if the client does not need to remain logged into the system or has not performed any action, the system logs the client out.)



Project objective :

The objective of the project is to enable a user/customer to open a bank account in complete security in a trusted environment and to use it with greater serenity when authenticating or carrying out operations while his account is open. For this, all operations take place in a secure and reliable environment. The data will then be provided to the user if the operations have gone well. In addition to this, there will be generation and use of asymmetric keys to encrypt the password which complicates authentication for a malicious third party.

Problem resolved :

The problem ? that her generation has never seen bank robberies? embezzlement scam and theft of money...

The banking system must be ultra secure and must not allow any failure of the system in order to prevent anyone from entering an account and stealing money or even confidential data.

The issue here is creating a banking system in a secure environment with cybersecurity related functions for a safer and more serene experience for the user. He can thus create an account, authenticate himself and carry out banking operations such as a withdrawal of money or a deposit. The user also has the option of modifying his password if he wishes. All of this can be done through a clear graphical interface in order to facilitate the user's experience when choosing the various operations he would like to perform in order to be able to fully exploit his account.

Conception and Architecture :

The advantage of our solution will therefore be that the applet makes the intrusion almost impenetrable and therefore the data and interactions are secure in our application.

The project is composed of several layers. The base of the project is in visual studio which creates and opens sessions in the applet. It is also in visual studio that all the graphics are located. All the operations, when to them, are in the applet, that is to say in eclipse in a java file. The visual studio part is written in C#. The WPF creates sessions each time the user wants to perform an operation and will display a menu with the available options. The user will send the necessary data for the operation to run. If the data is not good, the operation will not take place and the user will be informed and can start again if he wishes. Operations are performed in the applet without anyone having access to it. Once the operation has been completed, the user is sent back what was requested. Thus, this structure allows for increased security because everything is hidden and also very complicated to penetrate.

SECURITY ANALYSIS

Mode of work :

Each of us mainly focused on one IDE. The Visual studio part with the implementation of the sessions and the graphic part were mainly carried out by Noam Semhoun while the Eclipse part was mainly developed by Yoël Obadia. (this does not prevent us from knowing the project in its entirety, of course...)
To achieve the coherence of the project, we stayed in touch with a development of our various advances and findings in order to coordinate the two works well. The Eclipse part once completed allowed a faster progress of the graphics because we knew what data to return and display. Then there was the part of the tests to verify the efficiency of the creation of the sessions and that the data was indeed in the Applet and that the functions worked well. This caused a lot of difficulties with a little unknown progress for the graphics while waiting for the functions of the Applet. Then there were also difficulties in understanding through the graphical interface the various bugs at several levels: conversion problems with bytes, memory conservation problems, verification problems and technical problems with the software which was a great source of problems that have slowed down our progress in the project enormously.

Besides, here are for your information some examples of bugs with which we had to measure ourselves, and which for some were not at all won in advance

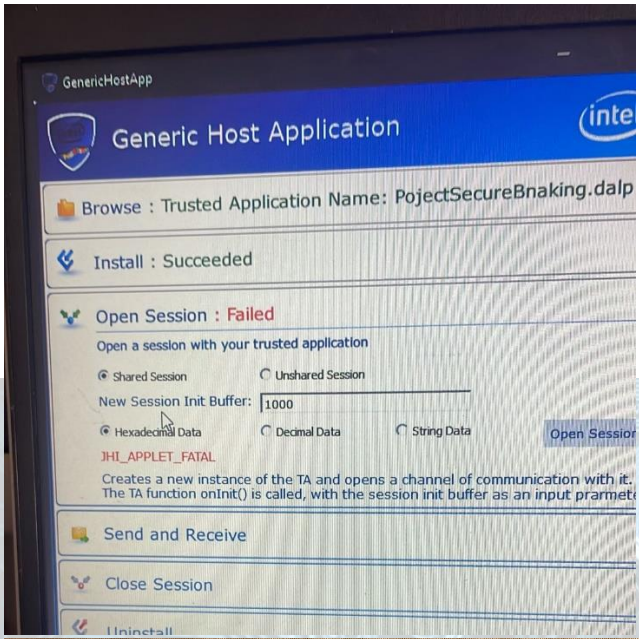
- First for reasons that are still unknown to us, in our version of Visual Studio 2019 created by our Appet, and despite the installed plugins, we did not have the possibility to implement a WPF graphical interface (hence our email exchanges).
  And we don't see the bank account details displayed on the console... :)
  After dozens of hours of research we had miraculously succeeded in integrating a semblance of an external WPF window (who obviously didn't want to connect to the applet by any means :) to which, thanks to small vices and forced methods, we were able to patch up these references to our project (I'm not talking about the difficulty importing everything in a single window)
  No other solution was possible...

