COMPRESIÓN Y SEGURIDAD

TRABAJO FINAL

Cripto Pong

Grupo 12

Realizado por:
Dmitrijs Baziks
José Luis Noé Fernández
Néstor Sabater

Contenido del archivo comprimido

El archivo comprimido entregado contiene el código fuente del programa, estructurado de la siguiente manera:

cs-pong
- > bin (archivos binarios compilados)
- > src (archivos .java)
- background_pong.png (imagen de fondo)
- privkey.txt (archivo binario de clave RSA privada)
- pubkey.txt (archivo binario de clave RSA pública)

Manual de usuario

El programa está pensado para un uso sencillo e intuitivo, teniendo un total de cinco teclas para su uso.
Dichas teclas son las siguientes y se explican con ejemplos de la interfaz:

Ventana principal (menú principal)

S -> Crear partida (host del juego)

Se debe proveer un puerto para escuchar y un nombre de jugador

Campos:

- Puerto
- Nombre jugador

C -> Conectarse a partida (cliente)

Se debe proveer una IP, un puerto y un nombre de jugador

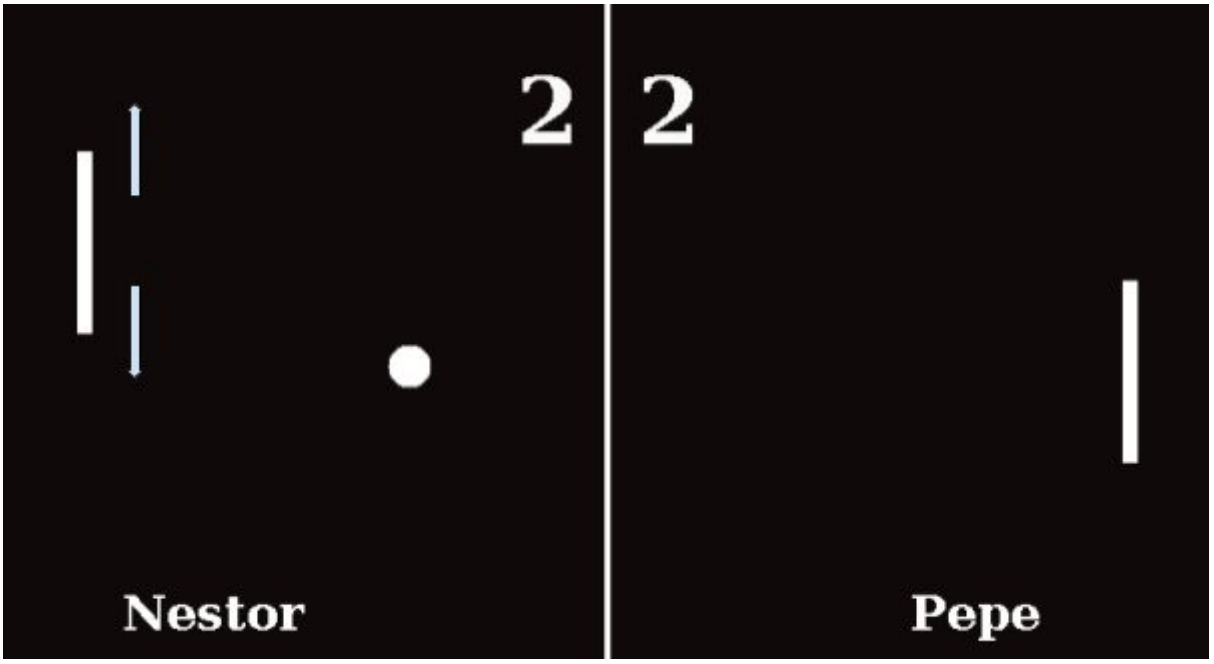
Campos:

- IP
- Puerto
- Nombre jugador

Ventana de juego

Arrow up -> Desplazar pala hacia arriba

Arrow down -> Desplazar pala hacia abajo



Teclas post partida

N/Esc -> Volver al menú
Tecla -> Nueva partida

Implementación

El juego multijugador online sigue una arquitectura cliente servidor a través de sokets.
Uno de los jugadores actua de servidor (host) en un puerto determinado por él mismo en el momento de creación de la partida, y el otro jugador se conecta, como cliente, a la IP y puerto donde se aloja el host.

Se mantiene una conexión abierta durante toda la partida para realizar actualizaciones del estado de la partida.

