
Mestrado Integrado em Engenharia Informática e Computação

**SSIN**

# Practical Security Assignment

**Grupo ID:** 2

**Autores:**

Ana Rita Fonseca Santos up201605240@edu.fe.up.pt

Eduardo João Santana Macedo up201703658@edu.fe.up.pt

Pedro Leite Galvão up201700488@edu.fe.up.pt

Raúl Manuel Fidalgo da Silva Teixeira Viana up201208089@edu.fe.up.pt

23 de maio de 2021

# Índice

# Introdução

Este trabalho prático tem como objetivo cimentar os tópicos teóricos abordados nas aulas de Segurança em Sistemas Informáticos, nomeadamente a criptografia, as suas aplicações e as comunicações seguras. Para isso foi pedido que se implementasse um sistema de informação digital para uma hipotética empresa, que permitisse a utilização de determinados serviços de uma forma segura e confidencial.

Dessa forma foram apresentados determinados requisitos e foi aconselhado um desenho do sistema na forma da figura seguinte:

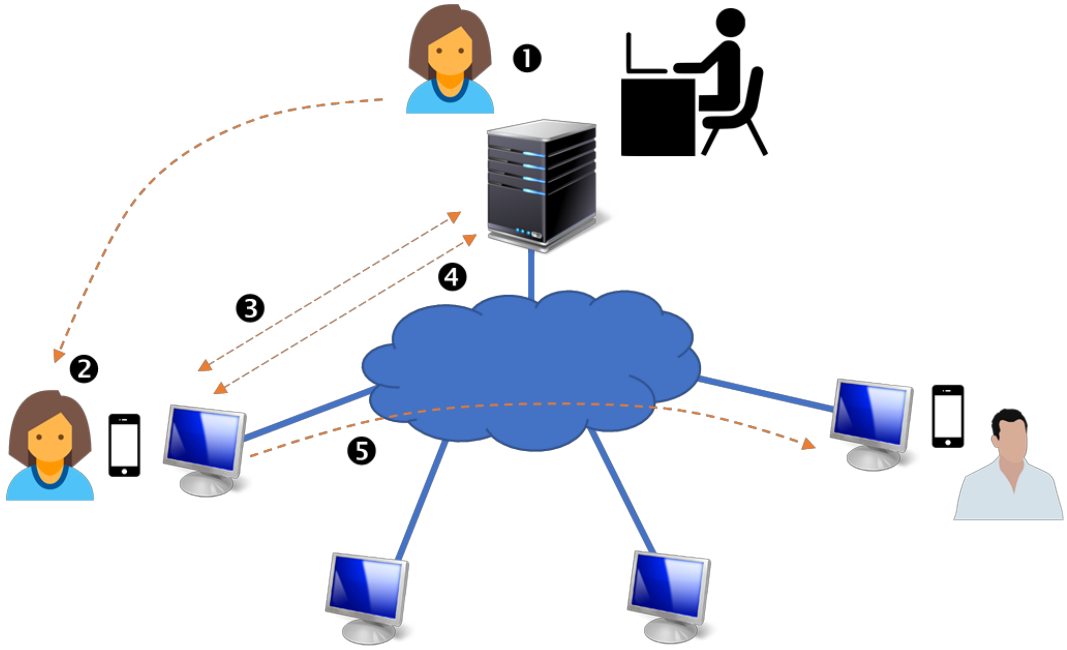


Figura 1: esquema do sistema a implementar

Como é possível constatar na figura 1 o sistema é composto por um servidor central que lida com os pedidos dos clientes, havendo ainda a possibilidade de os clientes comunicarem entre si. Estão definidos cinco casos de uso que serão descritos com mais detalhe adiante.

1. Registo presencial do colaborador no sistema da empresa;
2. Registo por parte do colaborador no seu cliente pessoal;
3. Registo do cliente no servidor;
4. Pedido de um serviço pelo cliente ao servidor e consequente resposta;
5. Envio de mensagem entre clientes.

