

# Four in a row online

Project of Foundation of Cybersecurity

Barsanti Nicola, Tumminelli Gianluca

2019-2020

## Contents

<b>1 Requirement Analysis</b>	<b>2</b>
1.1 Specification Document . . . . .	2
1.2 System Requirements . . . . .	3
1.3 System Specification Document . . . . .	5
<b>2 Application Mockups</b>	<b>8</b>
<b>3 Message Flow</b>	<b>9</b>
<b>4 UML Diagram</b>	<b>9</b>

# 1 Requirement Analysis

This paper will document the development of the game *Four in a row online*. The application is a multiplayer online game accessed by a graphic interface where each user must be authenticated before access the application and his communication must be confidential and secure. When logged the user could choose another player, start a match and then they could talk each other and play the game. The application will have a score table where there will be printed general statistics about all the users of the application.

## 1.1 Specification Document

- The user will access the application through a GUI
- The user will access the application remotely
- The user could register a new account
- The user messages must be secure, confidential and authenticated
- The user could log-out from the application
- **The user must authenticate to access the application**
  - The user will use a username and password to login
- **The user could see all the available players and interact with them**
  - The user will have a list of all the free users available
  - The user will select an adversary and send him a challenge
  - The user will see all the pending challenges
  - The user could withdraw a previous sended challenge
  - The user could accept or reject a challenge
- **The user could play a match with other players**
  - The user will have a dedicated window for play a match
  - A match is played by two users
  - A match is composed by rounds
  - In each round the control is assigned to the opposite player
  - In the first round the control is assigned to the player who has sent the challenge
  - In each round the user in charge selects an available column of the gameboard
  - Each round lasts a maximum of 15s

- The first user who put four tokens in a row win
- If all the gameboard is full without a winner the match ends with a tie
- The user could logout from a match
- The user who left a game automatically loses
- **During a match users could talk**
  - The user during a match became unavailable for challenges and reject automatically all the pending requests
  - Into the game window there will be a chat
  - The user during a match could always read messages from the chat
  - The user during a match could always write a message into the chat
- **The application has a rank table**
  - The user could see the ranks of all the users
  - A rank will be defined as  $\langle TotalMatch, WonMatch, LostMatch, TieMatch \rangle$

## 1.2 System Requirements

- **The application is composed by a server and several clients which communicate remotely**
  - A client-server protocol is adopt for the communication between clients and server
  - A peer-to-peer protocol is adopt for the communication between clients
  - All messages must be confidential
  - All messages must be protected from replay
  - All messages must be authenticated
  - All messages must be sanified before being used by the application
  - There will be a symmetric session key for each sendd message
  - Each user will have a personal RSA key
  - The private key of the user must be stored securely by the clients
  - The user RSA key will be generated by the client
  - The server will have a personal RSA key
  - The server will store all the PKE public keys of the users
  - RSA key will be used by users to cipher the peer-to-peer session keys and authenticate them
  - RSA key will be used by the server to cipher the client-server session keys and authenticate them

- **the GUI is composed by four windows:**
  - a Login/SignUp window
  - a Main window
  - a Game window
  - a Rank window
- **The first window showed is the *Login/SignUp* window**
  - The user could login with a username and a password
  - The user could register giving a username and a password
- **After a successfull login will be showed the *Main* window**
  - Into the main window there will be a list of the current player
  - Each player in the list shows its username and percentage of wins
  - The user can choose a user and send him a challenge
  - The user can make only one challenge a time
  - The user can withdraw the currently sended challenge
  - The user can see a list of all the pending received challenges
  - The user can reject a pending challenge
  - The user can accept a pending challenge
  - The user can reject directly all the pending challenges
  - The user can log-out from the application and return to the *Login/SignUp* window
  - The user can change the current window to the *Rank* window
- **The *Rank* window is accessible only from the *Main* window**
  - The user can see statistics of all the registered users
  - The statistics are defined as  $\langle TotalMatch, WonMatch, LostMatch, TieMatch \rangle$  all messages must be sanitized before being used by the application
  - The user could return to the *Main* window
- **If the user is active whenever a challenge is accepted the current window is changed to the *Game* window**
  - The *Game* window has a chat
    - \* The user during a match could always write into the chat
    - \* The chat is updated upon receipt of a message
  - The *Game* window has a matrix of 6x7 as *Gameboard*
  - The match is composed by rounds
  - During a round only one user can insert one token into the gameboard

