SecureChat

Relatório • Trabalho Prático

João Alegria | 68661

Segurança Informática e nas Organizações
LEI • 2016/2017

Troca de Mensagens Seguras

Introdução

No âmbito da Unidade Curricular Segurança Informática e nas Organizações, foi proposto aos alunos o desenvolvimento de um sistema que permitisse a troca de mensagens entre utilizadores de forma segura, sem que os conteúdos trocados entre si sejam devassados por terceiros. Essa troca de mensagens é possível recorrendo a um servidor, previamente fornecido pelos docentes da UC, onde o mesmo servirá apenas para fazer a ponte entre as mensagens enviadas por uma pessoa e quem as recebe.

A solução conta com três funcionalidades essenciais para o bom funcionamento da mesma, semelhantes a outras soluções do mercado.

É possível um dado utilizador registar-se para se tornar disponível tanto para enviar como para receber mensagens; enviar uma mensagem / conjunto de mensagens para um dado utilizador e listar os utilizadores que estejam registados num dado momento, também conhecido por "utilizadores online" em soluções idênticas.

A implementação destas três funcionalidade será descrita posteriormente na secção Implementação.

O presente relatório é constituído por quatro partes:

- Introdução
- Contextualização
- Implementação
- Referências

Contextualização

A solução foi implementada na linguagem Java, tendo sido da responsabilidade dos alunos apenas o desenvolvimento da parte cliente.

Toda e qualquer manipulação das funções do chat são possíveis segundo comandos JSON, uma vez que o servidor fornecido apenas aceita o envio de comandos JSON como método de entrada.

É de notar que a implementação da solução assenta no protocolo TCP/IP, um protocolo de comunicação usado entre máquinas em rede para encaminhamento de dados, neste caso, no envio de mensagens.

Para gerir o envio e receção de mensagens entre a parte cliente e a parte servidor, é utilizada a primitiva de transporte *socket*, garantindo assim a bidirecionalidade entre processos.

Para o fim a que este trabalho académico se destina, a implementação será executada localmente, em *localhost* com indicação do porto *1025* (uma vez que o trabalho foi realizado no sistema operativo macOS onde todos os portos até ao 1024 estão bloqueados, o que acontece, semelhantemente, em sistemas operativos Linux).

O envio e receção de mensagens, como descrito anteriormente, é através dos comandos JSON onde estes serão detentores de toda a informação.

Assim, o envio de mensagens será através de um objeto OutputStream associado a uma *socket* que será responsável de enviar a mensagem até ao servidor. O processo contrário, o de receber mensagens, é através de um objeto InputStream, associado à mesma *socket* e será responsável por distribuir a mensagens do servidor para um ou mais utilizadores.

Para que seja possível um utilizador receber mensagens provenientes do servidor de forma constante, isto é, que este não recorra a um método que retorne a última mensagem presente no servidor, é implementado uma alternativa às *threads*, denominada por espera ativa, fornecida pelos docentes.

Esta, é composta por um ciclo onde nele estarão duas instruções a avaliar repetidamente: a introdução no teclado por parte do utilizador para o servidor e a ocorrência de respostas por parte do servidor.

Deste modo, o envio de uma mensagem entre dois utilizadores está assegurado e o destinatário receberá no seu terminal a identificação do remetente, bem como o corpo da mensagem.

Implementação

Como referido anteriormente, a solução conta com três funcionalidades principais, e são três o número de comandos que o servidor aceita:

Funcionalidade	Comando
Registar Utilizador	`register`
Listar Utilizador / Utilizadores	`list`
Enviar Mensagem	`send`

Uma vez que o servidor apenas aceita comandos JSON [1] como método de entrada, toda a implementação é construída sobre pares chave-valor alfanuméricos.

`register:` O comando "register" deverá ser o primeiro a ser executado, sendo apenas possível executar este comando uma vez (após um registo válido) por cada ligação TCP do cliente ao servidor, e tem como intuito registar a identidade de um novo utilizador no sistema de troca de mensagens.