Assim, é possível concluir que existem variados pontos vulneráveis susceptíveis de ataques maliciosos que poderão roubar ou adulterar mensagens, ou mesmo tentar personificar clientes ou colaboradores. Será então necessário utilizar meios criptográficos para mitigar o risco de que a informação que circula na rede, e se destina apenas e só à empresa e aos seus colaboradores, possa ser interceptada ou adulterada.

Será assim imperativo que todas as mensagens que circulam entre os vários dispositivos e os ficheiros por eles guardados sejam cifradas, e que contenham assinaturas digitais.

Foi considerado, tendo em conta o grau de experiência de programação e a familiaridade com as linguagens dos membros da equipa de desenvolvimento, que a melhor opção seria desenvolver o cliente em *python* e o servidor em *NodeJS*. Esta combinação garante os melhores resultados, tendo em conta os requisitos apresentados e as características da equipa de desenvolvimento.

# 1. Registo Presencial

O registo presencial deverá ser realizado presencialmente, na empresa, pelo colaborador antes da primeira utilização do cliente. Neste registo será adicionada uma entrada na base de dados do servidor com a informação do colaborador, nomeadamente o seu nome e o *username*. É também criado um *one time ID* que é adicionado à entrada do colaborador na base de dados e é disponibilizado ao colaborador. Este *one time ID* vai ser posteriormente utilizado no próximo ponto.

Com o registo presencial inicial eliminam-se alguns problemas de segurança relativos a este procedimento, sendo que pode ser considerado um procedimento bastante seguro.

# 2. Registo do Colaborador no Cliente

O registo do colaborador no cliente é um passo importante, uma vez que vai vincular esse cliente a esse colaborador. Na primeira utilização, depois do *download* e instalação do cliente, o colaborador insere o *one time ID* que lhe foi fornecido no ponto anterior, o que comprova inequivocamente a sua identidade. É também pedido ao colaborador que insira um *username* e uma *password,* que vão ser utilizados daí em diante para realizar a autenticação do colaborador no cliente. Assim, a autenticação do colaborador será realizada apenas com um fator de autenticação, a *password*, no entanto, como a aplicação será para ser utilizada num dispositivo pessoal isso será suficiente para garantir um bom nível de segurança.

A *password* pedida é bastante forte, sendo apenas aceites *passwords* com uma extensão mínima de 8 caracteres, constituídos por letras minúsculas e maiúsculas e números.

Todos estes dados, e outros que serão descritos nos pontos seguintes, serão guardados num ficheiro ".env" que, se não existir, ou se se encontrar corrompido, é criado no cliente. Este ficheiro será encriptado e desencriptado utilizando como chave a *password* facultada pelo colaborador no momento da autenticação. É utilizada a biblioteca "pyAesCrypt" e o algoritmo AES256-CBC.

Desta forma o segredo não é guardado fisicamente no dispositivo, diminuindo assim o risco de exposição e transferindo o ônus da manutenção da segurança da *password* para o colaborador.

# 3. Registo do Cliente no Servidor e Autenticação

O registo do cliente com as credenciais do colaborador é realizado da primeira vez que o cliente se liga ao servidor. É realizada uma adaptação do protocolo de Diffie–Hellman. Temos consciência que em criptografia não devemos modificar protocolos que comprovadamente resultam e são seguros, mas como se trata de uma prova de conceito, e o objetivo passa por mostrar que compreendemos os conceitos teóricos, optámos por adaptar o protocolo à nossa realidade específica.