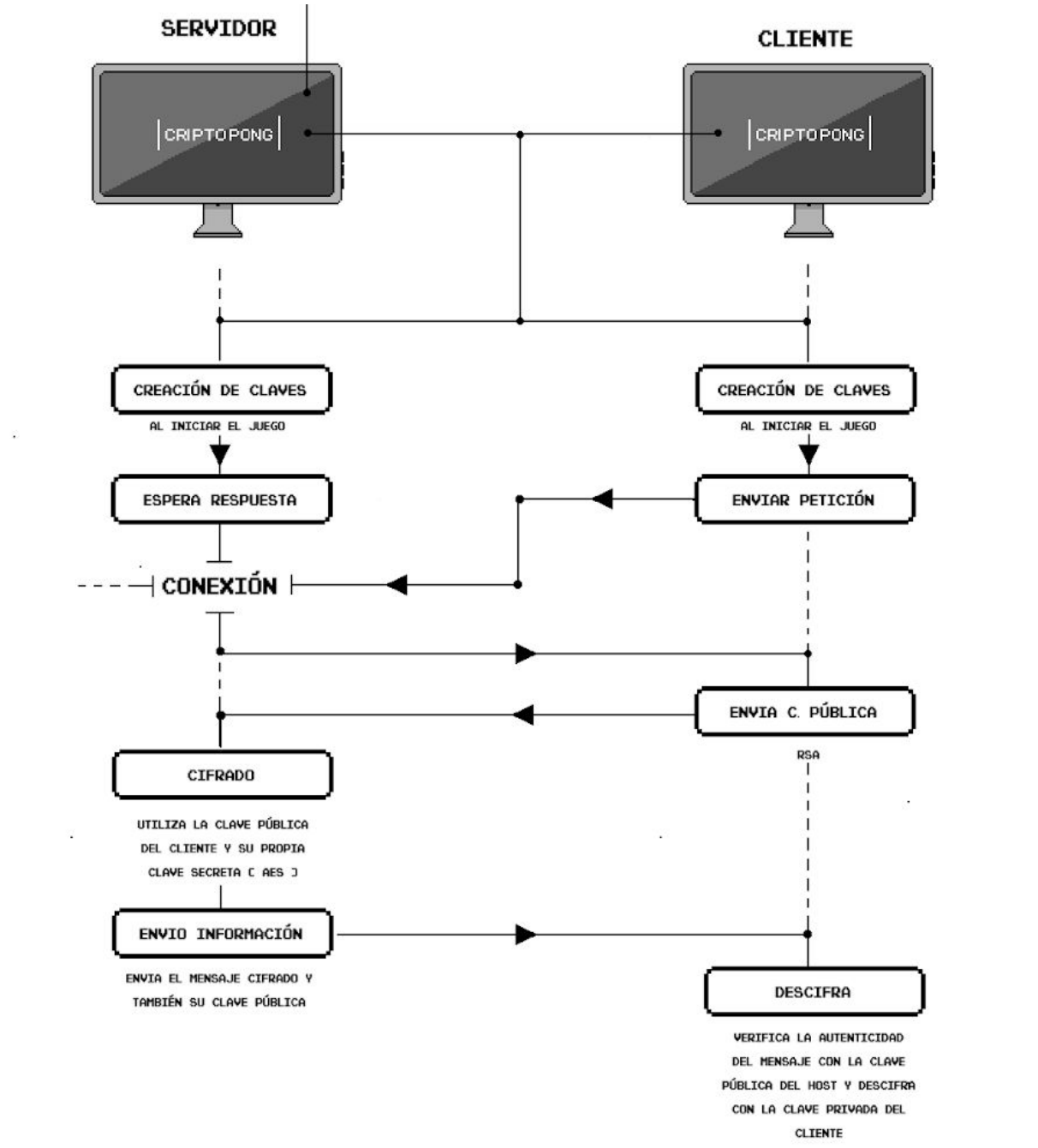
Protocolo de seguridad

Concepto

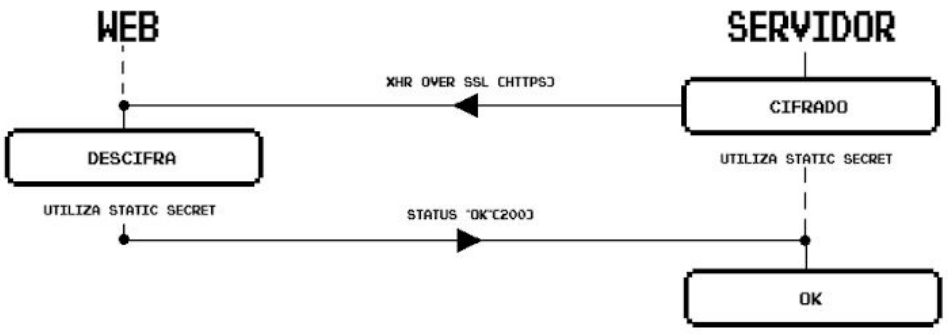
Las comunicaciones durante la partida serán cifradas mediante un algoritmo de clave secreta AES para asegurar la máxima velocidad posible en la interacción.
Para asegurar la integridad y confidencialidad de dicha clave secreta, se utilizará, en el momento de la conexión, un algoritmo RSA de clave publicoprivada donde se cifrará y firmará el contenido de la misma.

Codigo

Para la facilidad y mejora de calidad del código, se crearon dos clases.
PongSecurity -> Conteniendo todas las funciones de seguridad utilizadas, así como para almacenar los datos en memoria de todas las claves usadas en el momento.
SecurityData -> Clase contenedora de la clave secreta AES (cifrada y firmada con RSA) + la clave pública del emisor, con el fin de ser enviada por la red.


SERVIDOR
CLIENTE
CRIPTOPONG
CRIPTOPONG
CREACIÓN DE CLAVES
AL INICIAR EL JUEGO
CREACIÓN DE CLAVES
AL INICIAR EL JUEGO
ESPERA RESPUESTA
ENVIAR PETICIÓN
CONEXIÓN
ENVIA C. PÚBLICA
RSA
CIFRADO
UTILIZA LA CLAVE PÚBLICA DEL CLIENTE Y SU PROPIA CLAVE SECRETA (AES)
ENVIO INFORMACIÓN
ENVIA EL MENSAJE CIFRADO Y TAMBIÉN SU CLAVE PÚBLICA
DESCIFRA
VERIFICA LA AUTENTICIDAD DEL MENSAJE CON LA CLAVE PÚBLICA DEL HOST Y DESCIFRA CON LA CLAVE PRIVADA DEL CLIENTE

CIFRADO WEB


WEB
SERVIDOR
XHR OVER SSL (HTTPS)
CIFRADO
DESCIFRA
UTILIZA STATIC SECRET
UTILIZA STATIC SECRET
STATUS "OK"(200)
OK

En el siguiente fragmento de código podemos observar el esquema anteriormente mostrado indicando el algoritmo de seguridad para el intercambio inicial de las claves

```
// - Get client public key from client - //
ObjectInputStream getObj = new ObjectInputStream(clientSoc.getInputStream());
SecurityData sDataC = (SecurityData) getObj.readObject();
getObj = null;
System.out.println("Client security data received: " + sDataC.toString());

