Universidade de Coimbra, Departamento de Engenharia Informática

Segurança em Tecnologias da Informação

Trabalho Prático 3

Pedro Janeiro
2012 143 629
pjaneiro@student.dei.uc.pt

Samuel Nunes
2011 158 011
snunes@student.dei.uc.pt

Introdução

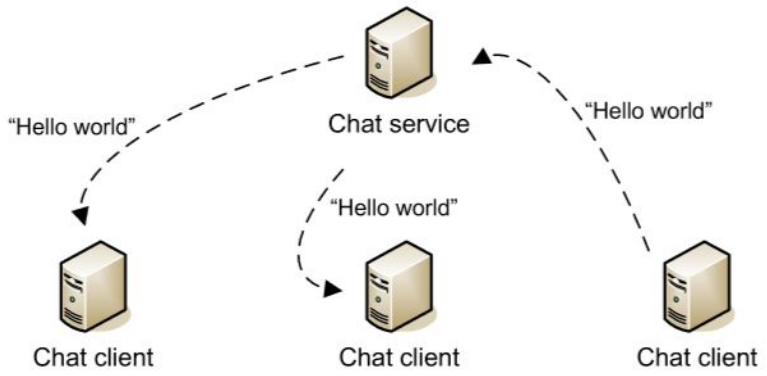
Este projeto desenvolveu-se no âmbito da cadeira de Segurança em Tecnologias da Informação do Mestrado de Engenharia de Software do Departamento de Engenharia Informática.

O objetivo deste é solidificar os conhecimentos sobre segurança e explorar as funções de encriptação do Java, tendo como base um chat que utiliza sockets para a troca de mensagens entre um servidor e vários clientes.

O nosso trabalho passa por implementar, em cima do programa fornecido, mecanismos que garantam a troca de mensagens de forma segura e automática.

Os requisitos fornecidos para este projeto são os seguintes:

- Confidencialidade
- Autenticidade
- Integridade
- Não Repúdio
- Controlo de acesso
- Controlo de chaves
- Gestão segura de Informação Confidencial



Esquema do serviço de chat (retirado do enunciado)

Funcionamento Geral

Inicialização

No cliente, o primeiro passo é a autenticação. É pedido ao utilizador para introduzir o seu username e password, permitindo ao programa encontrar o ficheiro correspondente e desencripta-lo. O ficheiro “[username].keys” está encriptado com a password pessoal e contém a chave pública e privada do utilizador.

Caso não exista um ficheiro com o nome de utilizador introduzido, é gerada uma nova chave pública e privada, sendo estas depois guardadas num novo ficheiro que irá ser encriptado com AES e com a password do utilizador.

No servidor, o primeiro passo é também a autenticação, mas apenas é pedido para introduzir uma password (o ficheiro é automaticamente reconhecido). Este ficheiro, “serverKeys”, contém a chave privada e pública do servidor, e uma lista de chaves públicas que estão proibidas de comunicar com o mesmo (*black list*).

Caso não exista um ficheiro chamado “serverKeys”, é gerada uma nova chave pública e privada, sendo estas depois guardadas num novo ficheiro que irá ser encriptado com AES e com a password do servidor.

Handshake

Partindo do princípio que as chaves privadas e públicas de ambos foram lidas/criadas com sucesso, o passo seguinte consiste na ligação ao servidor e no handshake.

Na imagem abaixo podemos perceber como o handshake é feito: inclui a troca das chaves públicas e da chave simétrica, garantindo autenticidade, integridade e não repúdio.

