Universidade de Aveiro 2017
Departamento de Eletrónica, Telecomunicações e Informática
47232 - Segurança

SEGURANÇA

Practical Project

Secure Messaging Repository System

Grupo : P4G2

Fábio Cunha - 76677
João Amaral - 76460

INDICE

Introdução **3**

Estratégias implementadas **4**

Chave de sessão 4

Criptografia assimétrica 6

Criptografia simétrica 8

Certificados de chave pública 8

Cartão de Cidadão 9

Cache de chave pública 10

Funcionalidades implementadas **10**

Mensagens 11

Mensagem Diffie-Hellman (“dh”) 11

Mensagem CREATE 12

Mensagem de aquisição de chave pública (“recipient_pk”) 13

Mensagem SEND 13

Mensagem RECV 13

Mensagem RECEIPT 14

Mensagem ALL 14

Mensagem LIST 14

Mensagem NEW 14

Mensagem STATUS 15

Arquitetura **15**

Funcionamento da aplicação **15**

Pré-requisitos da aplicação **18**

Problemas da aplicação **19**

Trabalho futuro **20**

Validação de cadeia de certificação dos certificados de chave pública 20

Tradução de id internos por nicknames ou email 20

Conclusões **20**

Referencias **21**

1. Introdução

Este relatório diz respeito ao projeto prático desenvolvido no âmbito da Unidade Curricular de Segurança lecionada no 1º semestre do 4º ano do curso Mestrado Integrado em Engenharia de Computadores e Telemática (MIECT) da Universidade de Aveiro. A àrea da segurança no contexto da informática é bastante vasta, possuindo diversos protocolos e algoritmos, alguns destes já descontinuados como é o caso do SHA1 ou MD5, e outros ainda bastante presentes nos sistemas atuais como, por exemplo, o RSA aliado a um Public Key Cryptography Standard (PKCS) de confiança ou AES (Advanced Encryption Standard). É também uma área em constante mudança uma vez que todos os dias são descobertas novas vulnerabilidades nas demais aplicações e sistemas informáticos, o que por sua vez implica a avaliação e correção das mesmas para evitar possíveis ataques que destas possam vir a surgir. Assim sendo é necessário definir estratégias criptográficas bem estabelecidas e que não comprometam a integridade do sistema no futuro. Relativamente ainda às vulnerabilidades o maior risco surge a partir de uma que seja desconhecida aos "developers" de determinada aplicação. Esta ocorrência é conhecido como "Zero Day Vulnerability", uma vulnerabilidade que se for explorada com intenções maliciosas por um atacante, ou pior, se for divulgada abertamente ao público, pode conduzir a um "Zero Day Attack", um ataque impossível de prever ou evitar com consequências graves, o que se pretende evitar a todo o custo.

Neste relatório será descrita em detalhe a aplicação desenvolvida, as suas funcionalidades, assim como as estratégias usadas para cumpir o principal objetivo do trabalho: a criação de uma aplicação de troca de simples mensagens entre clientes e entre clientes e um servidor central com o qual interagem todos os utilizadores. Um dos objetivos propostos do trabalho foi ainda a utilização do Cartão do Cidadão na aplicação, de modo a autenticar utilizadores e realizar assinaturas digitais, fazendo uso das capacidades criptográficas presentes no mesmo.

Ao realizar este trabalho tivemos como principal objetivo o desenvolvimento e aplicação de estratégias criptográficas seguras e incorporar as mesmas na nossa aplicação, de modo a garantir a integridade e confidencialidade das mensagens trocadas, a confirmação da receção de uma mensagem e ainda a preservação da identidade de um cliente (com o auxilio do Cartão do Cidadão) . Por estratégias seguras entende-se algoritmos criptográficos não depreciados e usados vastamente nas aplicações atualmente, assim como algumas das ferramentas criptográficas oferecidas pelo Cartão de Cidadão português.

2. Estratégias implementadas

a. Chave de sessão

Uma vez que todas as mensagens enviadas pelos utilizadores têm como destino direta (mensagens cliente-servidor) ou indiretamente (mensagens cliente-cliente) o servidor central era necessário a implementação de métodos que permitissem a troca segura de mensagens entre estas duas entidades. De modo a cumprir tal objetivo, foi implementada uma estratégia de chave de sessão, chave esta que é acordada entre um cliente e o servidor. Esta chave é uma chave simétrica e é obtida usando o algoritmo criptográfico de Diffie-Hellman. Este algoritmo tem como objetivo a geração de uma chave entre duas entidades, de tal modo que apenas estas duas tenham conhecimento do seu valor. O algoritmo segue, de modo simplificado, os seguintes passos:

1. Uma das entidades, geralmente a que inicia a comunicação, gera um valor de Diffie-Hellman público e outro valor privado.
2. O valor público é enviado para a outra entidade com a qual se pretende gerar a chave partilhada
3. Ao receber esse valor público, a entidade receptora gera então também um par de valores de Diffie-Hellman, e usando uma operação complexa de exponenciação modular, gera uma chave usando o valor público recebido juntamente com o seu valor privado.
4. De seguida esta última envia o seu valor público, acabado de gerar, para a entidade que enviou a primeira mensagem, que irá então usar esse valor em conjunto com o seu privado e obter uma chave semelhante à do seu par.