- The player who can insert the token is changed after each round
- The first player who inserts four tokens in a row wins
- If the matrix is full with no winner the match ends with a tie
- Each round lasts at maximum 15 seconds
- The user can log-out from a match and return to the main window
- If a user log-out from a match he loses automatically

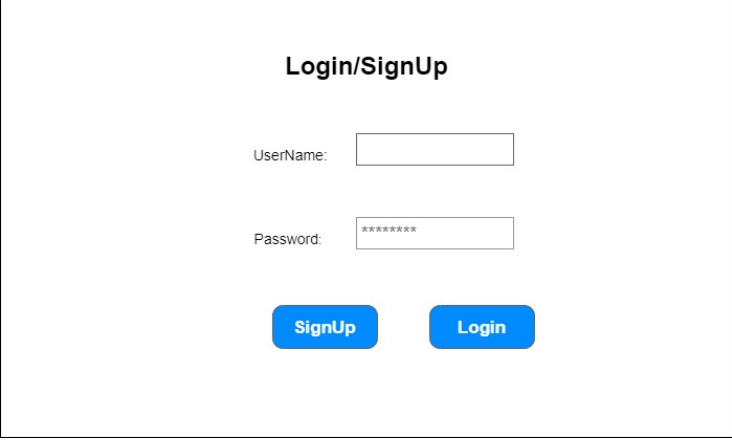
### 1.3 System Specification Document

- The application will be implemented in C++ with Secure Coding
- The application will use OpenSSL library for crypto algorithms
- The application will use .NET libraries to implement the GUI
- The server application will use a MySQL database to store information about registered accounts and stored public RSA keys
- The client and the server have a RSA key
- The RSA private key must be crypted using the user/server\_administrator password
- The client must obtain the server RSA certificate from the server
- The server will register all the clients connected to the application
  - For each client it will maintain a clientID and the IP address, port and socket of the communication
- The server will register all the logged users
  - For each user it will maintain the  $\langle clientID, username, public\_key, AES_{key} \rangle$
- The server will register all the active match
  - For each match it will maintain the username of the users and a status of the match
  - The status of a match can be OPENED,STARTED,PLAYED,CLOSED
- **The application will have a login window**
  - The login window will have a *Username* field
  - The login window will have a *Password* field
  - The login window will have a *SignUp* button
  - The login window will have a *Login* button
  - On click on the SignUp button the Username and Password field must be non empty

- During login the client will send its username to the server
  - \* Message must be crypted with the client private RSA key
  - \* Message must prevent small message attack
  - \* Message must be secured against replay
  - \* Message must be protected against corruption
- The client will receive after a login request a **LOGIN\_SUCCESS**, **LOGIN\_FAILURE** response
  - \* Message must be crypted with the client public RSA key
  - \* Message must be protected against replay
  - \* Message must prevent small message attack
  - \* Message must be protected against corruption
- During a signup the user will send its username and its public key to the server
  - \* Message must be crypted with the server RSA public key
  - \* Message must prevent small message attack
  - \* Message must be secured against replay
  - \* The clients generate during each registration a new RSA certificate and save it
  - \* Message must be protected against corruption
- When a server receives a login request it will control the authentication
- After a successfull login server will generate an AES symmetric session key for the communications with the client
- The client will receive after a SignUp request a **SIGN\_SUCCESS** message including an AES symmetric session key or a **SIGN\_FAILURE** response
  - \* Message must be crypted with the client public RSA key
  - \* Message must be protected against replay
  - \* Message must prevent small message attack
  - \* Message must be protected against corruption
- **After a successfull login the first page showed will be the Main Page**
  - The server will send a list of all the currently active users
  - Each user in the active users list is defined by its username and the percentage of the wonned matches
  - The server periodically will send an *Update Message* to update the currently active users
  - An update message is a list of username with an ADD/REM status associated

