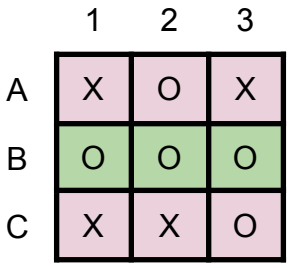
UNIVERSITÉ DE MONTPELLIER

I.U.T MONTPELLIER - SETE

Cahier des charges

Création d'un morpion en réseau



Réalisé par:

BINAUD David, FABIANI Ylona, ROURE Elie

CODE :

1xy Connexion

101 : Connexion établie
102 : Serveur introuvable
103 : Pseudo valide
104 : Pseudo non valide
105 : Déconnexion du joueur adverse
106 : Déconnexion

2xy Déroulement partie

201 : Votre tour
202 : Tour adversaire
203 : Coup valide
204 : Coup non valide
205 : MAJ grille

3xy Fin de partie

301 : Partie gagnée
302 : Partie perdue
303 : Égalité
304 : Rejouer
305 : L’adversaire veut rejouer
306 : L’adversaire ne veut pas rejouer
307 : Redémarrer partie
308 : Fin de la partie

CONTRAINTES:

Contraintes serveur :

Notre application serveur a plusieurs contraintes afin d'assurer son bon fonctionnement. Premièrement, il est limité à un certain nombre de joueurs. Nous ne pouvons pas accepter un nombre de connexions infini, cela ralentit grandement les performances de notre serveur. De plus, si un utilisateur ne fait aucune action pendant deux minutes alors la connexion de ce dernier est fermée et la victoire revient à l'autre joueur. Pour finir, une partie ne peut se lancer que s'il y a deux joueurs disponibles, ils se rencontreront automatiquement sans qu'ils n'aient à faire une action dans leur application.

Contraintes Clients :

Notre application cliente, quant à elle, doit vérifier uniquement les saisies des utilisateurs. Lors d'une partie, le serveur attend une chaîne de caractère avec un format précis. Le client doit vérifier si l'utilisateur respecte ce format.

SPÉCIFICATIONS :