É assim possível obter uma chave partilhada por duas entidades, de modo que apenas estas duas possuem conhecimento do seu valor, assumindo que ambos os valores privados não são comprometidos. Esta chave permite a aplicação de algoritmos criptográficos às mensagens trocadas entre o cliente e o servidor enquanto a sessão esteja aberta, isto é, até que o cliente desligue a aplicação, ou vice-versa no caso do servidor. Quando a sessão é dada como terminada esta chave nunca mais é usada e assume-se assim que todo o conteúdo trocado fazendo uso da mesma foi trocado de forma segura e como a chave é esquecida podemos garantir que o conteúdo trocado no passado não pode ser comprometido no futuro (um requisito do princípio “Perfect forward secrecy”). No entanto se a chave de sessão for comprometida ou se alguém se fizer passar por outra entidade (conhecido por “Man-in-the-middle attack”) que não a mesma, estas garantias podem não ser asseguradas e toda a segurança do sistema pode desmoronar. De modo mitigar estes problemas foram aplicadas

mais duas estratégias: assinatura de valores Diffie-Hellman (protocolo “Station-to-Station”) e autenticação usando o Cartão do Cidadão português. No início da aplicação é pedido ao utilizador que se autentique usando o seu PIN de autenticação do Cartão de Cidadão e logo após é feita a assinatura digital do valor de Diffie-Hellman público gerado usando a chave privada de autenticação, também esta presente no Cartão do Cidadão. Por sua vez o servidor envia também ao cliente o seu valor público de diffie-hellman assinado usando a sua chave privada. Deste modo não é possível alguém se fazer passar por um cliente da aplicação, ou pelo próprio servidor, uma vez que a assinatura é feita usando a chave privada que apenas o cliente/servidor possui.

Inicialmente era nossa intenção usar a chave de sessão para cifrar o conteúdo todo das mensagens trocadas entre cliente e servidor. Para isso, após obter a chave de sessão era realizada uma operação de extensão da chave de modo a transformar a chave numa que fosse suportada pelo algoritmo de cifra escolhido, que no nosso caso foi o AES (Advanced Encryption Standard) que usa chaves de tamanho fixo de 256 bits. No entanto esta estratégia foi descartada, uma vez que seria mais simples, e menos pesado computacionalmente, a geração de um valor HMAC (Hash Message Authentication Code), usando a chave de sessão, que seria então enviado juntamente com a mensagem, e validado pelo seu recetor. A validação consiste na geração de um valor HMAC independente do recebido e comparando o valor obtido. Se o HMAC gerado coincide com o recebido então é possível afirmar que a mensagem recebida foi gerada pela entidade que possui a mesma chave de sessão.

Numa etapa mais avançada decidimos alterar o método tradicional de como o estabelecimento da chave de sessão era realizado. Em vez do servidor gerar um par de valores Diffie-Hellman para cada cliente com que comunica e seguidamente enviar o seu valor público ao cliente, optámos por assumir que o servidor possui uma componente pública bem conhecida por todos os clientes e um valor único privado que está armazenado em disco, devidamente protegido, do lado do servidor. Assim o cliente que pretende comunicar já conhece o valor público do servidor. Deste modo o cliente gera à mesma o seu par de valores Diffie-Hellman e calcula a chave de sessão mesmo antes de enviar o seu valor público ao servidor. Do lado do servidor a chave é gerada como anteriormente. No entanto agora como já não necessita de enviar a sua componente pública, envia uma mensagem “Ack”, uma confirmação ao cliente de que o estabelecimento da chave de sessão ocorreu sem problemas, ou uma mensagem “Error” caso o servidor não tenha conseguido validar a assinatura do cliente ou um outro erro tenha ocorrido, o que conduz ao aborto da sessão.

b. Criptografia assimétrica

A criptografia assimétrica, também conhecida por criptografia de chave pública foi implementada na aplicação com o objetivo de garantir a segurança do conteúdo das mensagens trocadas entre clientes. O algoritmo escolhido para este propósito foi o RSA (Rivest–Shamir–Adleman). Cada cliente, assim como o próprio servidor possuem um par de chaves assimétricas RSA. No caso do servidor poderia também ser criado um certificado de chave pública autoassinado. No entanto este iria servir o mesmo propósito do par de chaves, que é de implementação mais simples, daí a sua preferência. As chaves privadas dos clientes possuem 2048 bits enquanto que a privada do servidor é uma chave mais longa possuindo 4096 bits. O par de chaves do cliente é gerado durante o processo de criação de conta (mensagem CREATE), enquanto que o par de chaves do servidor foi gerado previamente. A chave pública é então enviada para o servidor, devidamente assinada, o qual a armazena.

Numa fase inicial a chave pública do servidor era enviada ao cliente durante o estabelecimento da chave de sessão (juntamente com o valor público de Diffie-Hellman do servidor devidamente assinado). No entanto adotamos o mesmo método (referido em a)) que o valor público de Diffie-Hellman para a chave pública do servidor, isto é, assumimos que o servidor possui uma chave pública bem conhecida por cada cliente de modo a simplificar o processo. Tanto as chaves dos clientes como a do servidor estão armazenadas em ficheiros em disco. Do lado de cada cliente está armazenada a sua chave pública e privada em formato PEM (Privacy Enhanced Mail). No entanto a chave privada é armazenada após ser devidamente cifrada usando uma password escolhida por este no processo de criação de conta. Do lado do cliente também está armazenada a chave pública do servidor e a componente pública de Diffie-Hellman, ambas também em formato PEM. Já do lado do servidor está armazenado o seu par de chaves assimétricas e par de Diffie-Hellman, estando também a chave privada e componente privada Diffie-Hellman armazenada de forma segura através de uma password apenas conhecida pelo servidor. O servidor ao ser inicializado faz “caching” dos seus valores privados de modo a evitar futuras leituras desnecessárias destes valores. O mesmo acontece para o par de chaves do cliente, caso este já possua uma conta no servidor.