Na solução existem dois tipos de registos disponíveis:

- Um registo pelo cartão de cidadão
- Um registo manual

Aquando o registo manual de um dado utilizador, este tem de introduzir o seu nome, sendo assim gerados dois pares de chaves distintos: um par de chaves RSA e um par de chaves Diffie-Hellman; e também é gerado um número aleatório.

Na eventualidade do registo ser realizado a partir do Cartão de Cidadão, o nome do utilizador é extraído automaticamente do cartão tendo este ainda a liberdade de poder atribuir um *username* caso o seu nome seja muito extenso. É ainda obtido o certificado "CITIZEN SIGNATURE CERTIFICATE" presente no cartão para uma

completa identificação futura por parte de terceiros, adicionando também os dois pares de chaves RSA e Diffie-Hellman, bem como o número aleatório.

De notar que, dos pares de chaves gerados apenas a pública é enviada para o servidor aquando o registo.

Um exemplo de pseudo-comando de um registo utilizando o Cartão de Cidadão pode ser o seguinte:

```
{
"command":"register",
 "src":"João Tiago Faria Alegria",
 "username":"JTAlegria",
 "registoViaCC":"true",
 "randomNumber":getRandomNumber(),
 "chavePublicaRSA":getChavePublicaRSA(),
 "chavePublicaDH":getChavePublicaDH(),
 "certificadoCC":getCertificadoCC()
}
```

Exemplo de Pseudocódigo 1 - Registo Via Cartão de Cidadão

De notar também que, cada par chave-valor apenas retém valores alfanuméricos, onde os últimos quatro pares (`randomNumber`, `chavePublicaRSA`, `chavePublicaDH` e `certificadoCC`) são convertidos previamente para *String* antes do envio destes para o servidor.

Caso o servidor retorne como resultado o comando `{"error":"ok"}` o novo utilizador ficou registado com sucesso. Na eventualidade de dois utilizadores escolherem o mesmo nome, o servidor verifica a existência do nome pretendido e retorna o comando `{"error":"registered"}` que impede o registo de utilizadores com o mesmo nome, sendo possível assim a escolha de um outro nome.

`list:` O comando "list" pode ser consultado sempre que desejado e tem como propósito listar um dado utilizador ou todos os utilizadores que estejam registados até ao momento. Esta funcionalidade é conhecida pelos utilizadores de salas de chat ou outras plataformas de troca de mensagens como "Utilizadores On-Line".

Se a escolha for listar um dado utilizador, será necessário a introdução do nome da pessoa pretendida ou do *username* da mesma.

Uma vez que, a introdução de um *username* a um dado utilizador foi um *feature* adicionada, a procura por um dado utilizador tinha que estar otimizada, podendo esta ser pelo nome completo do utilizador a procurar ou pelo seu *username*. Assim, no momento do registo tanto o nome completo como o *username* são adicionados a uma estrutura de dados - *HashMap.* Deste modo, se na procura for introduzido o *username,* primeiramente será obtido o nome completo a partir da estrutura e será a partir deste que a procura irá ser realizada.

```
String nomeAProcurar = nome;

for(Map entry : nameMap.entrySet()){
  if(nome.equals(entry.getValue()){
    nomeAProcurar = entry.getKey();
  }
_____________________________________________

{
 "command":"list",
 "id":nomeAProcurar
}
```

Exemplo de Pseudocódigo 2 - Listagem de um dado Utilizador

O resultado da procura apresentará o nome completo do utilizador e as chave públicas RSA, Diffie-Hellman, a do Cartão de Cidadão (caso tenha sido um registo via Cartão de Cidadão) e o valor aleatório.

```
>> JOÃO TIAGO FARIA ALEGRIA: online
Chave Publica RSA: MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCt2ya
Chave Publica DH: MIIBpzCCARsGCSqGSIb3DQEDATCCAQwCgYEA/X9TgR11
Chave Publica CC: MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDARFRi
Valor Aleatorio: DWaVXjWziCZ6/+vs1q7MPA==
```

Figura 1 - Listagem de um dado Utilizador (username)