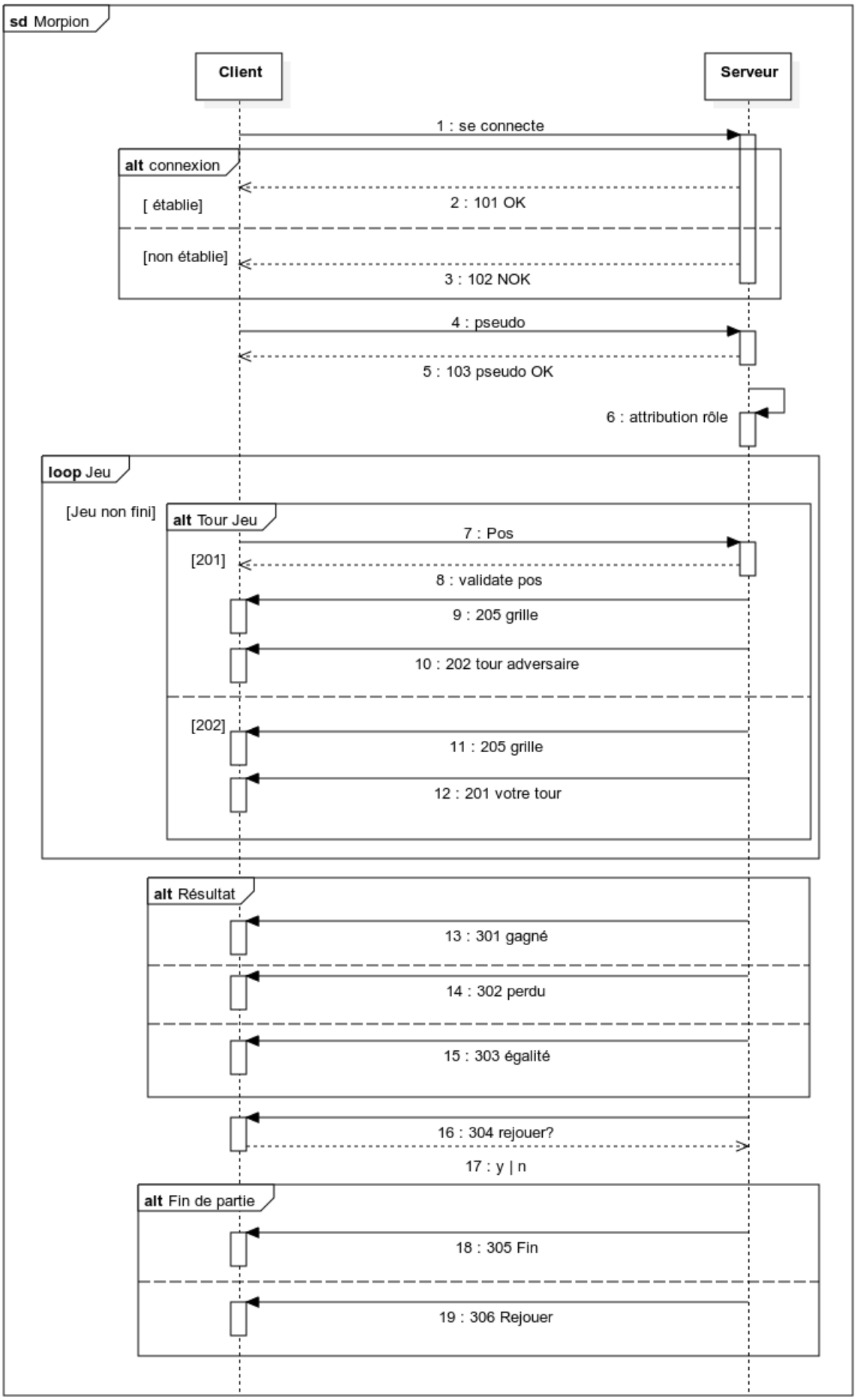
Réalisation d'un jeu en ligne de morpion en Java. Cette application est sans interface graphique et se joue uniquement dans une console. L'application se décompose en deux parties. La première, est le serveur. Il a pour rôle de permettre à l'utilisateur de jouer contre d'autres joueurs. La deuxième est l'application cliente. C'est cette partie que l'utilisateur va manipuler. C'est par le client que l'utilisateur peut renseigner le coup qu'il joue. Il voit également les coups de son adversaire.

Notre grille est une matrice de trois par trois. Elle a comme abscisse les caractères : "1", "2" et "3". Pour les ordonnées nous avons les caractères : "A", "B" et "C".

	1	2	3
A	X	O	X
B	O	O	O
C	X	X	O

Figure 1 : Représentation du plateau de jeu

La partie débute quand deux joueurs sont connectés et qu'ils ont choisi un pseudo. Les formes ainsi que le joueur qui débute sont choisis aléatoirement. Le joueur auquel est attribuée la forme "O" commence. Dès que l'attribution des formes est faite, la partie se déroule au tour par tour, joueur après joueur. Le joueur, afin d'annoncer son coup, écrit dans sa console, son placement. À la fin de la partie, qu'elle soit gagnée par un des joueurs ou bien qu'il n'y ait pas de gagnant, ils ont la possibilité s'ils le souhaitent de faire une autre partie l'un contre l'autre.


sd Morpion
Client
Serveur
1 : se connecte
alt connexion
[établie]
2 : 101 OK
[non établie]
3 : 102 NOK
4 : pseudo
5 : 103 pseudo OK
6 : attribution rôle
loop Jeu
[Jeu non fini]
alt Tour Jeu
7 : Pos
[201]
8 : validate pos
9 : 205 grille
10 : 202 tour adversaire
[202]
11 : 205 grille
12 : 201 votre tour
alt Résultat
13 : 301 gagné
14 : 302 perdu
15 : 303 égalité
16 : 304 rejouer?
17 : y | n
alt Fin de partie
18 : 305 Fin
19 : 306 Rejouer

