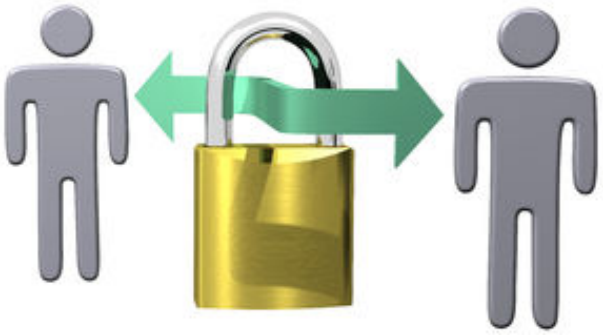
DEPARTAMENTO DE ELETRÓNICA, TELECOMUNICAÇÕES E INFORMÁTICA,
UNIVERSIDADE DE AVEIRO

Segurança

Safe Comunication



João Branquinho, 76543
Tiago Ramalho, 76718
31 de Dezembro de 2017

Professores
André Zuquete
João Paulo Barraca

CONTEÚDO

1 INTRODUÇÃO

No contexto da disciplina de Segurança, do 4º ano do Mestrado Integrado em Engenharia de Computadores e Telemática foi pedido ao grupo que desenvolvesse um sistema que permitisse a troca de mensagens assíncronas entre os clientes. O sistema só pode ser utilizado por clientes que tenham um cartão de cidadão pois é um requisito imposto na criação do utilizador e só com a sua utilização é que pode ser garantida a preservação da identidade do cliente. É possível ainda listar clientes e mensagens na caixa de entrada, enviar e ler mensagens, enviar e ler os *receipts* das mensagens trocadas. O sistema deve garantir o máximo de segurança, deste modo, deve assegurar a confidencialidade, integridade e autenticação das mensagens trocadas entre clientes e entre cliente servidor. Deve ainda ser possível que determinado cliente prove que leu o conteúdo de uma mensagem recebida. Ao longo deste relatório é descrita mais detalhadamente a implementação do grupo e a justificação de cada uma das escolhas tomadas.

2 FUNCIONALIDADES

2.1 ESTABELECIMENTO DA CHAVE DE SESSÃO ENTRE O CLIENTE E O SERVIDOR

Para existir a comunicação cliente/servidor, antes de qualquer troca de mensagens, é necessário estabelecer uma ligação segura entre os dois. O método **Diffie-Hellman (DH)** permite a geração de uma chave secreta comum entre duas entidades sem que uma terceira entidade, que possa estar a escutar a comunicação, obtenha essa mesma chave. Contudo, após alguma pesquisa o grupo decidiu utilizar o protocolo de acordo de chaves **Elliptic-curve Diffie–Hellman (ECDH)**. Para além de obter a chave secreta comum entre o cliente e o servidor o grupo acrescentou segurança pois a chave secreta não vai ser usada nem para cifrar nem para gerar o *HMAC* das mensagens enviadas. Vai apenas ser utilizada para derivar outras chaves que irão ser usadas nas funcionalidades acima mencionadas. Desta maneira as mensagens trocadas anteriormente não estão expostas a ataques de criptoanálise pois as chaves utilizadas são sempre diferentes e a partir de uma chave derivada atual é impossível chegar as chaves anteriores ou seguintes.

Na nossa implementação o cliente é quem toma a iniciativa para estabelecer a sessão assim os pacotes que são trocados são:

1. O **cliente** envia uma mensagem para o servidor a dizer que quer estabelecer uma sessão, nessa mensagem vai também um número pseudo-aleatório. Toda esta mensagem é assinada pelo cliente que, deste modo, envia a mensagem para estabelecer a sessão, a assinatura e o seu certificado.

2. O **servidor** ao receber a mensagem anterior verifica se a assinatura é válida e de seguida envia para o **cliente** a chave pública gerada por **ECDH** e o número pseudo-aleatório que recebeu. Esta mensagem do servidor vai também assinada para garantir ao cliente que a origem da mesma é do servidor e não de outra entidade.