Caso a intenção seja a de listar todos os utilizadores, o servidor irá procurar todos os utilizadores registados até ao momento, retorna-os e apresenta as suas chaves e valor aleatório.

```
>> Outro Utilizador: online
Chave Publica RSA: MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCXUL3
Chave Publica DH: MIIBpzCCARsGCSqGSIb3DQEDATCCAQwCgYEA/X9TgR11
Valor Aleatorio: 0byToeJ9ip17Zgr3iUrWfA==

>> JOÃO TIAGO FARIA ALEGRIA: online
Chave Publica RSA: MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCt2ya
Chave Publica DH: MIIBpzCCARsGCSqGSIb3DQEDATCCAQwCgYEA/X9TgR11
Chave Publica CC: MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDARFRi
Valor Aleatorio: DWaVXjWziCZ6/+vs1q7MPA==
```

Figura 2 - Listagem de todos os Utilizadores

Na hipótese de nenhum utilizador estar registado é retornado um array vazio que representará a ausência de utilizadores registados no sistema.

send: Através do comando "send" que é possível o envio de mensagens, podendo este ser destinado a um utilizador específico, ou enviado por difusão para todos os utilizadores registados.

De notar também que é através do comando "send" que é realizado o acordo de chaves.

Quando é escolhido o envio por difusão, a mensagem não será cifrada, uma vez que a mensagem está partilhada por todos os utilizadores e não se pretende a confidencialidade da mesma.

Por sua vez, quando uma mensagem é enviada para um dado utilizador, esta tem de ser cifrada e autenticada. Para tal, foram implementadas as seguintes funcionalidades:

1) Envio de uma mensagem para um destinatário, cifrada com uma chave derivada de uma senha.

2) Envio de uma mensagem para um destinatário, cifrada com a chave pública do mesmo e usando cifra híbrida.

3) Acordo de uma chave de sessão entre dois interlocutores, e envio de mensagens usando:

(i) uma senha,

(ii) o algoritmo de Diffie-Hellman

(iii) o envio de uma chave de sessão cifrada com a chave pública do interlocutor.

1) Envio de uma mensagem para um destinatário, cifrada com uma chave derivada de uma senha

- - Lado Remetente - -

Para se proceder ao envio de uma mensagem cifrada com uma chave derivada de uma senha são realizados os seguintes passos:

Passo 1: Inserir um destinatário e verificar a existência deste no sistema. Caso não haja correspondência, é novamente questionado.

Passo 2: Inserir uma senha (conhecida por ambos os interlocutores)

Passo 3: Geração de um *salt* (valor aleatório).

Passo 4: Inserir a mensagem.

Passo 5: Derivação da senha originando uma *SecretKey*, utilizando o salt previamente gerado e a instância "PBKDF2WithHmacSHA1" [2] [3].

Passo 6: Cifrar simetricamente a mensagem através do algoritmo DES (com o reforço de segurança de cifra tripla "DESede").

Passo 7: Gerar um *MAC - Message Authentication Code* - para a concatenação da mensagem cifrada e salt, de modo a autenticar a mensagem a enviar.

Passo 8: Envio da mensagem para o servidor.

```
{
"command":"send",
 "dst":"Outro Utilizador",
 "remetente":"JTAlegria",
 "msg": {
         "msgBody":getMensagemCifrada(),
         "salt":getSalt()
        }
 "HMAC": getHMac(),
 "ID":"DESede"
}
```

Exemplo de Pseudocódigo 3 - Envio de uma Mensagem cifrada Simetricamente

- - Lado Destinatário - -

Passo 1: Inserir uma senha

Passo 2: Gerar um novo MAC - *Message Authentication Code* - com a mensagem cifrada e o salt provenientes da leitura do servidor.

Passo 3: Comparar se o novo MAC gerado é igual ao MAC recebido.
Caso seja igual a mensagem é decifrada e apresentada ao utilizador.