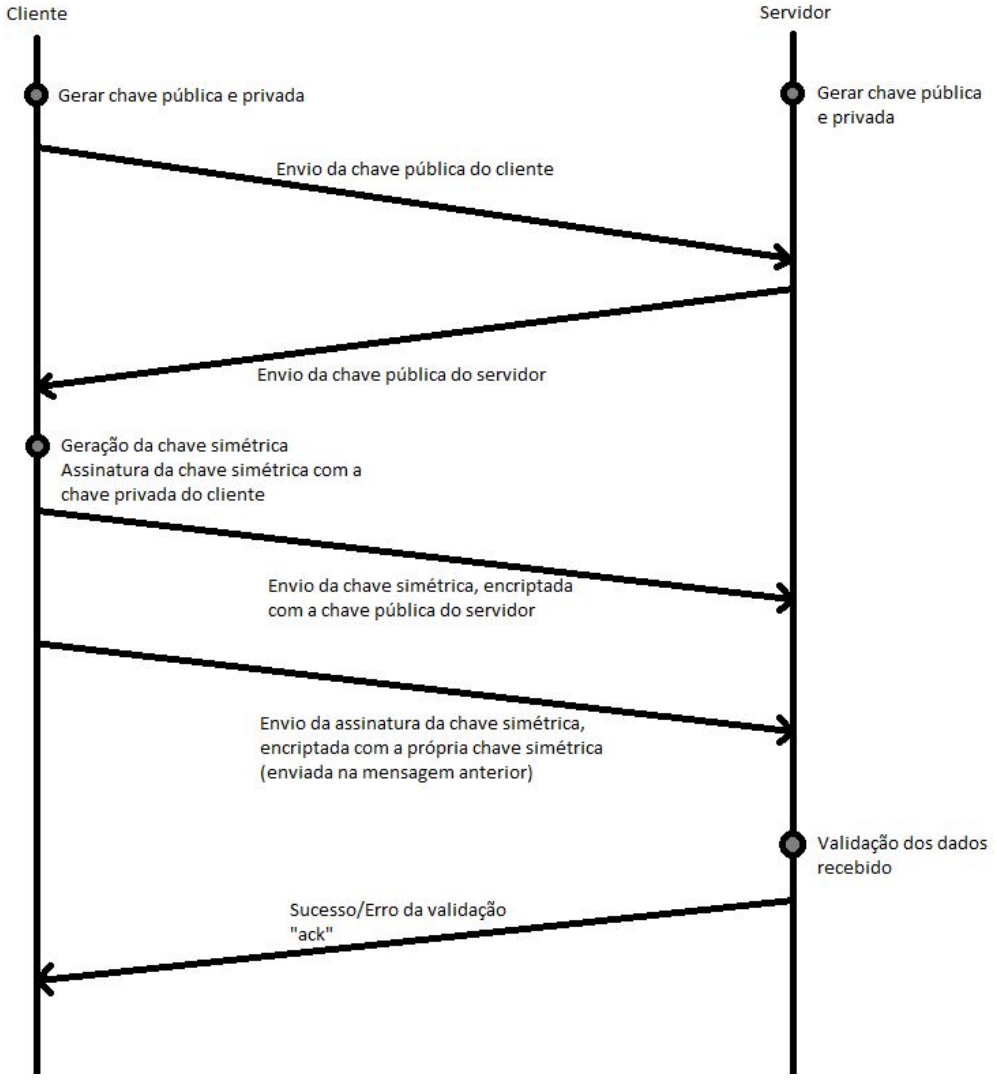
1. O cliente é o *trigger* de todo o procedimento, começando por enviar para o servidor, cujo IP e porta são conhecidos, a sua **chave pública**.

2. Após a receção, o servidor responde com a sua **chave pública**. Estas duas primeiras mensagens são enviadas sem qualquer tipo de encriptação.

3. Neste passo, o cliente vai **gerar uma chave simétrica** utilizando AES. Esta nova chave vai ser assinada com SHA-256 e com a chave privada do cliente. Por fim, a chave simétrica é **encriptada com a chave pública do servidor** e enviada. Numa segunda mensagem é enviada a assinatura, **encriptada com a chave simétrica** enviada anteriormente.

4. Quando o servidor receber a informação, vai começar por desencriptar a chave simétrica com a sua chave privada. Seguidamente, ao **validar a assinatura com a chave pública do cliente**, guarda a **chave simétrica** recebida.

5. Finalmente, o servidor envia uma mensagem de sucesso ou erro ao cliente, dependendo do resultado do processo de validação.
   Caso seja sucesso, o cliente fica assim a saber que pode enviar mensagens "normais" para o servidor. Caso seja erro, a **ligação é encerrada** após o envio do *ack*.



Handshake entre o cliente e o servidor.

Envio de mensagens

Todas as mensagens enviadas, em ambos os sentidos da ligação, são criadas da mesma maneira. É criada uma assinatura com a chave privada do remetente, e a mensagem é encriptada com a chave simétrica da correspondente ligação com o receptor.

É assim garantido que apenas o receptor desejado tem acesso ao conteúdo e que este tem a certeza da identidade do remetente.

Renovação da chave simétrica

A renovação da chave simétrica é feita automaticamente após o envio de 10 mensagens “normais”. O servidor envia uma mensagem ao cliente com o texto “.renew” e este fica assim informado de que é necessário renovar a chave simétrica.
Os passos necessários para a renovação são os 3, 4 e 5 do handshake, mantendo as mesmas garantias.

No contexto do projeto, o limite foi definido previamente, mas noutros ambientes esta variável poderia ser partilhada em conjunto com o *ack* (por exemplo).

Bloqueio de utilizadores

É possível bloquear utilizadores, impedindo-os assim de comunicar com o servidor. Para se adicionar um utilizador à *black list*, basta introduzir o seguinte comando na consola do servidor:

```
> .block [Port number]
```

Após isto, a chave pública deste utilizador é adicionada à *black list*, garantindo que este nunca mais se consegue ligar. É também enviada uma mensagem ao utilizador a informá-lo do bloqueio, seguida de uma mensagem de “.quit” (que termina a sessão do lado do cliente).

