

Security Report - Hearts Game

Universidade de Aveiro

Alexandre Lourenço, João Monteiro



Version 1.0

Security Report - Hearts Game

Departamento de Eletrónica, Telecomunicações e
Informática
Universidade de Aveiro

Group 11

(79894) alexandre.lourenco@ua.pt

(80272) joaomonteiro24@ua.pt

January 29, 2020

Contents

1	Introduction to the project	1
2	Functionalities implemented	2
3	Code organization	3
3.1	threadedServer.py	3
3.2	client.py	3
3.3	game.py	3
3.4	cryptos.py	3
3.5	common.py	3
3.6	constants.py	4
4	Citizen Card	5
4.1	Authenticate player	5
4.2	With or without CC	6
4.3	Join Table	6
5	Communications	7
5.1	Between player and croupier	7
5.2	Between player and player	7
6	Deck ciphering and distribution process	9
6.1	Deck ciphering	9
6.2	Deck distribution	9
6.3	Hand deciphering	10
6.4	Selection of the symmetric cipher mode	10
7	Bit commitment	11
7.1	Generating and sharing	11
7.2	Utilization	11

8	Game process	13
8.1	Play a Game	13
8.2	Cheating	14
8.3	Game accounting	14
8.4	Accounting receipts	14
8.5	Rules of the game	15
9	Observations	16
9.1	Problems and deficiencies	16
10	Requirements	17
11	Conclusions	18

Chapter 1

Introduction to the project

The project in question is a solution to the challenge presented to us within the scope of the Security course unit: the develop of a secure hearts game.

The game will consist in four players and a croupier.

In order to play this game, the player will use their citizen card to be able to authenticate himself.

In this report we will explain all the stages we been through to achieve this project.

For implement this project we choose the python language.

Chapter 2

Functionalities implemented

- ★ Protection of the messages exchanged (in chapter 5)
- ★ Identification of users in a croupier with Citizen Card (in chapter 4)
- ★ Set up of sessions between players and a croupier (in chapter 5)
- ★ Set up of sessions between players (in chapter 5)
- ★ Agreement to join a table (in chapter 4)
- ★ Deck secure distribution protocol (in chapter 6)
- ★ Selection of the symmetric cipher mode by the player (in chapter 6)
- ★ Validate the cards played during a game by each player (in chapter 8)
- ★ Protest against cheating (in chapter 8)
- ★ Possibility of cheating (in chapter 8)
- ★ Game accounting and storage of accounting receipts (in chapter 8)

Chapter 3

Code organization

Architecture of the project.

3.1 `threadedServer.py`

Contains all the necessary functions for croupier (server) management.

3.2 `client.py`

Contains all the necessary functions for player management.

It was used a thread structure in which each player is a thread.

3.3 `game.py`

Contains all the necessary functions for game management.

Also contains the table informations.

3.4 `cryptos.py`

Contains all the necessary functions for citizen card management and all the ciphering/deciphering (Elliptic Curve, Diffie-Hellman, etc.) functions.

3.5 `common.py`

Contains the functions of sending and receiving content (signed or ciphered).

3.6 constants.py

Contains all the constants used in the project allowing a better organization of the code.

Chapter 4

Citizen Card

The citizen card is a core part of the project since it will be responsible for authenticating the player to the croupier and to the other players.

To allow the interaction with the citizen card we use the PyKCS11 library.

After the introduction of the right PIN we extract the citizen certificate (which will be signed with the citizen private key) so we can proceed to authenticate the player in the croupier and in the other players.

It's also with the help of the PyKCS11 library that we will be able to sign all the data needed.

4.1 Authenticate player

After the insertion of the PIN of the citizen card, we save in a JSON format the following informations:

- ★ certificate of the citizen card
- ★ public key (from an asymmetric key previously generated using Elliptic Curve)
- ★ username extracted from the citizen card

The player will then sign this JSON using his citizen card private key and send it to the croupier with the citizen card certificate and the JSON message (data) so the croupier can verify his signature using the public key presented in the certificate and finally be sure of the identity of the player.

After this validation the croupier will use the player certificate to validate the chain of trust.

4.2 With or without CC

When a client wants to create a player he can decide if will use a Citizen Card for authentication or not. If he decide to use¹ it than the flow will be the explain above, if he decide not to use² the CC the values needed for authentication and the username will be randomly generated.

4.3 Join Table

After a player prove his identity the croupier will show him the players that are already in that table (their usernames) and will ask the player if he will like to join them or not (if the player refuse, his thread will be killed and his seat will be free for another player to use).

¹In this case, the command to run the player in the terminal should be: `python3 client.py CC`

²In this case, the command to run the player in the terminal should be: `python3 client.py`

Chapter 5

Communications

Every message exchange will be signed using Elliptic Curve or ciphered using Elliptic Curve Diffie–Hellman (ECDH) algorithm, depending on the sensitivity of the data, as we will explain better in the next sections.