2) Envio de uma mensagem para um destinatário, cifrada com a chave pública do mesmo e usando cifra híbrida:

- - Lado Remetente - -

Para se proceder ao envio de uma mensagem cifrada com cifra híbrida [4] são realizados os seguintes passos:

Passo 1: Inserir um destinatário e verificar a existência deste no sistema. Caso não haja correspondência, é novamente questionado.

Passo 2: Recolher o valor em *String* da chave pública do par de chaves RSA do destinatário e convertê-lo para *PublicKey* [5] [6].

Passo 3: Inserir a mensagem.

Passo 4: Cifrar simetricamente a mensagem através do algoritmo DES.

Passo 5: Assinar a mensagem [7]:

Passo 5.1 - Caso o registo tenha sido pelo Cartão de Cidadão:
O Cartão de Cidadão é novamente lido e o utilizador que pretende assinar a mensagem terá de introduzir o código PIN do cartão referente à assinatura digital para ser assim possível assinar a mensagem com a chave privada destinada a esse fim presente no cartão.

Passo 5.2 - Caso o tenha sido manual:
É recolhido a chave privada do par de chaves RSA do remetente criado previamente no registo, e é com esta que se procederá à assinatura da mensagem.

Passo 6: Cifrar a chave (utilizada na cifra da mensagem) com a chave pública do destinatário.

Passo 7: Envio da mensagem para o servidor

```
{
"command":"send",
 "dst":"Outro Utilizador",
 "remetente":"JTAlegria",
 "msg": getMensagemCifrada(),
 "assinatura": getAssinatura(),
 "chave": getChaveCifrada()
 "ID":"cifraHibrida"
}
```

Exemplo de Pseudocódigo 4 - Envio de uma Mensagem cifrada Hibridamente

- - Lado Destinatário - -

Passo 1: Recolher a chave cifrada e decifrá-la com a chave privada do destinatário.

Passo 2: Validação da Assinatura e Decifra da Mensagem:

Passo 2.1 - Caso o registo tenha sido pelo Cartão de Cidadão:
Recolher o valor em *String* da certificado "CITIZEN SIGNATURE CERTIFICATE", convertê-lo para *Certificate* [8] e a partir deste recolher a chave pública neste presente.
Proceder com o método de verificação da assinatura e se o valor for "true", a mensagem é decifrada através do algoritmo DES com a chave recolhida no *Passo 1* e apresentada ao utilizador.

Passo 2.2 - Caso o registo tenha sido manual:
Recolher o valor em *String* da chave pública do par de chaves RSA do remetente e convertê-lo para *PublicKey.*
Proceder com o método de verificação da assinatura e se o valor for "true", a mensagem é decifrada através do algoritmo DES com a chave recolhida no *Passo 1* e apresentada ao utilizador.

3 (i) Acordo de uma chave de sessão entre dois interlocutores, e envio de mensagens, usando uma chave:

Passo 1: Inserir um destinatário e verificar a existência deste no sistema. Caso não haja correspondência, é novamente questionado.
Passo 2: Recolher o valor aleatório previamente gerado no registo do destinatário.

- - Acordo de Chave - -

- - Lado Remetente - -

Para se proceder ao estabelecimento de um acordo de chave de sessão entre dois utilizadores são realizados os seguintes passos, sendo estes possíveis de executar apenas uma vez:
Passo 3: Inserir uma senha (conhecida por ambos os interlocutores)
Passo 4: Concatenar a senha do *Passo 3* com o valor aleatório do remetente e o valor aleatório do destinatário.
Passo 5: Realizar a síntese da concatenação através do *MessageDigest*, onde apenas 8 dos bytes (64 bits) constituintes do resultado são necessários para proceder à futura cifra da mensagem (Algoritmo DES [9] - Tamanho do bloco: 64 bits, Tamanho da chave: 56 bits (+8 bits de paridade)).
Passo 6: Envio da síntese para o servidor
Passo 7: Adição do nome do destinatário, resultado da síntese e password a uma estrutura de dados - *HashMap*.

```
{
"command":"send",
 "dst":"Outro Utilizador",
 "remetente":"JTAlegria",
 "chaveSessao": getDigest(),
 "ID":"acordoChavePassword"
}
```