Em termos práticos a chave pública de cada cliente é usada para cifrar a chave simétrica usada no processo de cifra híbrida das mensagens trocadas entre clientes (mensagem SEND), o que permite que apenas o destinatário possuindo a chave privada, correspondente à publica usada, possa decifrar a chave simétrica, conseguindo desse modo ler o conteúdo cifrado. Já o par de chaves do servidor desempenham um papel na geração da assinatura da componente pública de Diffie-Hellman no processo de geração de chave de sessão.

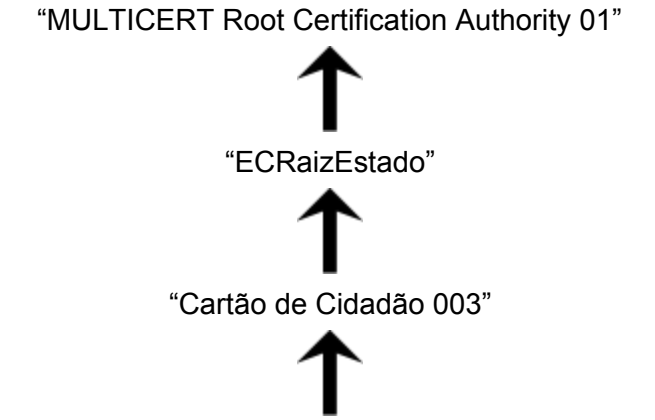
Para além das estratégias referidas acima, foi ainda aplicada criptografia assimétrica a partir das capacidades criptográficas presentes no Cartão de Cidadão dos clientes. Estas serão referidas em detalhe em 2.e).

c. Criptografia simétrica

Relativamente aos métodos de criptografia simétrica estes foram aplicados em duas operações criptográficas distintas: cifra do conteúdo das mensagens trocadas entre clientes (mensagem SEND) e geração de um valor HMAC que é enviado em conjunto com as mensagens trocadas entre os clientes e o servidor. Na primeira operação, o cliente remetente gera uma nova chave simétrica que será usada para cifrar o conteúdo da mensagem a enviar ("bulk encryption"), usando o algoritmo de cifra simétrica AES. Após a cifra ser realizada é obtido o criptograma e o vetor de inicialização ("Initialization Vector") sendo ambos enviados na mensagem SEND. A existência de um vetor de inicialização é importante uma vez que é este que garante que duas mensagens iguais cifradas com a mesma chave simétrica resulta em criptogramas diferentes, o que permite evitar ataques de texto escolhido ("Chosen-plaintext attacks" ou "CPA"). Por sua vez o destinatário da mensagem decifra a mensagem usando a mesma chave simétrica e o vetor de inicialização recebidos, obtendo assim a mensagem original em claro. Já a segunda operação, a geração de um valor HMAC, é usada de modo a garantir a autenticidade das mensagens trocadas entre cliente e servidor. Este valor é gerado usando a chave de sessão, conhecida apenas pelo cliente e pelo servidor, e a mensagem que se pretende enviar. O destinatário, ao receber a mensagem realiza a validação do HMAC recebido, calculando para tal um novo HMAC a partir do conteúdo da mensagem e da chave de sessão (chave de sessão associada ao utilizador que enviou a mensagem no caso do servidor). Se o novo HMAC produzido for igual ao recebido então é possível afirmar que o HMAC recebido foi gerado com a mesma chave de sessão e é possível concluir que a mensagem recebida foi de facto enviada por essa entidade e não outra.

d. Certificados de chave pública

Os certificados de chave pública são usados de forma a garantir a validade da chave pública que se pretende usar em algoritmos criptográficos. Para tal é necessário garantir que a chave pública associada a tal certificado é válida, o que requer a validação de toda a cadeia de certificação do certificado em questão. Esta validação de cadeia é feita através da validação do certificado de chave pública da entidade certificadora ("EC" ou "CA") imediatamente acima na cadeia de certificação e assim sucessivamente, até ser atingida a entidade certificadora raiz ("EC Raiz" ou "Root CA"). No caso do certificado de chave pública de autenticação um exemplo da cadeia de certificação é a seguinte:



Para realizar a validação de um certificado de chave pública é necessário o envio de uma mensagem POST (protocolo HTTP) de acordo com o RFC 2560 para os endereços http://ocsp.auc.cartaodecidadao.pt/publico/ocsp e http://ocsp.root.cartaodecidadao.pt/publico/ocsp respetivamente. É recebida uma resposta OCSP com a devida indicação da validade do certificado em questão. Esta funcionalidade não foi implementada na versão atual da aplicação. No entanto caso estivesse implementada, a validação ocorreria no momento do estabelecimento da chave de sessão, uma vez que o servidor ao receber o valor público de Diffie-Hellman autenticado necessita primeiro de validar o certificado e só após validar a assinatura. A validação da cadeia de certificação seria também realizada após o envio de uma mensagem “recipient_pk” na qual o cliente que enviou a mensagem obtém o certificado de chave pública de um dado cliente e as assinaturas sendo deste modo necessária a validação do certificado previamente à validação das assinaturas.