- One of the biggest enemies of security is the convenience of developer interactions:
  To transfer so much data to the account, with completely different type fields, through only one allowed byte table!
  Indeed, the only way to communicate with the applet was through the byte[] type, not very practical when you have to compare arrays or list of int and double digits or string password etc...
  we had to convert, then stack the data inside then separate them and reconvert them to process them and re-stack them to return them to our interface....
  We have through this learned by example, to use conversions from hexadeximat to octalASCII etc...
  Photo examples:

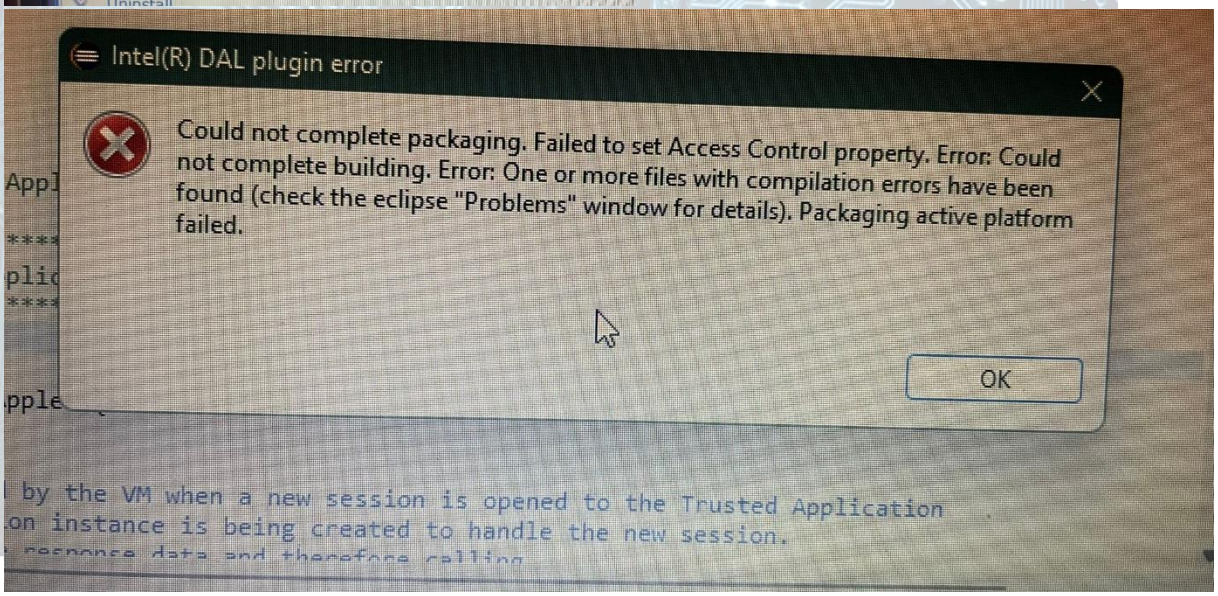
```
byte[] response = new byte[1];
byte[] arrayAccount=new byte[FlashStorage.getFlashDataSize(0)];

 //   request =     0-19   Noam Semhoun 9999123   31: 0944334444

/*       0-9 => FirstName    10 BYTE max
 *       10 => " "
 *       11-20 => LastName   10 BYTE max
 *       21 => " "
 *       22-27 => Balance    6 BYTE max
 *       28 => " "
 *       29-37 => TZ         9 BYTE max
 *       38 => " "
 *       39-46 => pwd        8 BYTE max   (min 5)
 *       //*/
    // arrayAccount = new byte[FlashStorage.getFlashDataSize(0)];

    FlashStorage.writeFlashData(0, request, 0, request.length );
    FlashStorage.readFlashData(0, arrayAccount, 0);  // copie du

    response[0] = 0;  // Ca a marché

    if (arrayAccount.length == 0)
        {
          response[0] = 1;
        }

    // arrayCrypto = createKeys(arrayAccount[39]); */
```

- After our project was almost finished and a few days before the submission date, a strange bug occurred in our code that no longer allowed us to create any sessions via the JHI applet: at that point, we had to try again to create about fifteen new projects and by reinstalling the

whole process several times before finding our open sessions with the help of a member of Intel...