Exemplo de Pseudocódigo 5 - Envio de um Acordo de Chave de Sessão usando uma senha

- - Lado Destinatário - -

Passo 1: Questionar pela senha (conhecida por ambos os interlocutores)
Passo 2: Recolher o valor da síntese (realizado no *Passo 3 do lado remetente)*
Passo 3: Recolher o valor aleatório do remetente previamente gerado no registo do destinatário
Passo 4: Concatenar a senha do *Passo 1* com o valor aleatório do remetente e o valor aleatório do destinatário.
Passo 5: Realizar a síntese da concatenação através do *MessageDigest*, onde apenas 8 dos bytes (64 bits) constituintes do resultado são necessários para proceder à futura cifra da mensagem (Algoritmo DES - Tamanho do bloco: 64 bits, Tamanho da chave: 56 bits (+8 bits de paridade)).
Passo 6: Se o resultado da nova síntese for igual à síntese recebida, o nome do remetente, resultado da síntese e a senha são adicionados a uma estrutura de dados - *HashMap*.

- - Envio de Mensagens - -

- - Lado Remetente - -

Para se proceder ao envio de mensagens entre dois utilizadores é primeiramente realizada a verificação se a estrutura de dados *HashMap* contém o nome do destinatário. Se o resultado for verdadeiro, então é possível afirmar que o acordo de chaves foi estabelecido e é possível a troca de mensagens.
Passo 1: Inserir a mensagem.
Passo 2: Cifrar simetricamente a mensagem com a síntese presente no *HashMap*.
Passo 3: Gerar um MAC - Message Authentication Code - com mensagem cifrada e a senha presente no *HashMap*, de modo a autenticar a mensagem a enviar.
Passo 4: Envio da mensagem para o servidor.
Passo 5 (Opcional): Caso a mensagem inserida seja "ENCERRAR", o acordo de chaves é terminado e é removido da estrutura de dados as informações alusivas ao destinatário.

```
{
"command":"send",
 "dst":"Outro Utilizador",
 "remetente":"JTAlegria",
 "msg": getMensagemCifrada(),
 "HMAC": getHMac(),
 "ID":"envioMsgPassword"
}
```

Exemplo de Pseudocódigo 6 - Envio de uma Mensagem através do Acordo de Chave de Sessão usando uma senha

- - Lado Destinatário - -

Passo 1: Recolher a mensagem cifrada

Passo 2: Gerar um novo MAC - *Message Authentication Code* - com a mensagem cifrada e a senha presente no *HashMap*.

Passo 3: Comparar se o novo MAC gerado é igual ao MAC recebido.

Caso seja igual a mensagem é decifrada com o digest presente no *HashMap* e apresentada ao utilizador.

Caso a mensagem decifrada seja "ENCERRAR" o acordo de chaves é terminado e é removida da estrutura de dados as informações alusivas ao remetente.

3 (ii) Acordo de uma chave de sessão entre dois interlocutores, e envio de mensagens, usando o algoritmo de Diffie-Hellman

Passo 1: Inserir um destinatário e verificar a existência deste no sistema. Caso não haja correspondência, é novamente questionado.

Passo 2: Recolher a chave pública do par de chaves Diffie-Hellman [10] [11] previamente gerado no registo do destinatário.

- - Acordo de Chave - -

- - Lado Remetente - -

Para se proceder ao estabelecimento de um acordo de chave de sessão entre dois utilizadores são realizados os seguintes passos, sendo estes possíveis de executar apenas uma vez:

Passo 3: Criar de um *KeyAgreement* [13] [12] com a instância *DiffieHellman*, utilizando a chave privada DH do remetente e a chave pública DH do destinatário, recolhida no *Passo 2*.

Passo 4: Após o sucesso do *Ponto 3* é gerada uma *SecretKey* com o resultado do KeyAgreement no algoritmo "DES".

Passo 5: Enviar da *SecretKey* para o servidor.