e. Cartão de Cidadão

O Cartão de Cidadão português foi implementado na aplicação com o objetivo de, não só autenticar os seus utilizadores, como também fazer uso das suas capacidades criptográficas para realizar autenticação de utilizadores e assinatura digital de conteúdo. Para tal foi utilizado o par de chaves assimétricas de autenticação presentes no mesmo, assim como o respetivo certificado de chave pública. O cliente que pretenda iniciar a aplicação necessita do Cartão de Cidadão devidamente inserido num leitor de smartcards conectado à máquina onde a mesma irá ser inicializada. Necessita então de introduzir o PIN de autenticação corretamente caso contrário a aplicação não pode continuar. É subentendido que todos os clientes possuem um Cartão do Cidadão, não sendo considerados utilizadores sem um. Após a autenticação, o certificado de chave pública de autenticação é extraído do cartão e é realizada a

assinatura digital, usando a chave privada de autenticação, do valor público de Diffie-Hellman a enviar ao servidor. O segundo processo de utilização da assinatura digital ocorre durante a criação de conta (mensagem CREATE), onde é realizada a assinatura da chave pública RSA do cliente e a assinatura do certificado de chave pública de autenticação. Neste passo o servidor não faz qualquer validação de assinaturas, estas são apenas dedicadas ao cliente que pretende enviar uma mensagem a outro, e que precisa para tal validar essas assinaturas ao fazer a aquisição da chave pública do destinatário. Também no processo de criação de conta é realizado um "Digest" do certificado de chave pública de autenticação do cliente, que serve como "impressão digital" do mesmo, usado como identificador único do cliente na aplicação.

f. Cache de chave pública

De modo a simplificar o processo de troca de mensagens entre clientes foi implementado um processo simples de "caching" da informação criptográfica pública dos mesmos, isto é, da sua chave pública RSA e do seu certificado de chave pública. Um cliente, antes de enviar uma mensagem a outro, envia uma mensagem do tipo "recipient_pk" de modo a obter a chave pública RSA do destinatário assim como o seu certificado de chave pública de autenticação usado para validar a assinatura da chave pública RSA. Deste modo ao fim da primeira comunicação bem sucedida com o cliente, a sua informação criptográfica é armazenada em memória num dicionário cuja chaves correspondem aos identificadores dos clientes (denominados internamente por "peers") e cujos valores são a chave pública e o seu certificado de chave pública. Assim sempre que dado cliente pretenda voltar a enviar uma mensagem não necessita novamente de obter a chave pública e o certificado pois já reconhece esse utilizador como seu "peer". No entanto esta é uma aproximação que, embora simplifique o processo de troca de mensagens entre clientes, é um processo "naive" uma vez que se assume que o certificado de chave pública não irá ser revogado durante o intervalo de tempo correspondente à sessão do cliente. Por outro lado sabemos que as CRLs (Certificate Revocation Lists) são emitidas geralmente num período de 24 em 24 horas, o que torna pouco provável que expire durante a sessão do cliente, que geralmente não é de longa duração, comparativamente a esse período.

3. Funcionalidades implementadas

a. Setup of a session key between a client and the server prior to exchange any command/response;
    i. Método de Diffie-Hellman + Station-to-Station.
b. Authentication (with the session key) and integrity control of all messages exchanged between client and server;

    i. HMAC gerado e transmitido para cada mensagem trocada cliente-servidor
c. Add to each server reply a genuineness warrant (i.e., something proving that the reply is the correct one for the client's request, and not for any other request);
    i. Implementação de um identificador de mensagem ("msg_id"). Este identificador é incrementado sequencialmente após receção da mensagem de resposta originada pela anterior. Cada resposta possui o mesmo identificador que a mensagem que lhe deu origem
d. Register relevant security-related data in a user creation process;
    i. Implementado na mensagem CREATE
e. Involve the Citizen Card in the user creation process;
    i. Implementado na autenticação de clientes e assinatura digital de valores
    ii. Digest do certificado de chave pública na criação de conta
f. Encrypt messages delivered to other users;
    i. Processo de cifra híbrida com uma chave simétrica gerada pelo remetente, posteriormente cifrada com a chave pública do destinatário
g. Signature of the messages delivered to other users (with the Citizen Card or another private key) and validation of those signatures;
    i. Ao enviar a mensagem "recipient_pk" é feita a validação das assinaturas
h. Encrypt messages saved in the receipt box;
    i. Mensagens cifradas com a chave pública do remetente uma vez que apenas este pode aceder ao seu conteúdo
i. Check receipts of sent messages;
    i. Implementado na mensagem STATUS
j. Prevent a user from reading messages from other than their own message box;
    i. Implementado na mensagem RECV
k. Prevent a user from sending a receipt for a message that they had not red
    i. Implementado na mensagem RECEIPT

4. Mensagens

a. Mensagem Diffie-Hellman ("dh")

Este tipo de mensagem é trocado durante o estabelecimento da chave de sessão entre o cliente e o servidor. O cliente gera os seus valores de Diffie-Hellman, usados durante a sua sessão, realiza a leitura da componente pública do servidor do seu ficheiro para memória e gera a chave de sessão. De seguida realiza a assinatura do seu valor e envia uma mensagem do tipo "dh" para o servidor. Esta mensagem contém o valor o valor público de Diffie-Hellman, a assinatura desse valor gerada usando a chave privada de autenticação do cartão de cidadão e o seu certificado de chave pública de autenticação. O servidor ao processar a mensagem lê o certificado do cliente

e valida a assinatura, usando a chave pública do mesmo. Após confirmação positiva da assinatura o servidor gera então uma chave de sessão usando o seu valor privado em conjunto com o público acabado de receber do cliente. Por fim envia uma mensagem “ack” caso a chave de sessão tenha sido gerada corretamente ou “Error” caso contrário, ambas assinadas devidamente com a chave privada do servidor.

Na Figura 2 está uma representação esquemática e simplificada deste processo, considerando as operações realizadas em ambos os intervenientes (utilizador e servidor).