All communications are done with messages in JSON format.

5.1 Between player and croupier

The player and the croupier will communicate using signed messages.

Both the croupier and the player will at the beginning of their life generate a pair of asymmetric keys using the Elliptic Curve algorithm and will later proceed to a exchange of public keys (the public key of the player will be signed as we explain in chapter 4).

Now, every time that the player want to communicate with the croupier he will sign his message using his private key and the croupier will be able to verify the signature with the public key and be sure of the authenticity of the source of that message.

The same applies to messages from the croupier to the player.

5.2 Between player and player

For the communication between players we will cipher¹ the messages using the ECDH algorithm.

To establish the communication, the messages will always pass throw the croupier which will than send to the wanted player, however as the message

¹This method will not be use when playing a card in the game phase once every card needs to be understand by the croupier

will be ciphered using a derivation of a shared key between that pair of players, the croupier will not be able to decipher it's content.

For the generation of the shared key the players will need to possess in the first place the public keys of each other (generated using the Elliptic Curve at the beginning). After that, for each communication the pair of players will generate a shared key using for that the public key of the player that we want to establish communication and our private key. At the end, we will derive this shared key (which will be equal in both players) and we will use it to cipher and decipher every content between this two players.

Chapter 6

Deck ciphering and distribution process

6.1 Deck ciphering

The process starts with a ordered deck will all the cards in the croupier.

To start the process of shuffle with ciphering the deck will be sent from the croupier to one of the four players randomly.

The chosen player, when receiving the deck, first of all will create a symmetric key, using the SHA-1 algorithm, that will be needed to cipher each card of the deck.

After that, the player will cipher each card using the AES algorithm with the encryption mode CBC (by default) and the previously generated key.

When the player have all the cards ciphered than he will be in conditions to shuffle the deck.

After that, the player will send the deck (ciphered) to the next player (using the communication sessions explained in chapter 5) which will apply the same method resulting in a second layer of ciphers and so on, until the deck arrives to the fourth player that will apply the fourth layer of ciphering before sending it back to the first player.

At this point the cards are unrecognizable for any of the players (including the croupier).

6.2 Deck distribution

For the process of distribution the deck will circulate (still ciphered) among the players, who will choose, based on a probability-based mechanism,

if they want to pick a card, trade one or more cards (if they have more than one card in hand at that moment) or do nothing.

The probability of picking a card (by default is 50%) can be easily modified just by updating the *PERCENTAGE CHANCE* variable present in constants.py.

6.3 Hand deciphering

This process takes place after all players have made their bit Commitment and have already sent it to the croupier and each player (which will be later explained in the chapter 7).

To decipher the hand we will use the reverse sequence that was used for ciphering (the players will share among them the symmetric keys used to cipher the cards and after that each player will decipher each layer of ciphering using the keys received).

6.4 Selection of the symmetric cipher mode

For the symmetric cipher used in this distribution, the implemented algorithm (as already mentioned) is AES, however the cipher mode can be chosen by the players (a method previously agreed upon by all). They will be able to choose between CBC (default), OFB or CFB.

Chapter 7

Bit commitment

The bit Commitment is needed to ensure veracity of a hand (to prove that some player is cheating if needed).

7.1 Generating and sharing

As was approach before (in the chapter 6) the bit Commitment is done when all the players already have a full hand still ciphered, not allowing to anyone to perform modifications¹.

To compute the bit Commitment we use the method that was suggested using for that: a one-way hash function (h) and two random values ($R1, R2$), where the C represent the cards and b the bit commitment

$$b = h(R1, R2, C)$$

To do this the $R1$ and $R2$ are randomly generated and stored in each player and for the one-way hashing we are using the SHA256 algorithm.

When all the players have computed their respective bit Commitment they will share with all the other players (and with the croupier) their $R1$ and their b (bit Commitment) signed (to ensure that cannot be modified by no one).

7.2 Utilization

If some player decide to accuse someone of cheating (when that one plays a card that belongs to the player hand, for example), the player who perform the accusation will send to the croupier and all the players his $R2$ and his

¹At the moment of sending the bit commitment the player will sign it

initial hand still ciphered (C) so that everyone can compute the one-way hash and verify if the previously received bit commitment (b) matches the result. If it does, the croupier will use the symmetric keys (used to cipher the cards four times for the players in the Deck ciphering section of the chapter 6) to decipher the hand of the player that make the accusation and compare with all the cards played (of all players) in the last round and decide if the cheat is real or not.

Chapter 8

Game process

8.1 Play a Game

After a player accept to join a table (chapter 4) he will need to wait until the table is full to start the game.