Nota: esta consola é gerida por uma nova thread criada para o efeito.

Requisitos cumpridos e testes

Garantias dos requisitos cumpridos

- **Confidencialidade** - “Confidencialidade é a propriedade de que a informação não esteja disponível a quem não tem autorização nem esteja credenciado.” [1]
    - Todas as mensagens enviadas (excepto as duas primeiras do handshake) são encriptadas ou com a chave pública ou com a chave simétrica, garantindo assim que apenas os destinatários certos podem aceder ao conteúdo.

- **Autenticidade** - “A autenticidade relaciona-se com a confirmação de autoria, a certificação e a originalidade da informação.” [1]
    - Todas as mensagens (excepto as duas primeiras do handshake) estão ou encriptadas ou assinadas, garantindo a identidade do remetente.

- **Integridade** - “Integridade é a “propriedade de que a informação não foi modificada ou destruída de maneira não autorizada ou acidental” (IN01 GSIPR, 2008).” [1]
    - Mais uma vez, todas as mensagens menos as duas primeiras do handshake estão assinadas, garantindo a integridade do conteúdo.

- **Não Repúdio** - “propriedade que garante a impossibilidade de negar a autoria em relação a uma transação anteriormente feita” [2]
    - A assinatura das mensagens é criada usando a chave privada do remetente, dando assim a oportunidade de verificar a sua identidade.

- **Controlo de acesso**
    - As propriedades do código implementado garantem que um utilizador apenas consegue aceder ao seu ficheiro utilizando o seu username e password. Desta forma, um utilizador não se pode fazer passar por outro.

- **Controlo de chaves**
    - Como referido, caso as chaves não estejam contidas no ficheiro pessoal encriptado, são geradas no início da execução do programa.
    - Como explicado, a chave simétrica é automaticamente renovada.

- **Gestão segura de Informação Confidencial**
    - Ambas as chaves, pública e privada, são armazenadas num ficheiro encriptado usando AES com a password do utilizador.

Testes

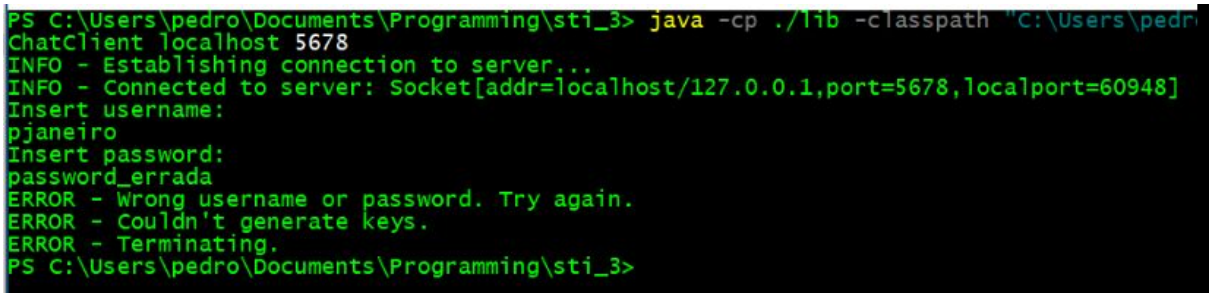
A seguinte tabela representa os testes efetuados ao chat com todas as funcionalidades de segurança implementadas.

#	Teste	Resultado Esperado	Resultado Obtido
1	Tentar abrir um ficheiro com username existente e password errada	Impossibilidade de abrir o ficheiro e aceder às chaves	Ficheiro não desencriptado e acesso às chaves negado
2	Tentar abrir um ficheiro com username existente e password certa	Desencriptação do ficheiro e acesso às chaves	Ficheiro desencriptado e chaves lidas com sucesso
3	Não encriptação das mensagens enviadas	Mensagens lidas em plain text no Wireshark	Mensagens encontradas em pacotes interceptados pelo Wireshark
4	Encriptação das mensagens enviadas	Impossibilidade de ler as mensagens através do Wireshark	Os conteúdos das mensagens não foram encontrados em plain text
5	Bloquear um utilizador	Ligação terminada	Ligação terminada por parte do servidor
6	Ligação de um utilizador bloqueado	A ligação não é estabelecida, pois a chave pública encontra-se na *black list*	A ligação com o servidor foi recusada