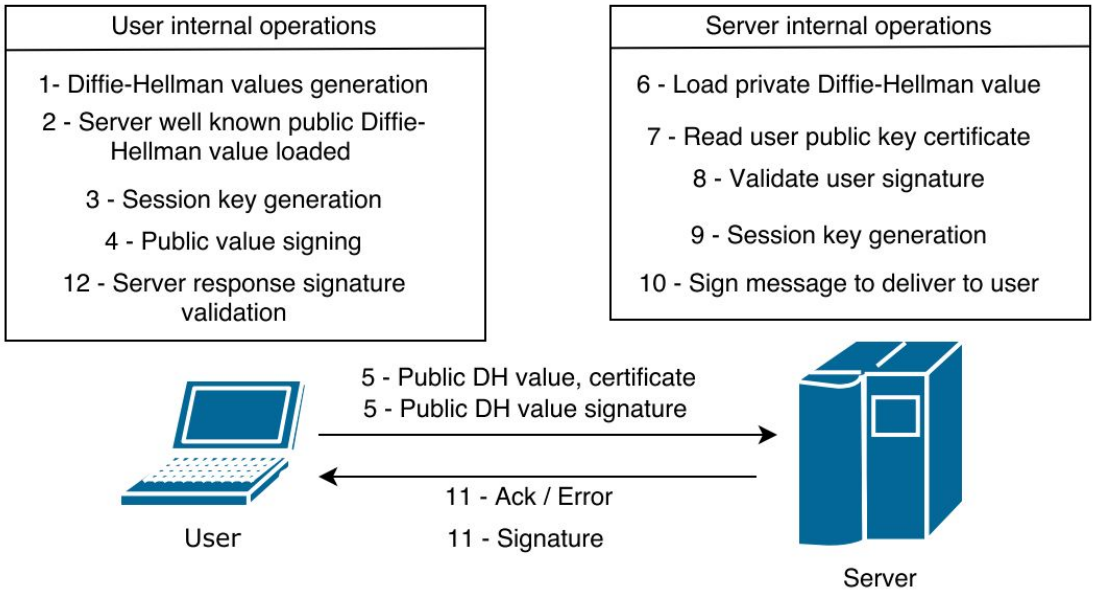


Fig. 2 - Estabelecimento da chave de sessão

b. Mensagem CREATE

A mensagem create é enviada sempre após o estabelecimento da chave de sessão. É gerado um par de chaves assimétricas RSA, é feito o “digest” do certificado de chave pública de autenticação (que corresponde ao campo “uuid” da mensagem CREATE) e são preenchidos os campos da mensagem com a chave pública RSA acabada de gerar, certificado de chave pública, e respetivas assinaturas destas duas últimas. O HMAC correspondente à mensagem é criado e enviado em conjunto com a mensagem para o servidor. O servidor ao validar o HMAC, processa a mensagem e verifica se já existe algum utilizador no servidor com o “uuid” acabado de receber. Caso exista é enviado o id interno do cliente juntamente com um indicador de existência de conta, caso contrário é enviado um novo identificador e indicação do mesmo. Em caso de existência de conta, o novo par de chaves gerado é ignorado e é feita a leitura das chaves em ficheiro do cliente, introduzindo a password de segurança da chave privada

(assume-se a existência das mesmas na pasta do cliente). No caso de uma conta nova então o novo par de chaves gerado não é descartado, o utilizador introduz uma password de modo a armazenar seguramente a chave privada, e ambas as chaves são armazenadas em ficheiro na máquina do cliente. Em ambos os casos é feita a validação do HMAC primeiro e só depois é processado o conteúdo recebido.

c. Mensagem de aquisição de chave pública (“recipient_pk”)

Esta mensagem é enviada sempre antes do envio de uma mensagem do tipo SEND para um destinatário cujo remetente não possua a sua informação criptográfica pública (certificado de chave pública de autenticação e chave pública RSA) na sua cache de “peers”. A mensagem contém o id do destinatário, ao qual se destina a mensagem SEND. Juntamente com a mensagem “recipient_pk” é também enviado o HMAC gerado a partir desta. O servidor ao fazer o processamento da mensagem valida o seu HMAC e de seguida verifica se o utilizador com tal id existe. Caso exista irá devolver então uma mensagem do mesmo tipo ao cliente contendo a chave pública RSA, assinatura da chave pública RSA, certificado de chave pública de autenticação e assinatura do certificado de chave pública de autenticação. É também gerado um HMAC e a mensagem é então enviada ao cliente. Ao receber a mensagem o cliente realiza a validação do HMAC, seguindo-se da validação das assinaturas. Por fim a informação do destinatário é incluída na cache do remetente.

d. Mensagem SEND

Esta mensagem é a mensagem principal de comunicação entre clientes. Posteriormente ao envio desta mensagem é feita a verificação se o destinatário já é “peer” do remetente. Caso contrário é gerada e enviada uma mensagem do tipo “recipient_pk” como visto acima em 4.c). De seguida realiza-se o processo de cifra híbrida no qual o remetente gera uma nova chave simétrica a qual é usada para cifrar o conteúdo da mensagem, obtendo o criptograma e o vetor de inicialização. A chave simétrica por sua vez é cifrada usando a chave pública do destinatário. Já a cópia da mensagem vai ser cifrada com a chave pública do remetente uma vez que se pretende que apenas este possa aceder ao seu conteúdo na caixa de recibos. O HMAC respetivo é gerado e a mensagem é então enviada. Na mensagem é enviado o criptograma, o vetor de inicialização e a chave simétrica cifrada e a cópia cifrada. Do lado do servidor a mensagem é recebida, HMAC é validado e o criptograma, iv, e chave cifrada são armazenados na caixa de recibos do destinatário. Já a cópia é armazenada na caixa de recibos do remetente. O servidor responde com uma mensagem contendo o identificador da mensagem na caixa de mensagens e da caixa de recibos, com o HMAC propriamente gerado.