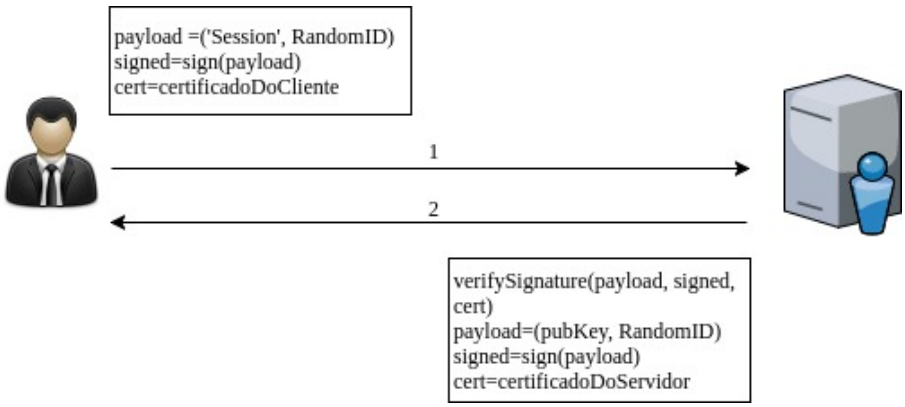


Figura 2.1: First Part ECDH

3. O **cliente** valida a assinatura do servidor e garante que o numero recebido é igual ao que enviou. Caso os dados sejam válidos o cliente usa a chave pública recebida do

servidor e gera a chave secreta de sessão. Por fim envia a sua chave pública gerada por **ECDH** para o servidor. Na mensagem vai ainda o *HMAC*. Para chegar a este autenticador é utilizada uma chave proveniente da derivação chave secreta e portanto o *salt* usado na derivação tem de ser também enviado na mensagem. No fim junta-se o *HMAC* à mensagem enviada e esta é assinada para de seguida ser enviada para o servidor.

4. Para finalizar **servidor** verifica a assinatura da mensagem, e usa a chave pública recebida para gerar a chave secreta de sessão. Ao obter esta chave o servidor vai usar o *salt* recebido para derivar a chave secreta. Gera um *HMAC* da mensagem recebida e compara com o autenticador de mensagem recebido pelo cliente. Caso sejam iguais, envia uma mensagem de *acknowledge* ao cliente. Esta mensagem vai acompanhada do seu *HMAC* e de um novo *salt,* salt este utilizado para derivar a chave secreta. Se os autenticadores de mensagem anteriormente comparados não forem iguais o servidor envia uma mensagem de erro para o cliente e o estabelecimento de sessão segura tem de ser novamente iniciado.