1. Tentar abrir um ficheiro com username existente e password errada

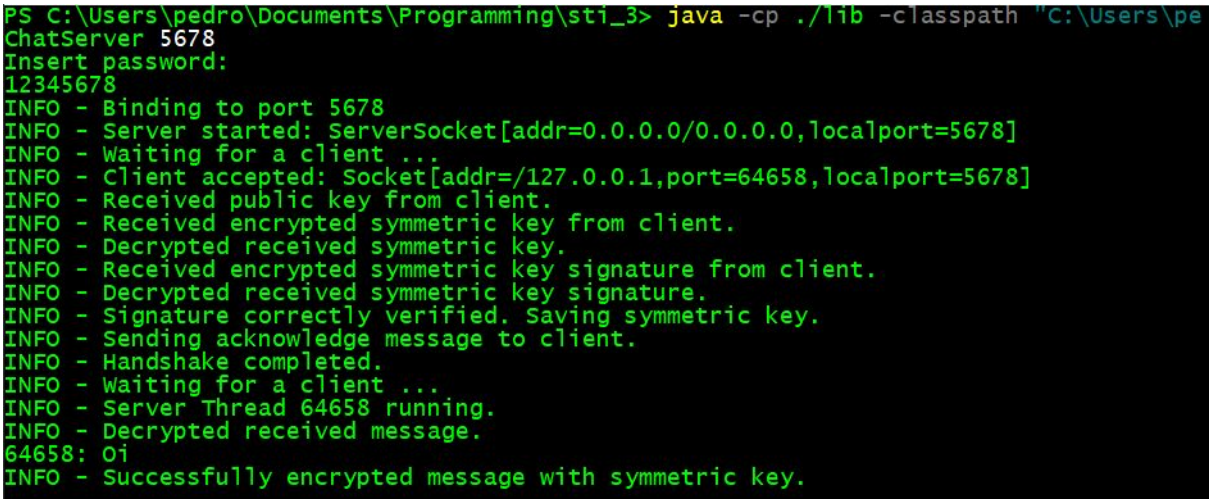
Servidor:

Cliente:

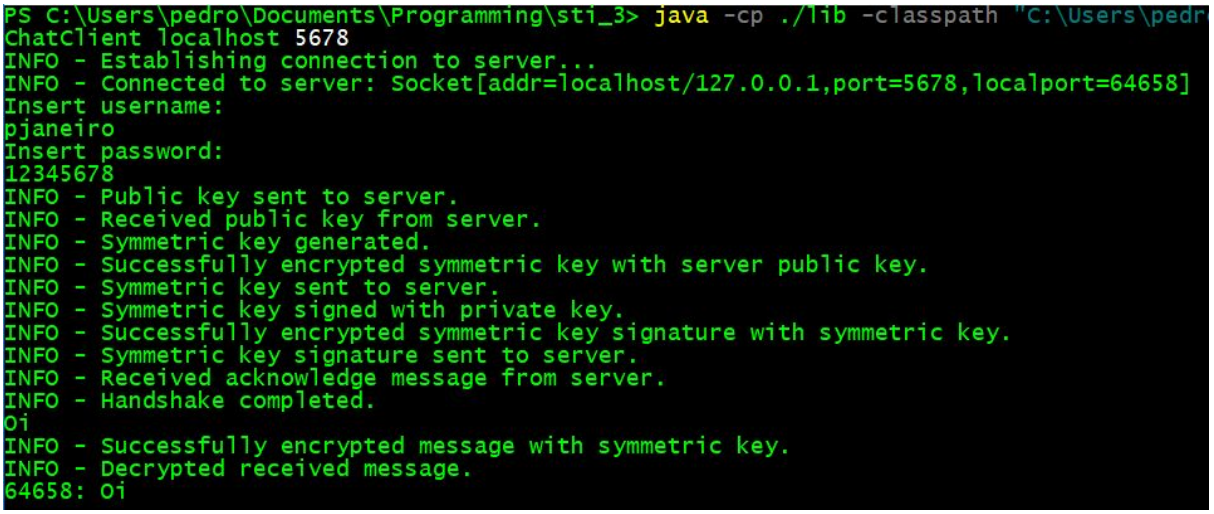


2. Tentar abrir um ficheiro com username existente e password certa

Servidor:



Cliente:



3. Não encriptação das mensagens enviadas

Servidor:

```
PS C:\Users\pedro\Documents\Programming\sti_3> java -cp ./lib -classpath "C:\Users\pe
ChatServer 5678
Insert password:
12345678
INFO - Binding to port 5678
INFO - Server started: ServerSocket[addr=0.0.0.0/0.0.0.0,localport=5678]
INFO - Waiting for a client ...
INFO - Client accepted: Socket[addr=/127.0.0.1,port=60085,localport=5678]
INFO - Waiting for a client ...
INFO - Server Thread 60085 running.
60085: Oi
60085: Tudo bem?
```

Cliente:

```
PS C:\Users\pedro\Documents\Programming\sti_3> java -cp ./lib -classpath "C:\Users\pedr
ChatClient localhost 5678
INFO - Establishing connection to server...
INFO - Connected to server: Socket[addr=localhost/127.0.0.1,port=5678,localport=60085]
Oi
60085: Oi
Tudo bem?
60085: Tudo bem?
```

Wireshark:

No.	Time	Source	Destination	Protocol	Length	Info
11	5.576029	127.0.0.1	127.0.0.1	TCP	40	5678 → 60085 [ACK] Seq=1 Ack=5 Win=525568 Len=0
12	5.576029	127.0.0.1	127.0.0.1	TCP	42	60085 → 5678 [PSH, ACK] Seq=5 Ack=1 Win=525568 Len=2
13	5.576029	127.0.0.1	127.0.0.1	TCP	40	5678 → 60085 [ACK] Seq=1 Ack=7 Win=525568 Len=0
14	5.576029	127.0.0.1	127.0.0.1	TCP	44	5678 → 60085 [PSH, ACK] Seq=1 Ack=7 Win=525568 Len=4
15	5.576029	127.0.0.1	127.0.0.1	TCP	40	60085 → 5678 [ACK] Seq=7 Ack=5 Win=525568 Len=0
16	5.576029	127.0.0.1	127.0.0.1	TCP	49	5678 → 60085 [PSH, ACK] Seq=5 Ack=7 Win=525568 Len=9
17	5.576029	127.0.0.1	127.0.0.1	TCP	40	60085 → 5678 [ACK] Seq=7 Ack=14 Win=525568 Len=0
18	8.974732	127.0.0.1	127.0.0.1	TCP	41	60085 → 5678 [PSH, ACK] Seq=7 Ack=14 Win=525568 Len=1
19	8.974732	127.0.0.1	127.0.0.1	TCP	40	5678 → 60085 [ACK] Seq=14 Ack=8 Win=525568 Len=0
20	8.974732	127.0.0.1	127.0.0.1	TCP	41	60085 → 5678 [PSH, ACK] Seq=8 Ack=14 Win=525568 Len=1
21	8.974732	127.0.0.1	127.0.0.1	TCP	40	5678 → 60085 [ACK] Seq=14 Ack=9 Win=525568 Len=0
22	8.974732	127.0.0.1	127.0.0.1	TCP	41	60085 → 5678 [PSH, ACK] Seq=9 Ack=14 Win=525568 Len=1
23	8.974732	127.0.0.1	127.0.0.1	TCP	40	5678 → 60085 [ACK] Seq=14 Ack=10 Win=525568 Len=0
24	8.975720	127.0.0.1	127.0.0.1	TCP	41	60085 → 5678 [PSH, ACK] Seq=10 Ack=14 Win=525568 Len=1
25	8.975720	127.0.0.1	127.0.0.1	TCP	40	5678 → 60085 [ACK] Seq=14 Ack=11 Win=525568 Len=0
26	8.975720	127.0.0.1	127.0.0.1	TCP	49	60085 → 5678 [PSH, ACK] Seq=11 Ack=14 Win=525568 Len=9
27	8.975720	127.0.0.1	127.0.0.1	TCP	40	5678 → 60085 [ACK] Seq=14 Ack=20 Win=525568 Len=0

```
> Frame 26: 49 bytes on wire (392 bits), 49 bytes captured (392 bits)
  Raw packet data
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 60085, Dst Port: 5678, Seq: 11, Ack: 14, Len: 9
> Data (9 bytes)
```

```
0000  45 00 00 31 41 e0 40 00  80 06 00 00 7f 00 00 01   E..1A.@. ........
0010  7f 00 00 01 ea b5 16 2e  bd ab 25 62 cc c0 d5 88   ........ ..%b....
0020  50 18 08 05 a5 cc 00 00  54 75 64 6f 20 62 65 6d   P....... Tudo bem
0030  3f                                                 ?
```