e. Mensagem RECV

A mensagem RECV é enviada quando o cliente quer ver a mensagem que recebeu na caixa de mensagens. Contém o tipo, o identificador do cliente, o identificador da mensagem, que pode ser obtido com a mensagem LIST(secção 4.h)), e o HMAC, gerado antes do envio da mensagem. No servidor a mensagem é recebida, HMAC é validado e o servidor responde com o identificador do remetente e com a mensagem lida, com o HMAC propriamente gerado. A mensagem lida pode não ser compreendida, se o utilizador que recebe a mensagem não seja o mesmo que está a ler pois não possui a chave privada correspondente.

f. Mensagem RECEIPT

A mensagem RECEIPT é enviada pelo cliente depois de receber a mensagem. A mensagem tem como campos o tipo, o id da caixa de mensagens do remetente, o id da mensagem pela qual vai ser enviado o recibo, o receipt que contém a mensagem assinada com a chave privada do Cartão de Cidadão e o HMAC. O servidor depois de receber a mensagem vai apenas validar o HMAC, não respondendo.

g. Mensagem ALL

A mensagem ALL vai listar todas as mensagens de um cliente na sua caixa de mensagens. A mensagem apenas precisa do tipo, do cliente em questão e, antes de enviar, o HMAC gerado. O servidor responde com dois arrays: um com os identificadores das mensagens recebidas e outro com os identificadores das mensagens enviadas. Anteriormente, recebe a mensagem e valida o HMAC. Posteriormente, gera um HMAC para o envio da resposta.

h. Mensagem LIST

A mensagem LIST é enviada a pedido do cliente, que quer saber que outros utilizadores têm uma caixa de mensagens no servidor. Esta mensagem tem duas opções de escolha: ou lista a informação de todos os clientes, ou apenas de um cliente em particular, passando para esse efeito o id desse cliente em questão. Antes da mensagem ser enviada é gerado o HMAC. Na mensagem é enviada a mensagem juntamente com o HMAC gerado, dentro da mensagem vai o tipo e o id do cliente em questão dependendo se essa foi a opção escolhida. Do lado do servidor a mensagem é recebida, HMAC é validado e o servidor responde com a informação do cliente, com o HMAC propriamente gerado. A informação corresponde à mensagem CREATE que está descrita em 4.b).

i. Mensagem NEW

A mensagem NEW é enviada quando o cliente quer saber que novas mensagens recebeu. Para tal é necessário o id do utilizador que se quer ver.
É gerado o HMAC e posteriormente a mensagem é enviada. Na mensagem é enviada a mensagem juntamente com o HMAC gerado, dentro da mensagem contém o tipo e o id do cliente. No servidor a mensagem é recebida e o validado o HMAC e o servidor responde com a lista de identificadores das mensagens novas, com o HMAC propriamente gerado. Se o cliente pedir para ver a caixa de mensagens que não a sua dá erro de permissão.

j. Mensagem STATUS

A mensagem STATUS serve para a verificação da receção de uma enviada mensagem, ou seja, se tem ou não RECEIPT(secção 4.f)) e se é válido. A mensagem requisita o identificador da caixa de recibos, que é o mesmo da caixa de mensagens, o identificador da mensagem enviada e, por fim, o tipo da mensagem. A mensagem é posteriormente enviada juntamente com o HMAC gerado. O servidor, após a receção e a validação do HMAC, vai enviar a mensagem enviada, cujo identificador foi enviado, e uma lista contendo os dados do recibo, a identidade do remetente do recibo e o próprio recibo e, posteriormente, o HMAC gerado.

5. Arquitetura

Em relação à arquitetura da aplicação esta está dividida em 3 módulos principais: o do cliente, o do servidor e o módulo usado pelo cliente e servidor que possui as funções usadas por ambos para aplicar algoritmos criptográficos na aplicação. O ficheiro “client.py” corresponde à aplicação a ser executada pelo utilizador da aplicação que contém o menu de operações que este pode realizar neste contexto. Cada operação corresponde a uma mensagem numa estrutura e formato bem conhecido ao servidor e que são enviadas ao mesmo. Já os módulo do servidor é composto por 4 ficheiros: “server.py”, “server_actions.py”, “server_client.py” e “server_registry.py”, sendo o primeiro destes o principal, que permite executar o servidor. Tanto o módulo do cliente como o do servidor importam as funções presentes no módulo criptográfico, constituído por dois ficheiros: “security.py”, que contém todos os algoritmos criptográficos usados, e o “citizen_card.py” que contém as funções e algoritmos usados para comunicar com o Cartão de Cidadão do cliente. Ainda para o primeiro, foram usados métodos provenientes da biblioteca python “cryptography” enquanto que no “citizen_card.py” foram implementados métodos provenientes da biblioteca python “PyKCS11”, para comunicação e extração de informação criptográfica

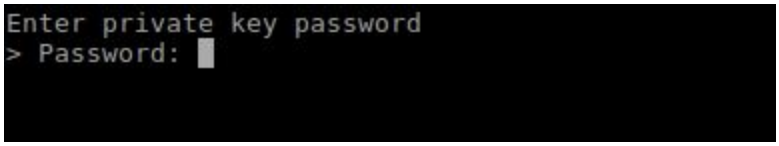
do Cartão de Cidadão, e da biblioteca “PyOpenSSL” para serialização e gestão de certificados de chave pública.