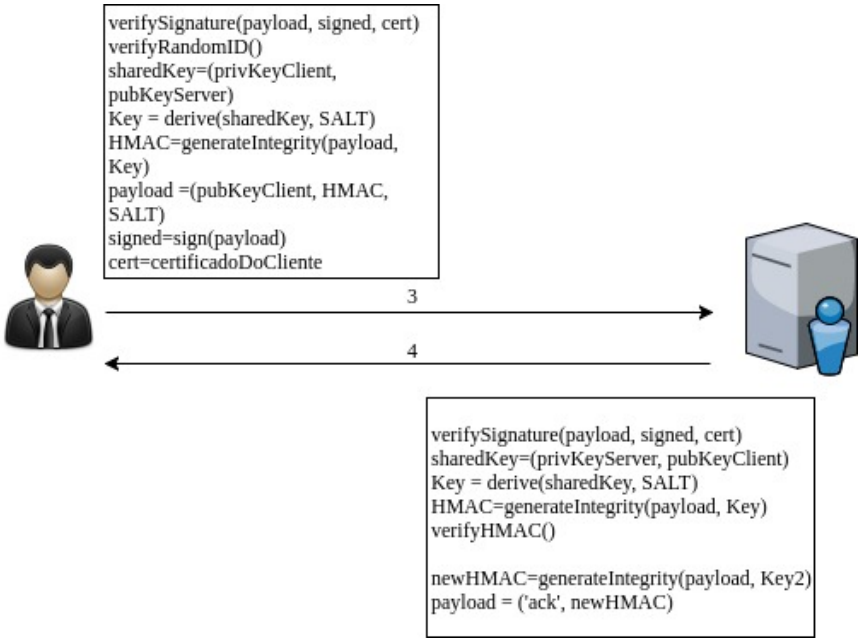


Figura 2.2: Second Part ECDH

5. O **cliente** ao receber esta ultima mensagem do servidor verifica se vem uma mensagem de erro ou um *acknowledge*. Caso seja este último o cliente deriva a chave secreta com o *salt* que vem na mensagem gera um *HMAC* da mesma e compara com o autenticador de mensagem recebido. Na hipótese de serem iguais é possivel afirmar que o cliente e o servidor estabeleceram a chave secreta e podem usa-la para comunicações futuras seguras. Se a mensagem recebida for de erro ou os *HMAC's* não forem iguais o cliente é forçado a recomeçar o processo.

2.2 Autenticação e Controlo de Integridade das Mensagens Trocadas entre o Cliente e o Servidor. Garantir que a Resposta do Servidor é Direcionada ao Pedido do Cliente

Após alguma pesquisa o grupo decidiu criar um modulo que simula uma *BlockChain* este módulo junto com a chave de sessão partilhada vai garantir a integridade das mensagens trocadas e garante ainda que a resposta do servidor é direcionada ao cliente certo. Assim, no fim de se estabelecer a chave secreta entre o cliente e o servidor, para além de partilharem a mesma chave, usam o último *HMAC* para inicializar a *blockchain*. Portanto as duas entidades ficam com *blockChain's* iguais e o *index* tem o valor um, a *previousHash* é vazio e a *currentHash* é o *HMAC*. À vista disso o procedimento existente no envio de mensagens para o servidor é:

- Gerar um *salt* e derivar a chave de sessão.
- Gerar um *HMAC* onde os dados utilizados são a mensagem a enviar, o *index* e a *currentHash*, a chave usada é a que foi obtida no ponto anterior.
- Atualizar a *blockchain* onde se soma um ao *index*, a *previousHash* passa a ser a *currentHash* e esta última fica igual ao *HMAC* gerado.
- Enviar o pacote para o servidor que tem nos seus campos o *HMAC*, o *salt* e o *payload*.

O servidor ao receber um pacote, antes de o processar, verifica a integridade precedendo da seguinte forma:

- Deriva a chave de sessão usando o *salt* que vem no pacote.
- Gera um *HMAC* onde os dados utilizados são o *payload* recebido, o *index* e a *currentHash* que tem na sua *blockchain*.
- Compara o *HMAC* gerado com o que recebeu do cliente, se forem iguais atualiza então a sua *blockchain* como descrito acima no cliente. Caso sejam diferentes envia uma mensagem ao servidor e desliga a sessão.

Neste momento a resposta do servidor segue os mesmos passos que o cliente. Através dos procedimentos descritos anteriormente é garantida a integridade da mensagem através do *HMAC* enviado e ao mesmo tempo é possível garantir que a resposta do servidor segue para o cliente certo porque, para além das *blockchain's* serem atualizadas, é sempre verificado se o autenticador enviado nas mensagens é o correto dado que depende do *HMAC* anterior (a cifra destas mensagens será descrita na secção 4.1 .

2.3 Registo do Cliente usando o Cartão de Cidadão

Na criação do cliente são armazenados vários parâmetros relevantes:

- *uuid* - identificador único do cliente (gerado do lado do cliente através do cálculo do *diggest* da sua *public key*)

- Nome do utilizador
- Chave publica para cifra de informação
- Certificado de autenticação presente no seu cartão de cidadão
- Assinatura de todos estes dados por parte do cliente.