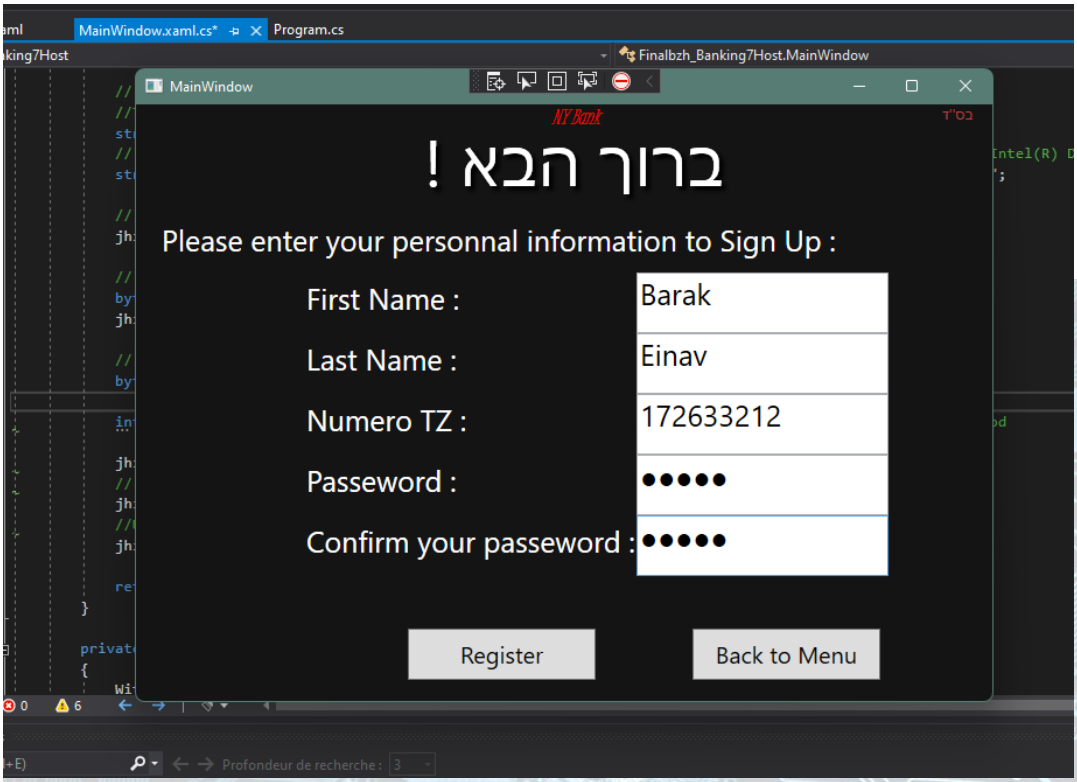
-



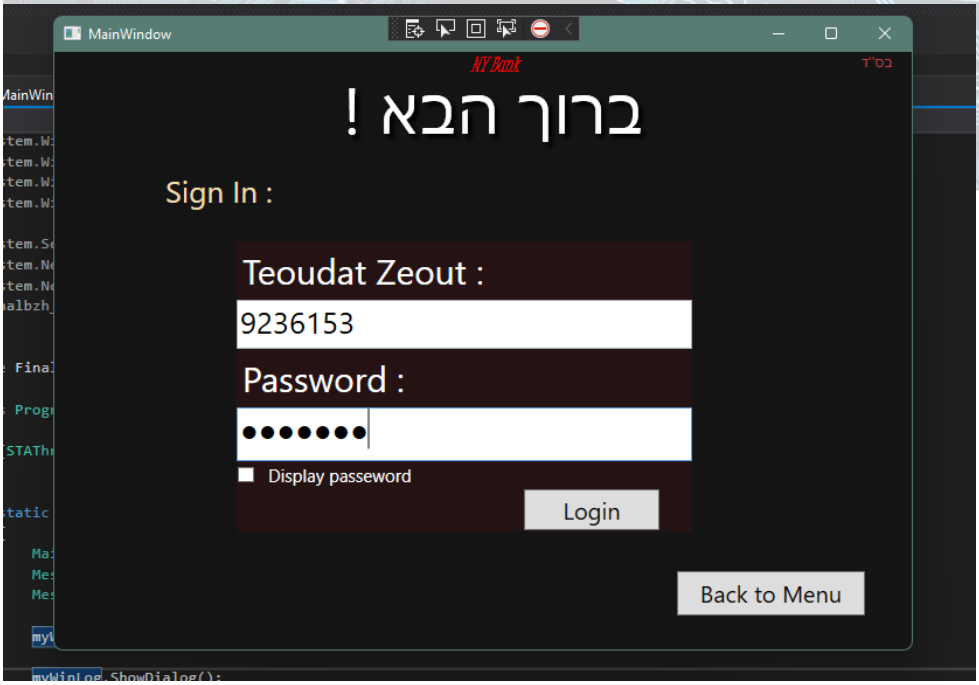
-

Program execution :