6. Funcionamento da aplicação

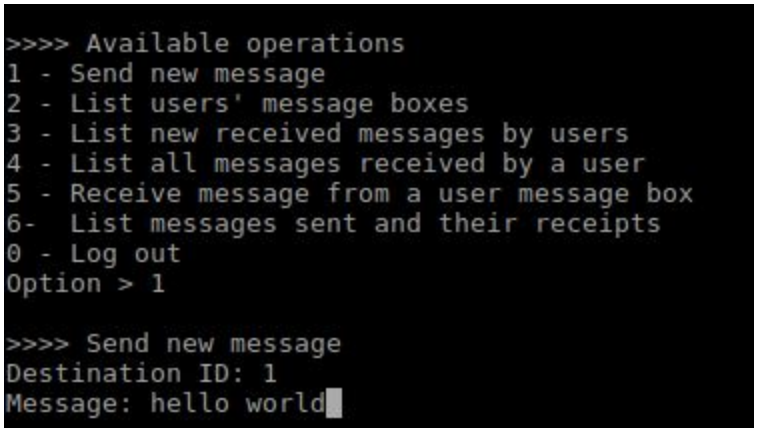
Inicialmente é pedido ao cliente que se autentique na aplicação, introduzindo para tal o seu PIN de autenticação do Cartão de Cidadão.

De seguida é pedido novamente o PIN de autenticação de modo a assinar o valor público de Diffie-Hellman que será enviado para o servidor. Após a sessão ter sido estabelecida corretamente é enviada uma mensagem CREATE, na qual é necessário assinaturas de chave pública RSA e do certificado de chave pública de autenticação a ser enviado para o servidor

De seguida, é feita a introdução de uma password, usada para armazenar seguramente a chave privada RSA em ficheiro:

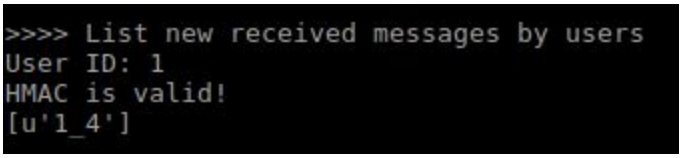


Por fim é apresentado ao utilizador o menu de opções que este pode efetuar, correspondendo cada uma delas a uma mensagem enviada ao servidor.



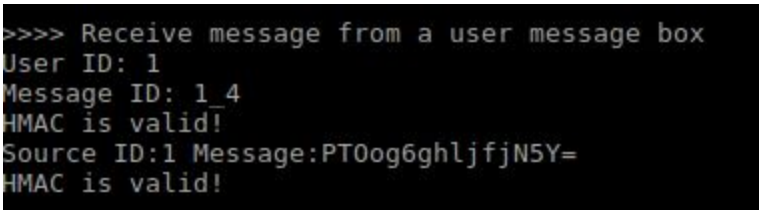
É introduzido o ID do destinatário da a mensagem a enviar, seguindo-se do conteúdo da mensagem.

É também possível ver as novas mensagens recebidas passando o ID do utilizador.

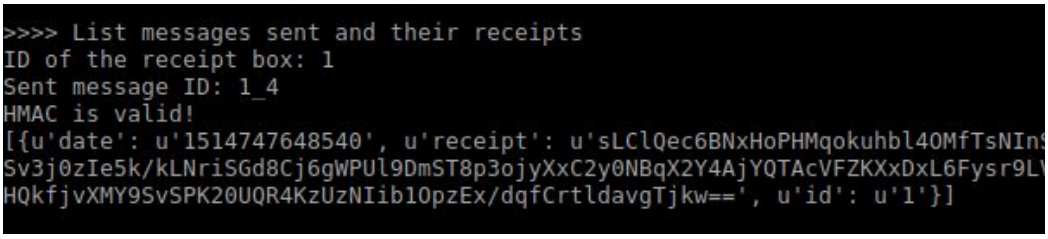


Podemos prosseguir para receber a mensagem recebida na opção 5

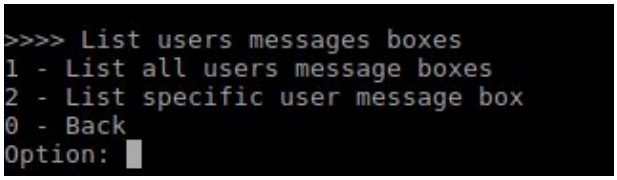
Nesta opção será pedido o ID do cliente e o ID da mensagem recebida consultada na última operação. Após receber a mensagem é enviado um recebido para o qual é pedido novamente o PIN de autenticação.



Podemos consultar o estado dos recibos na opção 6 na qual é pedido o ID da caixa de recibos, que é o mesmo da caixa de mensagens para facilitar, e o ID da mensagem.



Também é possível listar as caixas de mensagens dos utilizadores na opção 2 e pode-se listar todas a caixas de mensagens ou apenas de um utilizador.



Por fim existe a operação de listar todas as mensagens de um utilizador

Se o utilizador pretender encerrar a aplicação, basta escolher a opção 0 de log out, terminando assim a sessão com o servidor.