- Messages must be crypted with the AES symmetric session key
- Messages must be protected against replay
- Messages must be protected against corruption
- There will be a list of all the active users
- The user could click on a listed active user and send him a challenge by select him and click on the *Challenge Button*
- After the send of a challenge the *Withdraw Button* became available
- After the send of a challenge the *Challenge Button* became unavailable
- After the withdraw of a challenge the *Withdraw Button* became unavailable
- After the withdraw of a challenge the *Challenge Button* became available
- Clicking on the Unchallenge button will undo the challenge request
- In the main page there will be a list of all the pending challenge
- The challenge pending list will have an *Accept Button*
- The challenge pending list will have a *Reject Button*
- The challenge pending list will have a *Clean Button*
- If there isn't pending challenges the *Reject Button* is unavailable
- If there isn't pending challenges the *Clean Button* is unavailable
- If there isn't pending challenges the *Accept Button* is unavailable
- The user could select one user in the pending challenge list and accept his request by clicking on Accept button
- The user could select one user in the pending challenge list and reject his request by clicking on Reject button
- The user could delete all the challenge pending list by clicking on Clean button
- When a user Accept a challenge sends an ACCEPT message with the adversary username to the server and an AES session key
- When a user Accept a challenge the clients generate a new AES key to protect the peer-to-peer connection
- When a user sends a challenge sends a CHALLENGE message with the adversary username to the server
- When the server receives a CHALLENGE message control if it's a valid request
- When the server receives a valid CHALLENGE request it forward the message to the target user
- When the server receives an ACCEPT message controls if it's a valid request
- When the server receives a valid ACCEPT request it formward the message and the key to the target user

## 2 Application Mockups



The mockup shows a simple login and sign-up interface. At the top, the title "Login/SignUp" is centered. Below it, there are two input fields: "UserName:" followed by a text box, and "Password:" followed by a password box with asterisks. At the bottom, there are two blue buttons: "SignUp" and "Login".

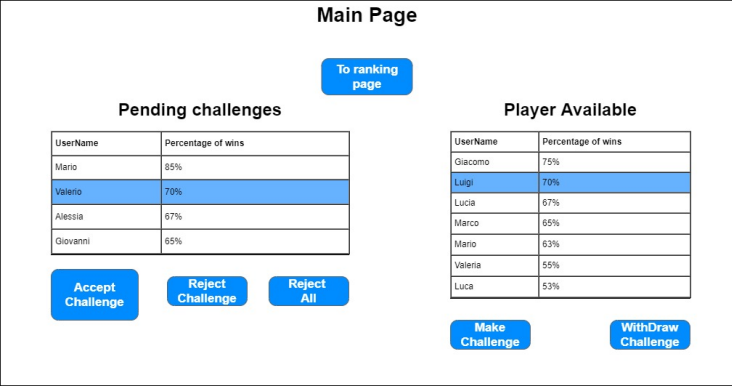
**Login/SignUp**

UserName:

Password:

**SignUp** **Login**

Figure 1: Login/SignUp page mockup



The mockup shows the main page of the application. At the top, the title "Main Page" is centered. Below it, there is a button "To ranking page". The page is divided into two main sections: "Pending challenges" and "Player Available". Each section contains a table with "UserName" and "Percentage of wins". Below the "Pending challenges" table are three buttons: "Accept Challenge", "Reject Challenge", and "Reject All". Below the "Player Available" table are two buttons: "Make Challenge" and "WithDraw Challenge".

**Main Page**

**To ranking page**

**Pending challenges**

UserName	Percentage of wins
Mario	85%
Valerio	70%
Alessia	67%
Giovanni	65%

**Accept Challenge** **Reject Challenge** **Reject All**

**Player Available**

UserName	Percentage of wins
Giacomo	75%
Luigi	70%
Lucia	67%
Marco	65%
Mario	63%
Valeria	55%
Luca	53%

**Make Challenge** **WithDraw Challenge**

Figure 2: Application main page mockup



## Ranking

UserName	Points	Games	Win	Draw	Loose
Giovanni	16	8	5	1	2
Luigi	14	8	4	2	2
Mario	10	8	3	1	4
Marco	9	8	1	6	1

Go to

Figure 3: Rank page mockup

## Login/SignUp

UserName:

Password:

SignUp

Login

Figure 4: Game page mockup

### 3 Message Flow

### 4 UML Diagram