Dados como *uuid*, o nome de utilizador ou a chave publica são importantes para a identificação e comunicação cifrada entre clientes. Por outro lado o certificado de autenticação e a assinatura de todos os dados referidos permite aos utilizadores a garantia que todos os dados presentes pertencem ao utilizador em questão e que a informação não foi adulterada, dado que o cliente os assinou. O recurso ao cartão de cidadão é obrigatório para a criação de um utilizador neste serviço de comunicação. Somente um utilizador Português consegue assinar com um certificado de autenticação presente no seu CC. O recurso ao cartão de cidadão, e às assinaturas por ele disponibilizadas, é descrito em detalhe na secção 3. Tal como é utilizado para a criação de uma nova conta, esta funcionalidade funciona da mesma forma para o *Log In*. Quando o servidor recebe esta mesma informação toma os seguintes passos:

1. Valida a assinatura;
2. Lista o uuid na base de dados e compara o certificado previamente armazenado com o certificado que acaba de receber.

Se os certificados forem os mesmos associa internamente o cliente ao respetivo *id* caso contrário retorna uma mensagem de erro. Esta associação é fulcral. Garante o Log In do utilizador para que posteriormente este possa enviar/receber mensagens ou enviar *Recipts* e receber *Status*

2.4 Cifra de Mensagens Entregues a outros Clientes

Para implementar esta funcionalidade foram utilizadas cifras mistas. No processo de criação o utilizador cria um par de chaves *RSA* de *2048 bytes* e fornece ao servidor a sua chave pública. Quando um utilizador quer enviar uma mensagem para outro utilizador pede ao servidor informação sobre o utilizador para o qual quer enviar a mensagem (faz um *List*), conseguindo assim acesso à chave publica. Depois de verificar se os dados obtidos pelo *List* são válidos, realiza uma cifra mista:

1. Cifra simetricamente a mensagem que quer enviar para o outro utilizador, utilizando para isso *AES* no modo *CTR*, com uma *KEY* de tamanho *256 bytes*;
2. Cifra, utilizando a chave publica do outro utilizador, os dados releveante para a decifra anteriormente referida (*IV*, *KEY* e *Padding*. Este padding garante a diferença das mensagens e pode, consequentemente, ser considerado lixo);
3. Envia para o servidor estas informações à qual só o dono do para de chaves *RSA* terá acesso pois só ele tem a chave privada.

2.5 Assinatura das Mensagens Entregues a outro Clientes e sua Validação

Quando o utilizador efetua um *Send* o conteúdo das mensagens são assinadas pela chave privada do cartão de cidadão. Esta mensagem faz-se sempre acompanhar pela assinatura. Assim o recetor ao obter a mensagem efetua um *List* pelo *ID* do emissor para adquirir o seu certificado. Com estas 3 informações (suposta mensagem, assinatura e certificado do emissor) pode agora:

- Validar a assinatura
- Validar da cadeia de certificação;
- Verificar a validade do certificado;
- Verificar o estado de revogação do certificado por *OSCP* ou *CRL's*;

Os 4 items referidos são cruciais para garantir a validade da mensagem e são por isso sempre efetuados no ato de receção.

2.6 Cifrar Mensagens Guardadas na Caixa de *Receipts*

A encriptação das mensagens guardadas na *receipt box* é feita seguindo o método de cifra referido na secção 1.5. A diferença é que a chave utilizada para cifrar a mensagem é a do utilizador que a enviou. Isto para que mais tarde possa verificar que enviou a mensagem.

2.7 Enviar um *Receipt* Seguro Depois de Ler a Mensagem

Depois da leitura de uma mensagem o utilizador envia automaticamente um *Receipt*(de acordo com a nossa implementação). Para efeitos de autenticação nesse *Receipt* é enviado uma assinatura da mensagem lida, garantindo que foi especificamente aquele utilizador a ler a mesma. É também possível enviar um *Receipt* manualmente para efeitos de teste.

Para uma garantia extra de que não é armazenado um *Receipt* de alguém mal intencionado que só quer poluir as caixas de receção com lixo, o servidor verifica que o *client id* que está a receber é igual ao que armazenou para aquele cliente (mais sobre isto na secção 1.3).