Passo 6: Adição do nome do destinatário, e a *SecretKey* a uma estrutura de dados - *HashMap*.

```
{
"command":"send",
 "dst":"Outro Utilizador",
 "remetente":"JTAlegria",
 "chaveSessao": getSecretKey(),
 "ID":"acordoChaveDH"
}
```

Exemplo de Pseudocódigo 7 - Envio de um Acordo de Chave de Sessão usando o algoritmo de Diffie-Hellman

- - Lado Destinatário - -

Passo 1: Recolher o valor em *String* da *SecretKey* e convertê-lo para *SecretKey* [14].

Passo 2: Recolher o valor em *String* da chave pública do par de chaves Diffie-Hellman do remetente e convertê-lo para *PublicKey*.

Passo 3: Criação de um *KeyAgreement* com a instância *DiffieHellman*, utilizando a chave privada do destinatário e a chave pública do remetente, recolhida no Passo 2.

Passo 4: Após o sucesso do *Ponto 3* é gerada uma *SecretKey* com o resultado do KeyAgreement no algoritmo "DES".

Passo 5: Se a nova *SecretKey* for igual à *SecretKey* recebida, o nome do remetente e a *SecretKey* são adicionados a uma estrutura de dados - *HashMap*.

- - Envio de Mensagens - -

- - Lado Remetente - -

Para se proceder ao envio de mensagens entre dois utilizadores é primeiramente realizada a verificação se a estrutura de dados *HashMap* contém o nome do destinatário. Se o resultado for verdadeiro, então é possível afirmar que o acordo de chaves foi estabelecido e é possível a troca de mensagens.

Passo 1: Inserir a mensagem.

Passo 2: Cifrar simetricamente a mensagem com a *SecretKey* presente no *HashMap*.

Passo 3: Assinar a mensagem cifrada. É recolhido a chave privada do par de chaves DH do remetente criado previamente no registo, e é com esta que se procederá à assinatura da mensagem.

Passo 5: Envio da mensagem para o servidor.

Passo 6 (Opcional): Caso a mensagem inserida seja "ENCERRAR", o acordo de chaves é terminado e é removido da estrutura de dados as informações alusivas ao destinatário.

```
{
"command":"send",
 "dst":"Outro Utilizador",
 "remetente":"JTAlegria",
 "msg": getMensagemCifrada(),
 "assinatura": getAssinatura(),
 "ID":"envioMsgDH"
}
```

Exemplo de Pseudocódigo 8 - Envio de uma Mensagem através do algoritmo de Diffie-Hellman

- - Lado Destinatário - -

Passo 1: Recolher a mensagem cifrada

Passo 2: Recolher o valor em *String* da chave pública do par de chaves DH do remetente e convertê-lo para *PublicKey*

Passo 3: Assinar a mensagem recebida (cifrada). É recolhido a chave pública do par de chaves DH do destinatário criado previamente no registo, e é com esta que se procederá à assinatura da mensagem.

Passo 4: Proceder com o método de verificação da assinatura e se o valor for "true", a mensagem é decifrada através do algoritmo DES com a *SecretKey* presente no *HashMap*.

3 (iii) Acordo de uma chave de sessão entre dois interlocutores, e envio de mensagens, cifrada com a chave pública do interlocutor

Passo 1: Inserir um destinatário e verificar a existência deste no sistema. Caso não haja correspondência, é novamente questionado.

Passo 2: Recolher a chave pública do par de chaves RSA previamente gerado no registo do destinatário.

- - Acordo de Chave - -

- - Lado Remetente - -

Para se proceder ao estabelecimento de um acordo de chave de sessão entre dois utilizadores são realizados os seguintes passos, sendo estes possíveis de executar apenas uma vez:

Passo 3: Criação de uma *SecretKey* a partir de um *KeyGenerator* com o algoritmo DES

Passo 4: Cifrar a *SecretKey* com a chave pública do destinatário, recolhida no Passo 2.

