Secure Messaging Repository System

Ana Cruz, 76351
Inês Moreira, 76471

Secure Messaging Repository System

Segurança
Departamento de Electrónica, Telecomunicações e Informática
Universidade de Aveiro

Ana Cruz
76351
anapatriciacruz@ua.pt

Inês Moreira
76471
simoreira@ua.pt

5 de Fevereiro de 2018

Conteúdo

Capítulo 1

Introdução

Nos dias que correm, a utilização da internet tem vindo a aumentar exponencialmente, sendo que a utilização de serviços de mensagens online, segue o mesmo esquema. O público utiliza estes sistemas para transmitir todo o tipo de informação, em particular informação pessoal, de carácter sensível. É, portanto, muito importante preservar a privacidade e integridade desta informação, o que leva a que seja fundamental a implementação de serviços seguros que não comprometam a informação dos seus utilizadores.

No âmbito da unidade curricular de Segurança foi, então, proposto para projeto final, a construção de um sistema de troca de mensagens assíncrono capaz de garantir confidencialidade e integridade de toda a informação sensível com a qual opera.

Neste relatório vão ser descritos todos os elementos da nossa solução, passando pela sua arquitetura, em particular a sua organização e os tipos de mensagens existentes no sistema, assim como a sua implementação, em especial as decisões tomadas e funcionalidades.

Capítulo 2

Arquitetura

2.1 Plano Geral

O sistema consiste num serviço de troca de mensagens assíncrono, portanto, existem dois componentes fulcrais para o seu funcionamento: o cliente e o servidor. Os clientes não trocam mensagens diretamente uns com os outros, por sua vez, enviam mensagens para o servidor e é este quem reencaminha para o cliente de destino. No entanto, o servidor não é confiável, o que leva a que as mensagens não possam ser vistas por ele, pois estariam a ser expostas a terceiros.

Tendo em conta este plano de funcionamento, é necessário que as mensagens, antes de serem efetivamente enviadas para o servidor, sejam submetidas à cifragem dos seus conteúdos. Isto é feito de modo a que apenas o destinatário ou quem enviou a mensagem consiga aceder ao seu conteúdo.

2.2 Servidor

O servidor estabelece uma ligação TCP/IP com os clientes, permitindo a troca de mensagens entre eles, na medida em que o servidor recebe as mensagens e reenvia-as para os destinatários corretos.

Assim, o servidor mantém uma lista com as caixas de entrada, denominadas por message boxes e outra lista de utilizadores em sua posse. Cada utilizador, tem associada a seguinte informação:

```
{
    "id": <Valor atribuído pelo servidor. É um inteiro sequencial.>
    "uuid": <User universal identifier. É um valor único gerado a
             partir da aplicação de uma função de síntese ao
             certificado presente no cartão de cidadão>
    "mbox": <Localização da message box>
    "rbox": <Localização da receipt box>
```

```
    "secdata": <chave pública, hash da chave pública, assinatura da
                chave pública, checksum, certificado do cartão de
                cidadão do utilizador>
}
```

Cada utilizador tem uma caixa de entrada e uma caixa de recibos associada. Dentro destas, estão contidas as mensagens recebidas assim como uma cópia das enviadas e os seus respetivos comprovativos de receção.

2.3 Cliente

Nesta aplicação, o papel do cliente é enviar e receber mensagens. Como já referido, o cliente troca mensagens com o servidor de forma a interagir com outros clientes. Para isso, inicialmente é autenticado através do cartão de cidadão. Esta é a única forma de autenticação no sistema. Assim, o serviço deteta se já existe um utilizador associado ao cartão de cidadão e decide se é necessário criar uma caixa de entrada para este ou não. Após esta verificação, é disponibilizado um menu com as ações possíveis. As ações não são mais que pedidos ao servidor, enviados em JSON. Durante tudo isto, são tomados procedimentos para a geração de toda a segurança de modo a manter as operações seguras.