4. Encriptação das mensagens enviadas

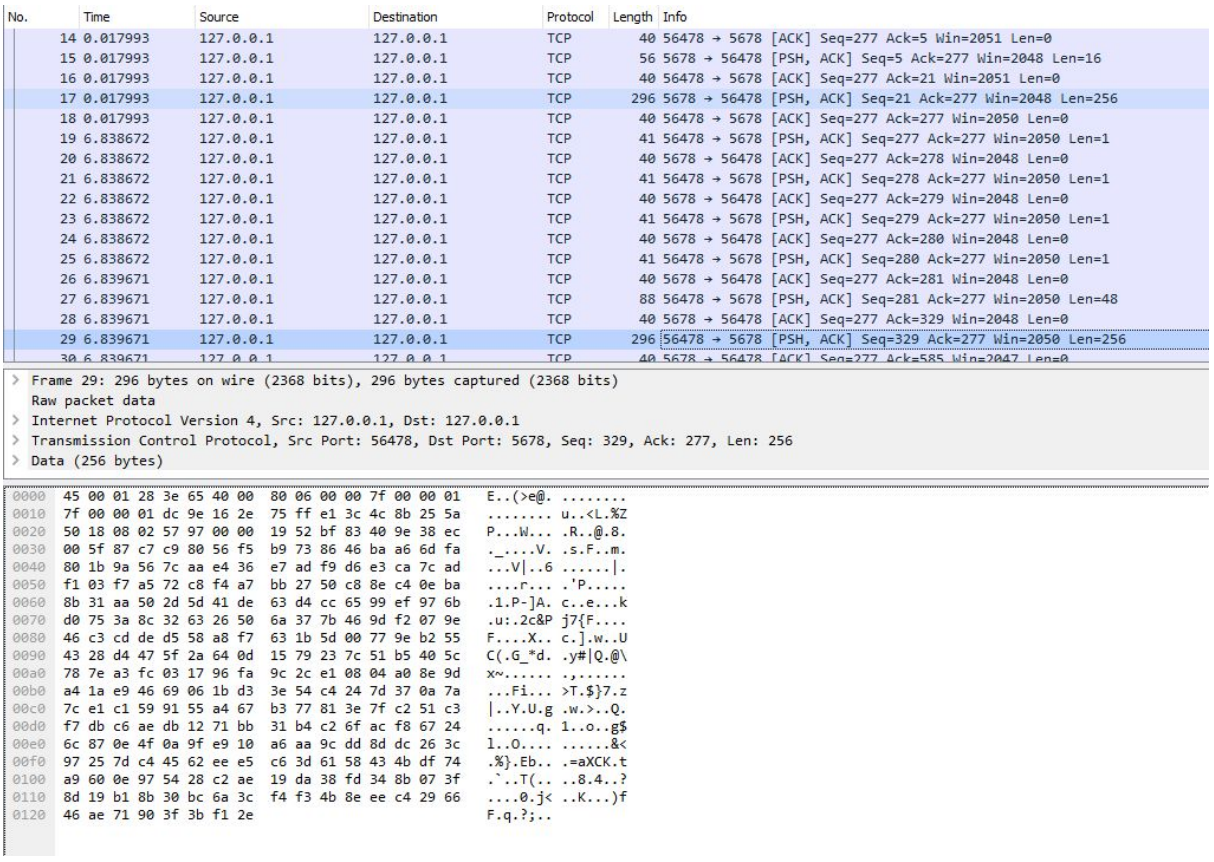
Servidor:

```
PS C:\Users\pedro\Documents\Programming\sti_3> java -cp ./lib -classpath "C:\Users\pe
ChatServer 5678
Insert password:
12345678
INFO - Binding to port 5678
INFO - Server started: ServerSocket[addr=0.0.0.0/0.0.0.0,localport=5678]
INFO - Waiting for a client ...
INFO - Client accepted: Socket[addr=/127.0.0.1,port=56478,localport=5678]
INFO - Received public key from client.
INFO - Received encrypted symmetric key from client.
INFO - Decrypted received symmetric key.
INFO - Received encrypted symmetric key signature from client.
INFO - Decrypted received symmetric key signature.
INFO - Signature correctly verified. Saving symmetric key.
INFO - Sending acknowledge message to client.
INFO - Handshake completed.
INFO - Waiting for a client ...
INFO - Server Thread 56478 running.
INFO - Decrypted received message.
56478: Oi
INFO - Successfully encrypted message with symmetric key.
INFO - Decrypted received message.
56478: Testing some security things here
INFO - Successfully encrypted message with symmetric key.
INFO - Decrypted received message.
INFO - Successfully encrypted message with symmetric key.
INFO - Removing client thread 56478 at 0
```

Cliente:

```
PS C:\Users\pedro\Documents\Programming\sti_3> java -cp ./lib -classpath "C:\Users\pedr
ChatClient localhost 5678
INFO - Establishing connection to server...
INFO - Connected to server: Socket[addr=localhost/127.0.0.1,port=5678,localport=56478]
Insert username:
pjaneiro
Insert password:
12345678
INFO - Public key sent to server.
INFO - Received public key from server.
INFO - Symmetric key generated.
INFO - Successfully encrypted symmetric key with server public key.
INFO - Symmetric key sent to server.
INFO - Symmetric key signed with private key.
INFO - Successfully encrypted symmetric key signature with symmetric key.
INFO - Symmetric key signature sent to server.
INFO - Received acknowledge message from server.
INFO - Handshake completed.
Oi
INFO - Successfully encrypted message with symmetric key.
INFO - Decrypted received message.
56478: Oi
Testing some security things here
INFO - Successfully encrypted message with symmetric key.
INFO - Decrypted received message.
56478: Testing some security things here
.quit
INFO - Successfully encrypted message with symmetric key.
INFO - Decrypted received message.
Exiting...Please press RETURN to exit ...
```

Wireshark:



No.	Time	Source	Destination	Protocol	Length	Info
14	0.017993	127.0.0.1	127.0.0.1	TCP	40	56478 → 5678 [ACK] Seq=277 Ack=5 Win=2051 Len=0
15	0.017993	127.0.0.1	127.0.0.1	TCP	56	5678 → 56478 [PSH, ACK] Seq=5 Ack=277 Win=2048 Len=16
16	0.017993	127.0.0.1	127.0.0.1	TCP	40	56478 → 5678 [ACK] Seq=277 Ack=21 Win=2051 Len=0
17	0.017993	127.0.0.1	127.0.0.1	TCP	296	5678 → 56478 [PSH, ACK] Seq=21 Ack=277 Win=2048 Len=256
18	0.017993	127.0.0.1	127.0.0.1	TCP	40	56478 → 5678 [ACK] Seq=277 Ack=277 Win=2050 Len=0
19	6.838672	127.0.0.1	127.0.0.1	TCP	41	56478 → 5678 [PSH, ACK] Seq=277 Ack=277 Win=2050 Len=1
20	6.838672	127.0.0.1	127.0.0.1	TCP	40	5678 → 56478 [ACK] Seq=277 Ack=278 Win=2048 Len=0
21	6.838672	127.0.0.1	127.0.0.1	TCP	41	56478 → 5678 [PSH, ACK] Seq=278 Ack=277 Win=2050 Len=1
22	6.838672	127.0.0.1	127.0.0.1	TCP	40	5678 → 56478 [ACK] Seq=277 Ack=279 Win=2048 Len=0
23	6.838672	127.0.0.1	127.0.0.1	TCP	41	56478 → 5678 [PSH, ACK] Seq=279 Ack=277 Win=2050 Len=1
24	6.838672	127.0.0.1	127.0.0.1	TCP	40	5678 → 56478 [ACK] Seq=277 Ack=280 Win=2048 Len=0
25	6.838672	127.0.0.1	127.0.0.1	TCP	41	56478 → 5678 [PSH, ACK] Seq=280 Ack=277 Win=2050 Len=1
26	6.839671	127.0.0.1	127.0.0.1	TCP	40	5678 → 56478 [ACK] Seq=277 Ack=281 Win=2048 Len=0
27	6.839671	127.0.0.1	127.0.0.1	TCP	88	56478 → 5678 [PSH, ACK] Seq=281 Ack=277 Win=2050 Len=48
28	6.839671	127.0.0.1	127.0.0.1	TCP	40	5678 → 56478 [ACK] Seq=277 Ack=329 Win=2048 Len=0
29	6.839671	127.0.0.1	127.0.0.1	TCP	296	56478 → 5678 [PSH, ACK] Seq=329 Ack=277 Win=2050 Len=256
30	6.839671	127.0.0.1	127.0.0.1	TCP	40	5678 → 56478 [ACK] Seq=277 Ack=585 Win=2047 Len=0

```
> Frame 29: 296 bytes on wire (2368 bits), 296 bytes captured (2368 bits)
  Raw packet data
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 56478, Dst Port: 5678, Seq: 329, Ack: 277, Len: 256
> Data (256 bytes)
```

```
0000  45 00 01 28 3e 65 40 00  80 06 00 00 7f 00 00 01   E..(>e@. ........
0010  7f 00 00 01 dc 9e 16 2e  75 ff e1 3c 4c 8b 25 5a   ........ u..<L.%Z
0020  50 18 08 02 57 97 00 00  19 52 bf 83 40 9e 38 ec   P...W... .R..@.8.
0030  00 5f 87 c7 c9 80 56 f5  b9 73 86 46 ba a6 6d fa   ._....V. .s.F..m.
0040  80 1b 9a 56 7c aa e4 36  e7 ad f9 d6 e3 ca 7c ad   ...V|..6 ......|.
0050  f1 03 f7 a5 72 c8 f4 a7  bb 27 50 c8 8e c4 0e ba   ....r... .'P.....
0060  8b 31 aa 50 2d 5d 41 de  63 d4 cc 65 99 ef 97 6b   .1.P-]A. c..e...k
0070  d0 75 3a 8c 32 63 26 50  6a 37 7b 46 9d f2 07 9e   .u:.2c&P j7{F....
0080  46 c3 cd de d5 58 a8 f7  63 1b 5d 00 77 9e b2 55   F....X.. c.].w..U
0090  43 28 d4 47 5f 2a 64 0d  15 79 23 7c 51 b5 40 5c   C(.G_*d. .y#|Q.@\
00a0  78 7e a3 fc 03 17 96 fa  9c 2c e1 08 04 a0 8e 9d   x~...... .,......
00b0  a4 1a e9 46 69 06 1b d3  3e 54 c4 24 7d 37 0a 7a   ...Fi... >T.$}7.z
00c0  7c e1 c1 59 91 55 a4 67  b3 77 81 3e 7f c2 51 c3   |..Y.U.g .w.>..Q.
00d0  f7 db c6 ae db 12 71 bb  31 b4 c2 6f ac f8 67 24   ......q. 1..o..g$
00e0  6c 87 0e 4f 0a 9f e9 10  a6 aa 9c dd 8d dc 26 3c   l..O.... ......&<
00f0  97 25 7d c4 45 62 ee e5  c6 3d 61 58 43 4b df 74   .%}.Eb.. .=aXCK.t
0100  a9 60 0e 97 54 28 c2 ae  19 da 38 fd 34 8b 07 3f   .`..T(.. ..8.4..?
0110  8d 19 b1 8b 30 bc 6a 3c  f4 f3 4b 8e ee c4 29 66   ....0.j< ..K...)f
0120  46 ae 71 90 3f 3b f1 2e                           F.q.?;..
```