Passo 5: Adição do nome do destinatário, e a *SecretKey* a uma estrutura de dados - *HashMap*.

```
{
"command":"send",
 "dst":"Outro Utilizador",
 "remetente":"JTAlegria",
 "chaveSessao": getSecretKey(),
 "ID":"acordoChavesPublicKey"
}
```

Exemplo de Pseudocódigo 9 - Envio de um Acordo de Chave de Sessão cifrado com a chave pública do interlocutor

- - Lado Destinatário - -

Passo 1: Recolher o valor em *String* da *SecretKey* e convertê-lo para *SecretKey.*

Passo 2: Decifrar a *SecretKey* com a chave privada do destinatário.

Passo 3: Criação de um *KeyAgreement* com a instância *DiffieHellman*, utilizando a chave privada do destinatário e a chave pública do remetente, recolhida no Passo 2.

Passo 4: Após o sucesso do *Ponto 3* é gerada uma *SecretKey* com o resultado do KeyAgreement no algoritmo "DES".

Passo 5: Adição do nome do destinatário, e a *SecretKey* a uma estrutura de dados - *HashMap*.

- - Envio de Mensagens - -

- - Lado Remetente - -

Para se proceder ao envio de mensagens entre dois utilizadores é primeiramente realizada a verificação se a estrutura de dados *HashMap* contém o nome do destinatário. Se o resultado for verdadeiro, então é possível afirmar que o acordo de chaves foi estabelecido e é possível a troca de mensagens.

Passo 1: Inserir a mensagem.

Passo 2: Cifrar simetricamente a mensagem com a *SecretKey* presente no *HashMap*.

Passo 3: Assinar a mensagem cifrada. É recolhido a chave privada do par de chaves RSA do remetente criado previamente no registo, e é com esta que se procederá à assinatura da mensagem.

Passo 5: Envio da mensagem para o servidor.

Passo 6 (Opcional): Caso a mensagem inserida a enviar seja "ENCERRAR", o acordo de chaves é terminado e é removido da estrutura de dados as informações alusivas ao destinatário.

```
{
"command":"send",
 "dst":"Outro Utilizador",
 "remetente":"JTAlegria",
 "msg": getMensagemCifrada(),
 "assinatura": getAssinatura(),
 "ID":"envioMsgPublicKey"
}
```

Exemplo de Pseudocódigo 10 - Envio de uma Mensagem cifrado com a chave pública do interlocutor

- - Lado Destinatário - -

Passo 1: Recolher a mensagem cifrada

Passo 2: Recolher o valor em *String* da chave pública do par de chaves RSA do remetente e convertê-lo para *PublicKey*

Passo 2: Validação da Assinatura e Decifra da Mensagem:

Passo 2.1 - Caso o registo tenha sido pelo Cartão de Cidadão:

Recolher o valor em *String* da certificado "CITIZEN SIGNATURE CERTIFICATE", convertê-lo para *Certificate* e a partir deste recolher a chave pública neste presente.

Proceder com o método de verificação da assinatura e se o valor for "true", a mensagem é decifrada através do algoritmo DES com a *SecretKey* presente no *HashMap.*

Passo 2.2 - Caso o registo tenha sido manual:

Recolher o valor em *String* da chave pública do par de chaves RSA do remetente e convertê-lo para *PublicKey.*

Proceder com o método de verificação da assinatura e se o valor for "true", a mensagem é decifrada através do algoritmo DES com a *SecretKey* presente no *HashMap.*

Referências

As referências consultadas para o desenvolvimento do trabalho prático foram as seguintes:

[1] - https://en.wikipedia.org/wiki/JSON
[2] - https://en.wikipedia.org/wiki/PBKDF2
[3] - https://goo.gl/rNxbly
[4] - https://goo.gl/rOGno0
[5] - https://goo.gl/je6y4t
[6] - https://goo.gl/glwYAu
[7] - https://goo.gl/Vb7Ya5
[8] - https://goo.gl/8nSTAl
[9] - https://goo.gl/5ioWRl
[10] - https://goo.gl/6yDm55
[11] - https://goo.gl/7Uswt7
[12] - https://goo.gl/SNPGm6
[13] - https://goo.gl/GjSkuz
[14] - https://goo.gl/OhLYwL