2.4 Mensagens

As features da aplicação estão diretamente relacionadas com os processos e mensagens possíveis no sistema. Assim, é necessário especificar todas as mensagens de modo a perceber o que cada uma envia para o servidor e a resposta que vai obter.

2.4.1 Connection

Neste momento, o nosso sistema não está a criar chave de sessão. A chave de sessão seria gerada a partir de Diffie Hellman mas não conseguimos que ficasse funcional. Assim, está presente no código, em comentário, e, portanto, decidimos que a mensagem associada fosse também explicada.

```
{
    "type": "connect"
    "pubk": <Chave pública Diffie Helman>
    "cert": <Certificado>
    "pkey_signed": <Chave pública assinada>
    "parameters": <Parâmetros>
}
```

O **type** define o tipo de mensagem, neste caso connect.

A **pubk** corresponde à chave pública gerada a partir de Diffie Hellman.

O **cert** é referente ao certificado extraído do cartão de cidadão do utilizador.

O **pkey_signed** contém a chave pública Diffie Hellman assinada com o cartão de cidadão.

O **parameters** definem os parêmetros usados para a geração do par de chaves Diffie Hellman.

2.4.2 Create

É a mensagem utilizada para a criação de uma caixa de entrada.

```
{
    "type": "create"
    "uuid": <identificador único universal>
    "cert": <Certificado>
    "pubk": <Chave pública RSA>
    "pkey_hash": <Hash da chave pública RSA>
    "pkey_hash_sig": <Assinatura da hash da chave pública RSA>
    "checksum": <String para a validação de respostas>
}
```

O **type** define o tipo de mensagem, neste caso create.

A **uuid** corresponde ao identificador único universal. No nosso caso, este é gerado através da aplicação de uma função de síntese ao certificado.

O **cert** é referente ao certificado extraído do cartão de cidadão do utilizador.

O **pkey** contém a chave pública RSA gerada para o utilizador.

O **pkey_hash** define o resultado da aplicação de uma função de síntese à chave pública RSA.

A **pkey_hash_sig** é a assinatura da hash da chave pública RSA com o cartão de cidadão.

O **checksum** consiste num valor para validação de respostas por parte do servidor.

2.4.3 List

O list é a mensagem responsável pelo pedido da listagem de todos os utilizadores com caixa de entrada no sistema.

```
{
    "type": "list"
    "checksum": <String para validação de respostas>
}
```

O **type** define o tipo de mensagem, neste caso list.

A **checksum** consiste num valor para validação de respostas por parte do servidor.

2.4.4 New

O new é a mensagem responsável pelo pedido da listagem de todas as novas mensagens na caixa de entrada de um utilizador.

```
{
    "type": "new"
    "id": <ID do utilizador>
    "checksum": <String para validação de respostas>
}
```

O **type** define o tipo de mensagem, neste caso new.

O **id** corresponde ao utilizador a quem pertence a caixa de entrada.

A **checksum** consiste num valor para validação de respostas por parte do servidor.

2.4.5 All

O all é a mensagem responsável pelo pedido da listagem de todas as mensagens, enviadas e recebidas, por parte de um utilizador

```
{
    "type": "all"
    "id": <ID do utilizador>
    "checksum": <String para validação de respostas>
}
```

O **type** define o tipo de mensagem, neste caso all.

O **id** corresponde ao utilizador a quem pertence a caixa de entrada.

A **checksum** consiste num valor para validação de respostas por parte do servidor.

2.4.6 Send

É a mensagem utilizada para enviar uma mensagem.

```
{
    "type": "send"
    "src": <ID do utilizador que envia a mensagem>
    "dst": <ID do destinatário>
    "msg": <Mensagem>
```

```
    "copy": <Cópia da mensagem>
    "aes_key": <Chave AES>
    "signature": <Mensagem assinada>
    "msg_digest": <Hash da mensagem>
    "checksum": <String para a validação de respostas>
}
```

O **type** define o tipo de mensagem, neste caso send.