CRYPTAGE DES DONNÉES :

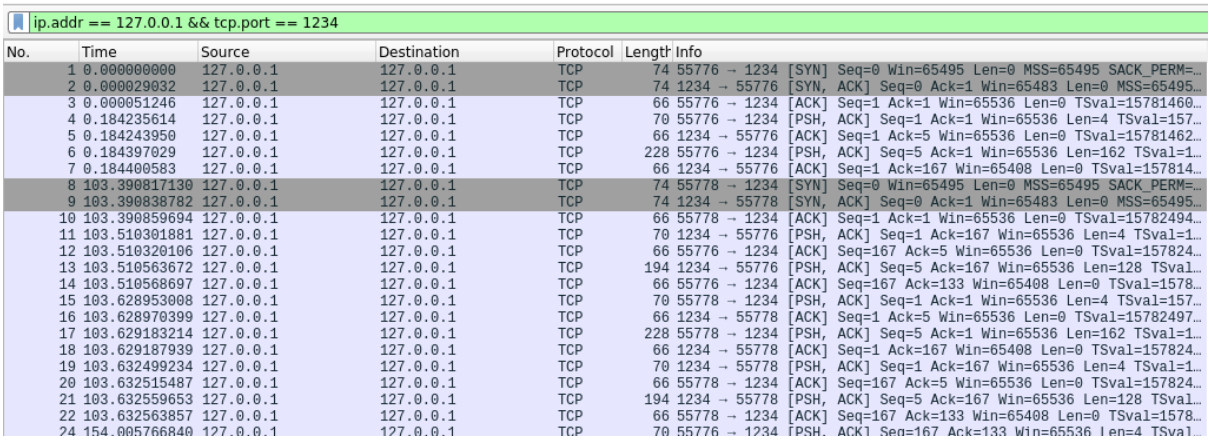
Nous avons crypté les messages échangés entre le serveur et les clients afin d'apporter plus de sécurité.

Pour cela, nous avons utilisé le système de clé DES et RSA.

Tout d'abord, le serveur va générer une clé secrète DES. Puis, il récupère les clés publiques des clients pour coder sa clé secrète, et enfin il va transmettre la clé codée aux clients.

Les clients vont alors par la suite, générer un couple de clé RSA et transmettre la clé publique au serveur. Ensuite, il va récupérer la clé codée envoyée par le serveur et la décoder avec sa clé privée.

Nous avons pu vérifier les échanges grâce à Wireshark :