2.8 Verificar *Receipts* de Mensagens Enviadas

Quando um utilizador quer verificar se outro recebeu a sua mensagem (*função Status*) percorre a seguinte lista de passos:

1. Pede ao servidor informação sobre os *Receipts* das mensagens que enviou para outros utilizadores;
2. Executa a função *List* para receber o certificado do cliente que lhe enviou o *receipt*;
3. Pede ao servidor a mensagem que enviou (cifrada com a sua chave publica);
4. Decifra a mensagem que enviou;
5. Utilizando a mensagem, assinatura presente no *Receipt* e o certificado do emissor do *Receipt* o utilizador(recetor) pode confirmar a veracidade do *Receipt*.

2.9 Verificação Adequada do Certificado de Chave Pública do Cartão do Cidadão

Para garantir que uma assinatura é válida é importante garantir que o Certificado de Chave Publica que a acompanha é também ele válido. Como é possível ler na secção 2.1 a cadeia de certificação é validada, partindo de certificados considerados de confiança (Certificados emitidos pela EC do estado e um auto assinado emitido por empresas especializadas e de confiança), e o estado de revogação dos certificados é validado de acordo com dois métodos: *OCSP* e *CRL's*. É também importante referir que sempre que um utilizador corre o programa *Client* todas as *CRL's* são atualizadas para que se mantenham atuais.

2.10 Impedir que o utilizador leia mensagens de uma caixa de entrada que não a sua **e** Impedir que o utilizador envie Receipts em nome de outrem

Estas funcionalidades são explanadas ao pormenor no final da secção 1.3. Quando um utilizador efetua o login fica associada à sessão o seu id(do lado do servidor). Assim, para além da validação de assinaturas na camada cliente-cliente, o servidor impede que sejam listadas/enviadas informações de/para um *ID* que não o seu quando não o deve fazer.

3 Cartão de Cidadão e validação de certificados e Assinaturas

Nesta secção são apresentadas várias funcionalidades disponibilizadas por três módulos desenvolvidos pelo grupo de forma a interagir com o CC e a verificar a validade de certificados e validar assinaturas. São eles *Certificate*, *Cert_Sign CC_Interaction*:

3.1 Certificate

O modulo *Certificate* tem como objetivo a validação de cadeias de certificação bem com a extração de informação de cada um dos certificados. Este modulo permite, entre outras, as seguintes funcionalidades:

- Leitura de informações básicas de cada certificado: Emissor, sujeito dono do certificado, extração de chave publica, extração de links *OCSP*, *CRL* e *CRL Delta*;
- Obtenção e validação da cadeia de certificação tendo por base a confiança na EC do estado e certificados raiz. Para validação de certificados do servidor é utilizado um certificado raiz auto assinado gerado por nós.
- Teste de revogação de cada um dos certificado presentes na cadeia por *OCSP* e *CRL's*(bem como o download destas) consoante a existência de ligação à internet.
- Validação de assinaturas com base no texto original e no certificado.

3.2 CERT_SIGN

Este modulo é muito simples e serve apenas para assinar um determinado texto com base numa chave primária. Tem ainda uma função que facilita a criação um pacote json complementando a assinatura de um determinado texto, o certificado correspondente e a mensagem original. O *Cert_Sign* é utilizado numa situação muito especifica: assinatura do lado do servidor, mais especificamente no estabelecimento da sessão (secção 1.1).

3.3 CC_INTERACTION

Por fim o modulo *CC_Interaction* agiliza a interação com o cartão de cidadão. Para além de, tal como o anterior, permitir assinar, este modulo permite obter o certificado de autenticação presente no CC. Tem ainda algumas *utility functions*:

- Para obter a chave publica do certificado de autenticação;
- Um hash desta chave publica.