// - Generate new secret key for the match (AES) - //
PSecurity = new PongSecurity();
byte[] secretKey = PSecurity.getKey();
System.out.println("Secret generated: " + secretKey.toString());


// - Encript AES secret key with Client's public RSA key - //
String secretKeyEncripted = PongSecurity.encryptWithPublicKey(secretKey, sDataC.getPublic());
System.out.println("Secret encripted: " + secretKeyEncripted);

// - Sign encripted secret to confirm origin - //
SignedObject signedSecretKeyEncripted = PSecurity.signObject(secretKeyEncripted);
System.out.println("Secret encripted and signed: " + signedSecretKeyEncripted);

// - Save encrypted AES into object to send over the network - //
SecurityData sDataS = new SecurityData();
sDataS.setSecret(signedSecretKeyEncripted);
sDataS.setPublic(PSecurity.getPublic());
System.out.println("Server security data sent: " + sDataS.toString());


// - Send cyphered AES (with RSA) to client - //
ObjectOutputStream sendObj = new ObjectOutputStream(clientSoc.getOutputStream());
sendObj.writeObject(sDataS);
sendObj = null;
```

Adjuntamos el contenido relevante de PongSecurity al final del documento como referencia.


```
public class PongSecurity {
    private byte[] secret; // Secret key unEncrypted
    private String transformation = "AES";
    private PublicKey PublicKey;
    private PrivateKey PrivateKey;

    private static String PublicKeyPath = "pubkey.txt";
    private static String PrivateKeyPath = "privkey.txt";
```

```java
private void generateNewSecert(){
    KeyGenerator keyGen;
    try {
        keyGen = KeyGenerator.getInstance(transformation);
        keyGen.init(128); // for example
        SecretKey secretKey = keyGen.generateKey();
        secret = secretKey.getEncoded();
    } catch (NoSuchAlgorithmException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```