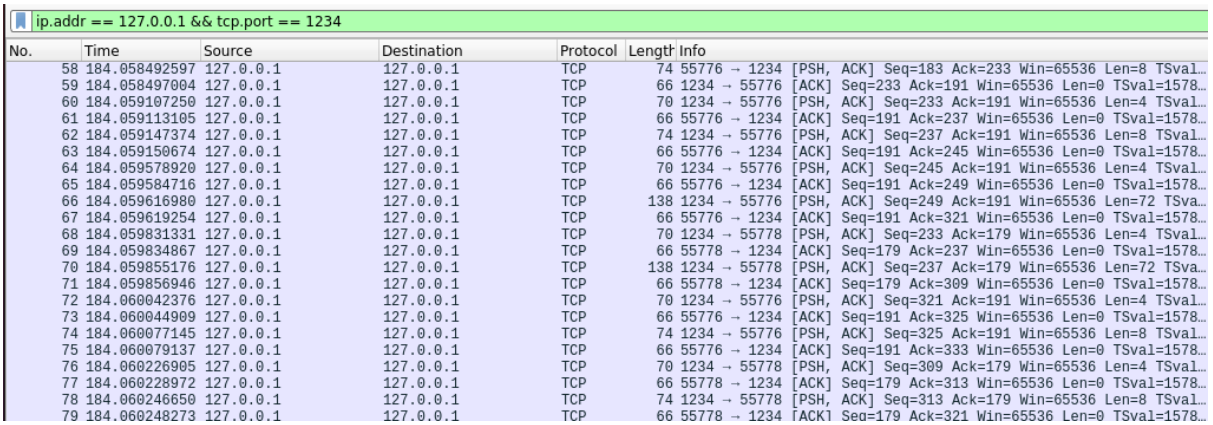


ip.addr == 127.0.0.1 && tcp.port == 1234

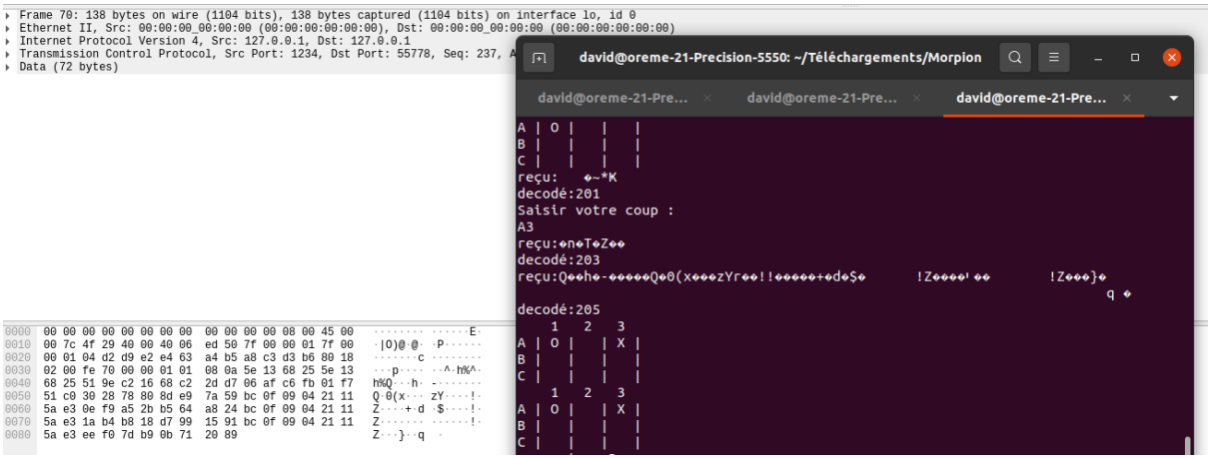
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	74	55776 → 1234 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=…
2	0.000029032	127.0.0.1	127.0.0.1	TCP	74	1234 → 55776 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495…
3	0.000051246	127.0.0.1	127.0.0.1	TCP	66	55776 → 1234 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=15781460…
4	0.184235614	127.0.0.1	127.0.0.1	TCP	70	55776 → 1234 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=4 TSval=157…
5	0.184243950	127.0.0.1	127.0.0.1	TCP	66	1234 → 55776 [ACK] Seq=1 Ack=5 Win=65536 Len=0 TSval=15781462…
6	0.184397029	127.0.0.1	127.0.0.1	TCP	228	55776 → 1234 [PSH, ACK] Seq=5 Ack=1 Win=65536 Len=162 TSval=1…
7	0.184400583	127.0.0.1	127.0.0.1	TCP	66	1234 → 55776 [ACK] Seq=1 Ack=167 Win=65408 Len=0 TSval=157814…
8	103.390817130	127.0.0.1	127.0.0.1	TCP	74	55778 → 1234 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=…
9	103.390838782	127.0.0.1	127.0.0.1	TCP	74	1234 → 55778 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495…
10	103.390859694	127.0.0.1	127.0.0.1	TCP	66	55778 → 1234 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=15782494…
11	103.510301881	127.0.0.1	127.0.0.1	TCP	70	1234 → 55776 [PSH, ACK] Seq=1 Ack=167 Win=65536 Len=4 TSval=1…
12	103.510320106	127.0.0.1	127.0.0.1	TCP	66	55776 → 1234 [ACK] Seq=167 Ack=5 Win=65536 Len=0 TSval=157824…
13	103.510563672	127.0.0.1	127.0.0.1	TCP	194	1234 → 55776 [PSH, ACK] Seq=5 Ack=167 Win=65536 Len=128 TSval…
14	103.510568697	127.0.0.1	127.0.0.1	TCP	66	55776 → 1234 [ACK] Seq=167 Ack=133 Win=65408 Len=0 TSval=1578…
15	103.628953008	127.0.0.1	127.0.0.1	TCP	70	55778 → 1234 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=4 TSval=157…
16	103.628970399	127.0.0.1	127.0.0.1	TCP	66	1234 → 55778 [ACK] Seq=1 Ack=5 Win=65536 Len=0 TSval=15782497…
17	103.629183214	127.0.0.1	127.0.0.1	TCP	228	55778 → 1234 [PSH, ACK] Seq=5 Ack=1 Win=65536 Len=162 TSval=1…
18	103.629187939	127.0.0.1	127.0.0.1	TCP	66	1234 → 55778 [ACK] Seq=1 Ack=167 Win=65408 Len=0 TSval=157824…
19	103.632499234	127.0.0.1	127.0.0.1	TCP	70	1234 → 55778 [PSH, ACK] Seq=1 Ack=167 Win=65536 Len=4 TSval=1…
20	103.632515487	127.0.0.1	127.0.0.1	TCP	66	55778 → 1234 [ACK] Seq=167 Ack=5 Win=65536 Len=0 TSval=157824…
21	103.632559653	127.0.0.1	127.0.0.1	TCP	194	1234 → 55778 [PSH, ACK] Seq=5 Ack=167 Win=65536 Len=128 TSval…
22	103.632563857	127.0.0.1	127.0.0.1	TCP	66	55778 → 1234 [ACK] Seq=167 Ack=133 Win=65408 Len=0 TSval=1578…
24	154.005766840	127.0.0.1	127.0.0.1	TCP	70	55776 → 1234 [PSH, ACK] Seq=167 Ack=133 Win=65536 Len=4 TSval…