When all distributions and deciphers have been made will be shown to the player who owns the two of clubs (2C) the following menu:

```
Current hand:
|  1|  2|  3|  4|
|----|----|----|----|
|    |    |    |    |

My Cards:

|----|----|----|----|----|----|----|----|----|----|----|----|
| TCl 3Cl 4Hl 3Hl 5Sl KCl ASl QSl 6Cl 5Hl 2Sl 9Cl 2Cl

Please select a card to play or complain (00):
```

Figure 8.1: Menu of the game with current played hand, hand of the player and the option input

This menu will show the current play (current hand: will be update every time a card is played), the player's cards and will accept two kinds of input: a card to play¹ (that needs to exist in the full deck), or a code '00' that's used

¹The card will be sign so the croupier can validate and pass it to the next player

to declare someone cheated in the last (or current, depending if the player already played in that round or not) play.

8.2 Cheating

There are three kinds of cheating:

- ★ A player can play a card that is not in his hand.
For example, in the 8.2 image the player can cheat by playing the 5C card.
- ★ A player can play a repeated card.
For example, in the 8.2 image the player can play two times the 3C card.
- ★ A player don't assist to a suit.
For example, the player needs to play a card of clubs an instead of that plays a card of hearts.

When some other player notice/thinks that a player is cheating (for example, in the first case when the player who owns the 5C card notice that some other player used his card) he can declare that someone cheated in that play ('00' code) and the game will be stopped.

The croupier will use the bit commitment to prove that one or more players cheated as we explained in the Utilization section in the (chapter 7).

In all this cases, the final decision about the cheating status is made by the croupier.

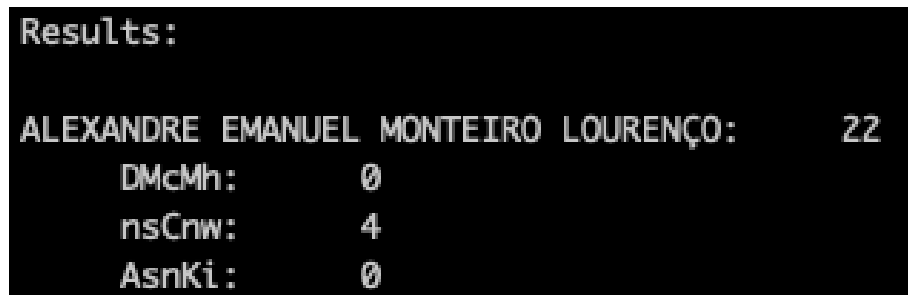
8.3 Game accounting

All the game accounting will be perform in the croupier side and will at the end of the game be shared with the players.

8.4 Accounting receipts

At the end of the game, the croupier will share with the players a record of all plays made (history) and the results of the game.

Each player will than sign that information with their citizen card and store it in a file with the date and hour of the game.



Results:	
ALEXANDRE EMANUEL MONTEIRO LOURENÇO:	22
DMcMh:	0
nsCnw:	4
AsnKi:	0

Figure 8.2: Final accounting provided by the croupier

Later, if needed, will be possibly to prove the authenticity of the data by verifying the signature of the information in the file with the public key of our citizen card certificate.

8.5 Rules of the game

- ★ The first player will be the one with the 2C in his hand
- ★ Hearts cards can only be played when a player have no more cards of a particular suit
- ★ A player needs (always) to assist to the first suit played in that round
- ★ Each heart card has a value of 1 point
- ★ The queen of spades (QS) has a value of 13 points
- ★ The player with less points at the end of the game will win the match

Chapter 9

Observations

9.1 Problems and deficiencies

★ At the beginning of the communication between the player and the dealer, when there is an exchange of public keys between them, there's no verification of the public key of the dealer (except the verification of the signature made by himself) thus enabling the existence of a MITM (Man In The Middle) attack in order to steal information.

★ Due to an unstable behavior of the CC software we are faced with the access to CC with digital signature limited which did not always allow us to check the certificate chain every time. AS we needed that validation to proceed in the game we decided to temporarily disabled this security mechanism. However, we still validate the authenticity of the players as we explain in chapter 4.

★ The croupier cannot verify if there was some kind of cheating in the present play (he can only see the plays that are in the historic) making it only possible to him to check the previous plays.

Chapter 10

Requirements

To execute this game we will need to have:

- ★ python3
 - ★ PyKCS11
 - ★ Cryptography
 - ★ PyOpenSSL
- ★ PKI of the Portuguese Citizen Card

Chapter 11

Conclusions

The development of this project allowed us to improve our knowledge of cryptographic methods learned in the Security course unit.

We realized that creating any kind of secure game or application requires a very meticulous working method, good programming practices as well as a good knowledge of cryptographic methods (having to consider all the possibilities of information leaks to preserve the good behavior of the program).

With regard to the user experience, it was found that the quality of this tends to decrease with the increase of security in that system.