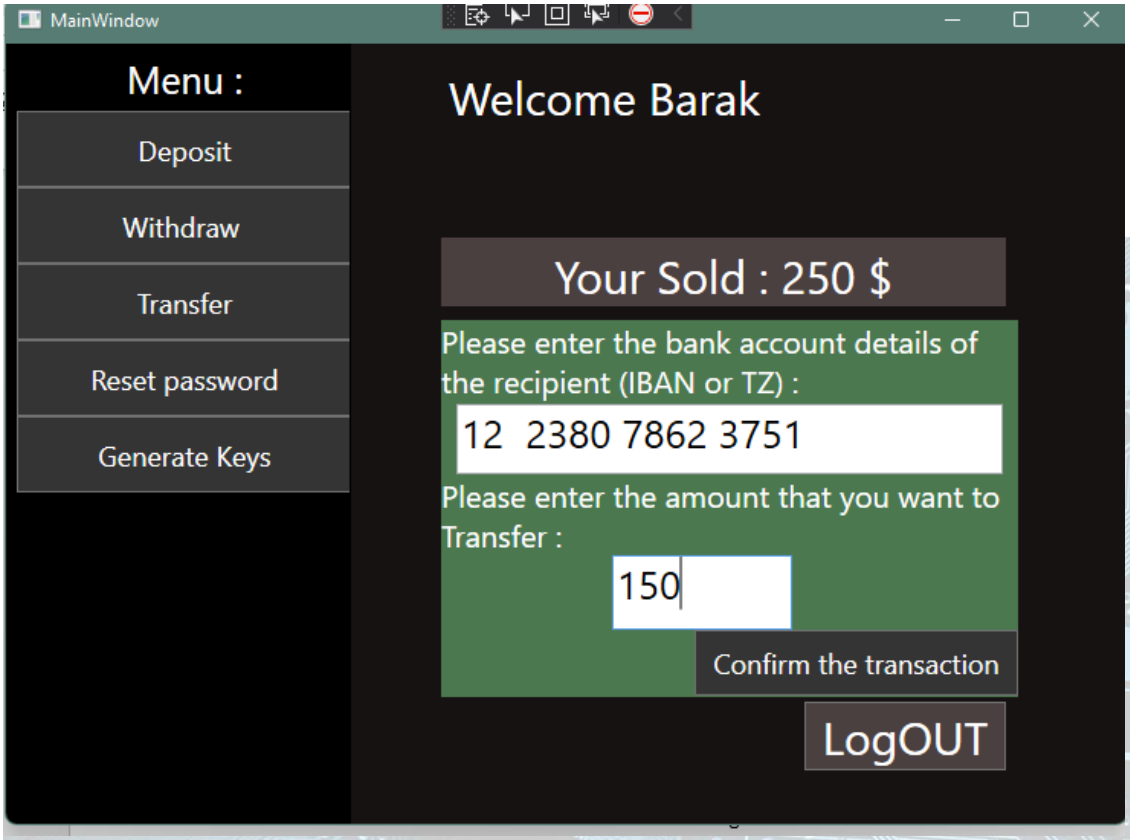
in the smooth running of the application, we first invite the customer who has registered as a new customer by completing the following form (mot de passe sécurisé, double authetification,...) :



Tthen he will receive these identifiers and will be able to connect to his platform through this fairly realistic graphical interface :



He then found himself in his client interface which displays the balance of his bank account (which he obviously filled in when he registered) and which also displays certain bank functions such as different transactions (deposit, withdraw; make a transfer, etc.)



Our application makes it possible to manage them all with security thanks to the communication of these functions with the applet.

Future work :

This project is designed for a single client. It would be necessary to be able to allow several customers to register and suddenly connect the project to a server so that they can interact with each other. This means making transactions to other customers. We could add an IBAN to customers in order to allow

transactions between users. This would offer even more security with additional personal data but also a more complete project. We could also add a savings or interest option to improve the model already established. These ideas were planned but due to the many problems encountered and the passage of time, we implemented an effective system for a single user.

Our challenge in the next stage will be to complete the security features that we have started (thinking that we will have the possibility of completing them of course) which among other things use the RSA algorithm to generate keys and control the strength of the pass of password and signature of our transactions by each user.