A **src** corresponde ao ID do utilizador que está a enviar a mensagem.

O **dst** é referente ao ID do utilizador que irá receber a mensagem.

A **msg** contém a mensagem cifrada com chave AES.

A **copy** contém uma cópia da mensagem cifrada com uma chave AES conhecida apenas pelo utilizador que envia a mensagem.

A **aes_key** é a chave AES utilizada para cifrar a mensagem cifrada com a chave pública RSA do destinatário.

A **signature** corresponde à assinatura da mensagem com o cartão de cidadão.

A **msg_digest** contém a hash da mensagem.

O **checksum** consiste num valor para validação de respostas por parte do servidor.

2.4.7 Recv

É a mensagem utilizada para receber uma mensagem.

```
{
    "type": "recv"
    "id": <ID do utilizador que recebe a mensagem>
    "msg": <ID da mensagem a ler>
    "checksum": <String para a validação de respostas>
}
```

O **type** define o tipo de mensagem, neste caso recv.

A **id** corresponde ao ID do utilizador que recebe a mensagem.

O **msg** é referente ao ID da mensagem a ler.

O **checksum** consiste num valor para validação de respostas por parte do servidor.

2.4.8 Receipt

É a mensagem utilizada para enviar um comprovativo de receção após a leitura de uma mensagem.

```
{
    "type": "receipt"
```

```
    "id": <ID do utilizador que leu a mensagem>
    "msg": <ID da mensagem lida>
    "checksum": <String para a validação de respostas>
}
```

O **type** define o tipo de mensagem, neste caso receipt.

A **id** corresponde ao ID do utilizador que leu a mensagem e quer enviar o comprovativo de receção.

O **msg** é referente ao ID da mensagem a lida.

O **checksum** consiste num valor para validação de respostas por parte do servidor.

2.4.9 Status

É a mensagem utilizada para verificar o estado de receção das mensagens enviadas.

```
{
    "type": "status"
    "id": <ID do utilizador que enviou as mensagens>
    "msg": <ID da mensagem enviada>
    "checksum": <String para a validação de respostas>
}
```

O **type** define o tipo de mensagem, neste caso status.

A **id** corresponde ao ID do utilizador que enviou as mensagens.

O **msg** é referente ao ID da mensagem enviada.

O **checksum** consiste num valor para validação de respostas por parte do servidor.

2.4.10 getUserDetails

É a mensagem utilizada para aceder a certos atributos de utilizadores. Esta mensagem não é acessível ao utilizador comum, uma vez que foi criada para facilitar a obtenção de informação necessária.

```
{
    "type": "userdetails"
    "id": <ID do utilizador do qual são necessárias informações>
    "checksum": <String para a validação de respostas>
}
```

O **type** define o tipo de mensagem, neste caso userdetails.

A **id** corresponde ao ID do utilizador que são necessárias informações.

O **checksum** consiste num valor para validação de respostas por parte do servidor.

Capítulo 3

Implementação

3.1 Criptografia

3.1.1 Chave de Sessão

Diffie-Hellman

É um algoritmo de troca de chaves muito conhecido e usado. Seria útil para utilizar o segredo gerado como chave simétrica. No entanto, não concretizamos esta parte.

3.1.2 Funções de Síntese

SHA-256

No caso das funções de síntese, é utilizada a SHA-256 visto que o tamanho do output é apropriado e porque ainda não são conhecidas colisões.

3.1.3 Cifras Simétricas

AES

O algoritmo fulcral para cifragem escolhido foi o AES porque é aceite como suficientemente seguro sendo que é usado em vários serviços conhecidos. Usamos uma key de 256 bits, o que nos concede a segurança de que, até agora, não tenham existido ataques bem sucedidos a esta variante.