7. Pré-requisitos da aplicação

a. Versão python 2.7(.13)
b. PyKCS11 versão 1.4.4
c. Cryptography versão 1.7.1
d. pyOpenSSL versão 16.2.0
e. Todo o código foi realizado e testado no sistema operativo Debian GNU/Linux 9.2 (Máquina virtual da unidade curricular)

8. Problemas da aplicação

a. Mensagem CREATE
    i. A mensagem CREATE é sempre enviada ao iniciar sessão na aplicação mesmo que o utilizador já possua uma caixa de mensagens no servidor. Internamente o servidor responde com o identificador interno do utilizador, verificando se este já existe, atribuindo-lhe e devolvendo o identificador correspondente, ou gerando um novo caso contrário. Não se trata, no entanto, de uma boa implementação uma vez que, embora o utilizador já possua uma caixa de mensagens, ainda assim é necessária a geração de um novo par de chaves assimétricas, da assinatura de chave pública e da assinatura do certificado de chave pública, operações estas que são descartadas e inúteis caso o servidor responda positivamente. Uma estratégia que poderia ser usada para corrigir este problema seria o envio prévio de uma mensagem ou "query", perguntado ao servidor se já existe alguma caixa de mensagens associada ao uuid que será usado na mensagem CREATE. O servidor iria então responder e só após receber esta resposta se iria gerar, ou não, o par de chaves assimétricas e as respetivas assinaturas.

b. Servidor local
    i. Uma vez que a aplicação foi desenvolvida tendo em conta o servidor como agente local, nada nos garante a correção das mensagens trocadas com o mesmo caso este fosse instalado numa máquina acessível pela internet

c. Armazenamento de chaves em disco
    i. O armazenamento das chaves assimétricas é feito em ficheiros em disco tanto do lado do cliente como do lado do servidor. Deste modo subentende-se que estas estão sempre presentes sempre no mesmo diretório não havendo mecanismos de deteção de erro caso contrário.
    ii. As chaves privadas dos clientes, armazenadas em ficheiro, são guardadas em formato PEM cifrado usando uma password escolhida pelo mesmo no processo de criação de conta. No entanto não foram implementados mecanismos de validação das mesmas, estando o cliente sujeito a escolher uma password criptograficamente fraca, podendo comprometer a sua chave privada, o que não pode acontecer.
    iii. Em relação à experiência de utilização do cliente da aplicação surge também aqui um problema uma vez que se torna enfadonho para o mesmo a necessidade de introdução de um password para proteger o conteúdo da sua chave privada, isto após ter de introduzir um PIN de autenticação e realizar 3 assinaturas digitais, ou seja mais 3 introduções de PIN. Não foi prestada muita atenção a este problema uma vez que não se enquadra dentro dos objetivos da cadeira nem do projeto, no entanto achamos adequado considerar o mesmo como um problema.

Uma aproximação teórica pensada para o resolver seria usar como password uma hash (SHA256) do valor do PIN de autenticação do cliente, em vez de uma hash da password como está implementado. No entanto seria um valor hash fraco e facilmente alcançável, pelo que novamente iriam surgir problemas de segurança (como visto em 8.c) ii.).

9. Trabalho futuro

Apesar de tudo não foi possível concluir todos os objetivos propostos inicialmente. A seguir estão listadas algumas das estratégias e aproximações que pretendiamos implementar na aplicação:

a. Validação de cadeia de certificação dos certificados de chave pública
b. Tradução de id internos por nicknames ou email
c. Envio de uma mensagem de verificação de existência de conta de um utilizador, previamente ao envio da mensagem CREATE
d. Implementação de algoritmos criptográficos alternativos

10. Conclusões

Após a realização deste trabalho foi possível compreender melhor os métodos criptográficos usados atualmente para implementação de sistemas informáticos seguros. Foi também possível concluir que se tratam de processos complexos e que requerem um grande investimento em tempo e recursos de modo a obter os melhores resultados possíveis. A construção de um sistema seguro necessita, antes da sua conceção, de um planeamento organizado e bem definido dos métodos criptográficos a implementar, assim como medir quer os benefícios, quer as desvantagens das suas implementações . É também um requisito fundamental, para a definição de estratégias de segurança, tomar a perspectiva de um intruso no sistema, considerando desde a falsificação de mensagens até à personificação de clientes da plataforma, de modo a poder prevenir estas ocorrências no futuro. A integração do Cartão do Cidadão na aplicação contribuiu imenso para o aumento da segurança do nosso sistema, pelo facto de possui pares de chaves assimétricas devidamente armazenadas no cartão, acessíveis através do “middleware” apropriado. Este é um elemento criptográfico poderoso que é, no entanto, pouco explorado pelo cidadão comum que desconhece das suas capacidades de autenticação e de assinatura. Ao realizar o trabalho foi também possível concluir da dificuldade existente entre conciliar uma boa experiência de utilização aos clientes da aplicação e mecanismos apropriados de segurança, que muitas vezes reduzem a anterior. Por fim podemos aferir que foi um trabalho bastante desafiador e complexo que no entanto contribui bastante para o nosso crescimento

como estudantes da área da informática e que permitiu o uso de ferramentas criptográficas complexas que nos eram anteriormente desconhecidas.

11. Referencias

a. Cryptography python library - https://cryptography.io
b. Python OpenSSL library - https://pyopenssl.org
c. PyKCS11 Python wrapper - https://github.com/LudovicRousseau/PyKCS11
d. PKI Cartão Cidadão - http://pki.cartaodecidadao.pt/
e. Politica de certificado de autenticação - http://pki.cartaodecidadao.pt/publico/politicas/PJ.CC_24.1.2_0011_pt_AuC.pdf
f. Main bibliography - Segurança em Redes Informáticas, André Zúquete, 4º Ed. Aumentada, FCA - Editora de informática, Lda