5. Bloquear um utilizador

Servidor:

```
PS C:\Users\pedro\Documents\Programming\sti_3> java -cp ./lib -classpath "C:\Users\pe
ChatServer 5678
Insert password:
12345678
INFO - Binding to port 5678
INFO - Server started: ServerSocket[addr=0.0.0.0/0.0.0.0,localport=5678]
INFO - Waiting for a client ...
INFO - Client accepted: Socket[addr=/127.0.0.1,port=65179,localport=5678]
INFO - Received public key from client.
INFO - Received encrypted symmetric key from client.
INFO - Decrypted received symmetric key.
INFO - Received encrypted symmetric key signature from client.
INFO - Decrypted received symmetric key signature.
INFO - Signature correctly verified. Saving symmetric key.
INFO - Sending acknowledge message to client.
INFO - Handshake completed.
INFO - Waiting for a client ...
INFO - Server Thread 65179 running.
INFO - Decrypted received message.
65179: Oi
INFO - Successfully encrypted message with symmetric key.
.block 65179
INFO - Successfully encrypted message with symmetric key.
INFO - Successfully encrypted message with symmetric key.
INFO - Removing client thread 65179 at 0
```

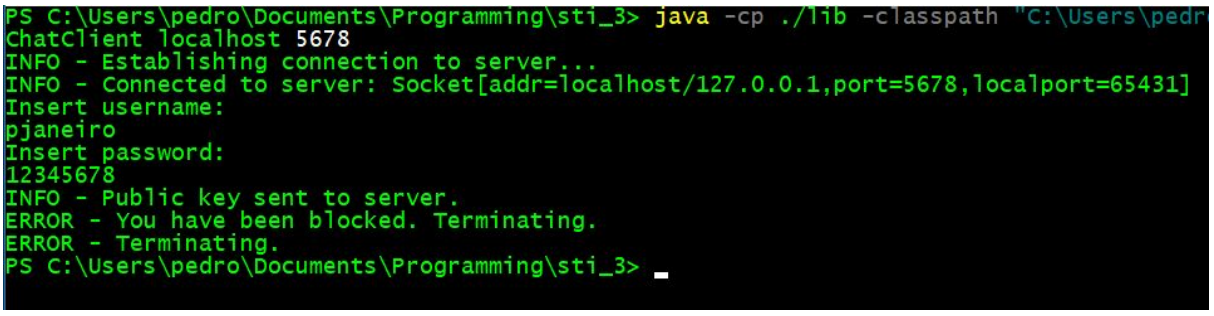
Cliente:

```
PS C:\Users\pedro\Documents\Programming\sti_3> java -cp ./lib -classpath "C:\Users\pedr
ChatClient localhost 5678
INFO - Establishing connection to server...
INFO - Connected to server: Socket[addr=localhost/127.0.0.1,port=5678,localport=65179]
Insert username:
pjaneiro
Insert password:
12345678
INFO - Public key sent to server.
INFO - Received public key from server.
INFO - Symmetric key generated.
INFO - Successfully encrypted symmetric key with server public key.
INFO - Symmetric key sent to server.
INFO - Symmetric key signed with private key.
INFO - Successfully encrypted symmetric key signature with symmetric key.
INFO - Symmetric key signature sent to server.
INFO - Received acknowledge message from server.
INFO - Handshake completed.
Oi
INFO - Successfully encrypted message with symmetric key.
INFO - Decrypted received message.
65179: Oi
INFO - Decrypted received message.
Your account has been blocked.
INFO - Decrypted received message.
Exiting...Please press RETURN to exit ...
```

6. Ligação de um utilizador bloqueado

Servidor:

Cliente:



Referências

1-“https://pt.wikipedia.org/wiki/Seguran%C3%A7a_da_Informa%C3%A7%C3%A3o_e_Comunica%C3%A7%C3%B5es”
2-“https://pt.wikipedia.org/wiki/Seguran%C3%A7a_da_informa%C3%A7%C3%A3o”
3-”http://www.java2s.com/Tutorial/Java/0490__Security/Catalog0490__Security.htm”