O modo de cifra usado é o CBC. Inicialmente, pensamos em utilizar um modo de cifra com autenticação, no entanto, achamos que não seria necessário uma vez que enviamos sempre uma assinatura nas mensagens. Assim, utilizamos o modo de cifra mais comum. O IV é random e adicionamos padding sempre que necessário.

3.1.4 Cifras Assimétricas

RSA

Apesar das boas características do algoritmo AES, este não é suficiente para garantir a confidencialidade desejada, uma vez que tem de existir o envio da chave. As chaves geradas são de 4096 bits, o que exige grande poder computacional e que leva a algum tempo a calcular.

Assim, usamos pares de chaves RSA de modo a satisfazer o conceito de cifra híbrida.

3.1.5 Assinaturas

Cartão de Cidadão

As assinaturas são feitas a partir da chave RSA contida no cartão de cidadão.

3.2 Funcionalidades

3.2.1 Chave de sessão

Caso funcionasse, uma mensagem connect seria a primeira a ser enviada assim que estabelecida a ligação com o servidor. Do lado do cliente, seriam gerados parâmetros que seriam posteriormente usados para gerar o par de chaves, pública e privada, Diffie Hellman. Posteriormente, extraía-se o certificado do cartão de cidadão do utilizador e assinava-se a chave pública Diffie Hellman com este. Com todos estes elementos obtidos, seria enviada a mensagem connect para o servidor. Este, verificava se a assinatura da chave pública estava correta, o que garante que foi de facto certo utilizador a enviar a mensagem, e a partir dos parâmetros gerava também o seu par de chaves e, consequentemente, um segredo. De seguida, enviava também para o cliente, em forma de resposta, a sua chave pública e o cliente gerava o seu segredo que deveria corresponder com aquele gerado pelo servidor. A chave de sessão, seria o segredo gerado pelo cliente e servidor. No entanto, não nos estava a ser possível validar os segredos, não sendo então possível gerar a chave de sessão.

3.2.2 Login/Registo com o cartão de cidadão

A nossa aplicação apenas funciona com o cartão de cidadão. Quando o serviço é iniciado, verifica se já existem chaves públicas RSA associadas ao utilizador com o cartão de cidadão inserido. As chaves RSA são guardadas no computador do utilizador com o serial number presente no certificado do seu cartão de cidadão, pois este número é único. Caso o utilizador não tenha essas chaves no seu computador, então, a primeira coisa a ser feita é

gerar o par de chaves RSA e guardá-las no computador do utilizador. De um modo análogo, é também gerada uma chave AES e guardada no computador. Sabemos que isto é potencialmente inseguro, mas não conseguimos pensar numa outra maneira de o fazer. Mais à frente vamos explicar o porquê desta chave.

Após a geração das chaves, precisamos de criar a caixa de entrada para esse utilizador. Assim, necessitamos dos restantes elementos que compõem a mensagem create. Deste modo, é calculado o uuid, através da função de síntese SHA-256 aplicada ao certificado presente no cartão de cidadão, isto garante que o uuid seja único.

É também necessário calcular a hash da chave pública RSA para que consigamos garantir que a chave não foi modificada na transmissão. Do mesmo modo, assinamos também esta hash com o cartão de cidadão para garantir que foi efetivamente aquele utilizador a enviar a mensagem.

Quando o servidor recebe esta mensagem, verifica a hash da chave pública e a sua assinatura, se tudo bater certo, guarda então a informação recebida e associa-lhe um id interno. Caso contrário, aborta a operação.

3.2.3 Garantia de resposta

Em todas as mensagens existe um campo checksum, esse campo é apenas o resultado da hash de um número random de 16 bits. Esse campo foi adicionado para ser possível, por parte do cliente, que garantidamente está a receber uma resposta correta por parte do utilizador.

O que acontece é que, aquando o envio de uma mensagem, esse checksum é recebido pelo servidor. Este, copia-o e envia-o novamente na sua resposta. Assim, o cliente ao receber a resposta do servidor, verifica se o checksum é aquele que esperava e valida ou não a resposta.