4 MEDIDAS EXTRA DE SEGURANÇA

4.1 MENSAGENS CIFRADAS

Como funcionalidade extra o grupo decidiu cifrar as mensagens trocadas com o servidor. Assim, quando uma mensagem é trocada, para além de existir uma derivação da chave secreta para ser usada na geração do *HMAC*, existe ainda outra derivação. Essa nova derivação é utilizada para cifrar as mensagens trocadas. Deste modo todas as mensagens trocadas com o servidor, já com a sessão estabelecida, vão ter o seguinte procedimento:

- Gerar um *SALT* e derivar a chave de sessão (esta nova chave vai ser usada na cifra);
- Gerar um *IV* e cifrar o *payload* a enviar;
- Gerar um novo *SALT* e derivar a chave de sessão (esta nova chave vai ser usada para obter o *HMAC* da mensagem);
- Gerar um *HMAC* onde os dados são: a mensagem cifrada, o *IV* e o *SALT* usado na primeira derivação;
- Atualizar a *blockChain* com o novo *HMAC*;
- Enviar a mensagem cifrada, o *IV*, os dois *SALT's* e o *HMAC*;

  Os pontos acima descritos ocorrem no envio da mensagem. Na receção verifica-se a integridade da mensagem por isso usa-se o *SALT* para derivar a chave secreta e verifica-se se o *HMAC* gerado com o *payload* recebido é igual ao autenticador de mensagens recebido. Se forem iguais a entidade (cliente ou servidor) que recebeu a mensagem usa o outro *SALT* para derivar a chave de sessão e com essa nova chave e o *IV* recebido procede à decifra de mensagem.

4.2 Chave de Sessão Efémeras

Nesta subsecção consideramos importante realçar e explicar o uso das diversas chaves usadas na comunicação com o servidor. As chaves usadas para estabelecer a chave secreta de sessão são geradas a partir da mesma curva elíptica que as do servidor e para cada sessão do cliente as chaves (do cliente e do servidor) são sempre diferentes. Garantido assim que todas as sessões sejam independentes uma das outras sem que qualquer uma comprometa uma determinada sessão. Para além da segurança entre sessões, na implementação do grupo, as mensagens trocadas durante a mesma também nunca estão em perigo pois para cada mensagem deriva-se duas novas chaves (chave para cifra e chave para produzir o *HMAC*) nunca repetindo essas chaves em mensagens futuras. Com os pontos descritos nos dois parágrafos anteriores o grupo garante que se determinada sessão for comprometida as outras sessões não o estarão e, para além disso, dependendo do problema existente nem todas as mensagens trocadas poderão estar comprometidas.

4.3 Cifra Simétrica da chave privada de cifra

Para qualquer utilizador novo é criada um novo par de chaves para cifra de mensagens enviadas. Esse par de chaves é armazenado no computador do dono do par de chaves. Qualquer atacante que tenha acesso ao dispositivo pode clonar esse par de chaves e utiliza-lo para decifrar mensagens que são dirigidas ao cliente dono do par de chaves. Para impedir que isto possa ocorra a chave privada é cifrada utilizando uma cifra simétrica de AES de 256 bytes e uma *Passphrase* fornecida pelo utilizador. O grupo assumiu que o utilizador é experiente e deve ser responsável pelo seu par de chaves, pelo que, se o utilizador não inserir *Passphrase* a chave privada não é cifrada.

5 Instalação de dependencias

Para facilitar a execução do projeto foi incluído um *script* de instalação de dependências num ambiente virtual de *python*. Isto impede que o sistema operativo do utilizador fique cheio com dependências que ele depois não vai utilizar. Assim estas dependências só existem dentro do ambiente virtual e são removidas quando este é removido. Para executar o *script* basta correr as seguintes linhas de código

```
sudo apt-get install virtualenv  # Install virtualenv
                                 # Redundant if already installed

cd project/folder/

# This script is created to install venv and its dependencies
./create_venv.sh

```

```
# Ativar ambiente virtual de python com todas as bibliotecas do
    trabalho
source venv/bin/activate
```

Depois disto podem ser corridos todos os *scripts python* do projeto. Para sair do ambiente virtual pode ser usado

```
# Exit venv
deactivate
```