Sur cette capture d'écran, on voit que les échanges ont bien lieux entre le serveur et les deux clients.

Lorsqu'il y a une longueur de message de 228 bytes, il s'agit de l'envoi de la clé publique RSA d'un client vers le serveur. Pour les messages de 194 bytes, il s'agit de la clé DES cryptée par la clé RSA correspondant au client, elle vient du serveur.



Sur cette image là, on voit des échanges qui vont correspondre aux choix du pseudo, et aussi à l'envoi de la grille de jeu par le serveur. Ce sont les messages de 138 bytes.



On voit ici que lors du jeu, le client reçoit une grille "dessinée" mais dans Wireshark on voit bien que la grille a été cryptée.

ARCHITECTURE :

Notre projet se décompose en plusieurs parties.

- **MonoGame :**

Permet de jouer au Morpion sans cryptage de données, et avec seulement 2 clients possibles sur le serveur.

- **MonoGameWithCrypt :**

Permet de jouer au Morpion avec les échanges cryptés. Seulement 2 clients peuvent se connecter sur le serveur.

- **MultipleGame :**

Permet de jouer au Morpion avec les échanges qui ne sont pas cryptés. Plusieurs clients peuvent se connecter à la fois.

- **MultipleGameithCrypt :**

Permet de jouer au Morpion avec des données cryptées, et en permettant à plusieurs clients de se connecter en même temps au serveur.

Dans chacun de ces packages se trouvent un package Client, permettant de lancer une application cliente. Mais aussi un package Server contenant une application pour le serveur.