3.2.4 Cifragem de mensagens

Na cifragem de mensagens utilizamos uma cifra híbrida. No envio de uma mensagem, a primeira coisa a ser feita é a geração de uma chave AES. Logo depois, é necessária a chave pública RSA do destinatário, isto porque, ciframos o texto a ser enviado com a chave AES e, para enviar a chave AES em segurança, ciframo-la com a chave RSA pública do destinatário. Assim, quando o utilizador tentar ler a mensagem, decifra a chave AES com a sua chave RSA privada e depois consegue então decifrar a mensagem enviada.

3.2.5 Assinatura de mensagens enviadas e sua validação

Para além da cifragem falada anteriormente, sempre que é enviada uma mensagem, é calculada a sua hash, através da função de síntese SHA-256 e a mensagem é também assinada. A hash, serve para garantir que a mensagem

não foi alterada e a assinatura serve para garantir que quem a enviou foi o próprio utilizador.

No processo de receive, após decifrar a mensagem, é calculado o seu hash e verifica-se se bate certo com aquele enviado pelo utilizador. Por outro lado, como também é enviado o certificado, é possível também verificar se a assinatura da mensagem está correta, após, claro, esta ser decifrada.

Note-se que o facto do certificado ser enviado não é problemático, uma vez que este é público.

3.2.6 Cifragem das mensagens guardadas na caixa de recibos

Na mensagem de envio, a send, existe um campo chamado de copy. Este campo, contém uma cópia da mensagem cifrada com a chave AES única gerada aquando ao login. Assim, garantimos que a mensagem fica cifrada na caixa de recibos e que apenas quem a enviou a consegue ler. O problema é o facto dessa chave estar guardada desprotegida no computador do utilizador. No entanto, não conseguimos arranjar outro método que nos parecesse mais apropriado.

3.2.7 Envio de comprovativo de leitura de mensagem

Após ser invocado o comando de recv, a mensagem é decifrada e é verificado o seu hash e assinatura. Se tudo bater certo é, então, enviado um receipt automaticamente, uma vez que assim garantimos que o receipt só é enviado se efetivamente a mensagem for lida.

O receipt é apenas a mensagem decifrada assinada com o cartão de cidadão do destinatário.

3.2.8 Verificação de comprovativos de leitura de mensagens enviadas

A mensagem status é a responsável por mostrar ao utilizador o estado das mensagens enviadas.

Como o destinatário envia o recibo com a mensagem assinada com o seu cartão de cidadão, a única coisa a fazer é verificar se essa assinatura bate certo.

3.2.9 Prevenir que outros utilizadores leiam mensagens de outras caixas de entrada

Uma vez que as mensagens são sempre enviadas cifradas, no caso das cópias das mensagens com a cifra AES pessoal do utilizador, os outros utilizadores não tem forma de leitura destas mensagens.

3.2.10 Prevenir que um utilizador envie um comprovativo de receção de uma mensagem que não leu

Tendo em conta que a opção de enviar um comprovativo de receção não está disponível para os utilizadores, isto não é possível.

Capítulo 4

Conclusões

O projeto foi realizado quase totalmente como proposto. Conseguimos fazer a maioria das coisas, no entanto, a chave de sessão e posterior utilização da mesma e a validação de certificados não conseguimos implementar.

Em geral, consideramos que o projeto foi adequado e que deu para conseguir aprofundar conhecimentos da unidade curricular, uma vez que uma coisa é estudar a teoria e outra coisa é efetivamente implementar.

As nossas maiores dificuldades passaram pelo nível de detalhe que é necessário ter em consideração para que tudo funcione e que, de facto, esteja seguro. São assuntos delicados e exigem concentração e saber exatamente o que se está a fazer, caso contrário, facilmente erramos.

Em suma, o trabalho ficou um pouco aquém daquilo que seria as nossas expetativas, mas ainda assim sentimos que conseguimos cumprir com, pelo menos, os requisitos minímos.