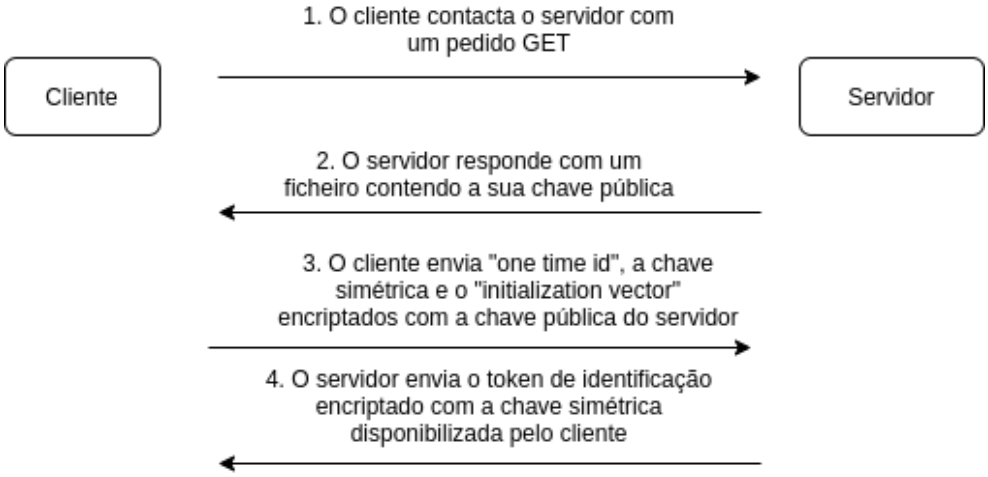


Figura 2: protocolo de registo

A figura 2 representa o protocolo inicial de registo do cliente no servidor. Todas as informações são transmitidas de forma encriptada pela rede, exceptuando a chave pública do servidor, sendo que nenhum segredo ou informação necessária para o estabelecimento das comunicações encriptadas é trocado de forma insegura. A chave simétrica trocada entre o cliente e o servidor vai servir para encriptar e desencriptar todas as mensagens trocadas entre eles, deste momento em diante. É comum que o *initialization vector* (IV) seja concatenado com a mensagem e que sejam transmitidos como um só. No entanto, como neste sistema estamos a utilizar REST API no servidor e a enviar as mensagens sob a forma de *json*, optámos por enviar o IV com uma *json key* própria, o que facilita o tratamento da mensagem. O *token* disponibilizado ao cliente pelo servidor no fim do protocolo servirá para o cliente se identificar nas próximas mensagens. Como forma de minimizar um ataque de repetição, para além de o *token* ser renovado sempre que o cliente inicia uma nova sessão, este pedido contém ainda um *lifetime* encriptado. O servidor verifica a validade do *token* e da janela temporal .

O servidor persiste todos estes dados referentes aos clientes numa base de dados *sqlite*. De forma a reduzir a possibilidade de um ataque por *sql injection* todos os valores que são introduzidos nas *queries* à base de dados são sanitizados e é utilizada a preparação dos *statements.*

A partir deste momento o servidor e o cliente possuem as ferramentas criptográficas para se comunicarem de forma segura e confidencial. É utilizada a chave simétrica trocada anteriormente para encriptar e desencriptar todas as mensagens trocadas entre as duas entidades. Esta encriptação e desencriptação é realizada através do algoritmo AES, modo

CBC. Neste algoritmo são utilizados como *inputs,* tanto para a encriptação como para a desencriptação, a mensagem, a chave simétrica e o IV. É criado um novo IV sempre que se deseja encriptar uma mensagem e este é enviado juntamente com a mensagem, de forma a poder ser utilizado na desencriptação.

As operações de registo do colaborador no cliente e de registo do cliente no servidor são operações únicas. Posteriormente, sempre que o colaborador utilizar a aplicação cliente terá de se autenticar com o seu *username* e *password.* Da mesma forma, o cliente realiza também uma autenticação no servidor, de forma automática. Esta autenticação implica resolver um *challenge* por parte do cliente, lançado pelo servidor. Se o cliente resolver de forma satisfatória este *challenge*, o servidor envia um novo *token* ao cliente, *token* esse que irá ser utilizado durante o resto da sessão.

Depois do registo o cliente cria um par de chaves pública e privada e envia a sua chave pública para o servidor, para que esta seja guardada por ele. Desta forma o servidor possui as chaves públicas de todos os clientes que estão que se lhe ligam, possibilitando a posterior troca de mensagens de forma criptograficamente segura entre clientes.

# 4. Pedido de Serviço

Depois da autenticação o cliente apresenta ao colaborador as várias opções disponíveis: requisitar um serviço ao servidor, enviar uma mensagem a outro colaborador ou ver as mensagens guardadas. Existem três serviços disponíveis, cada um com o seu nível de acesso. O serviço *“square root”* tem o nível de acesso definido mais baixo - 1. O serviço “*cubic root”* tem o nível intermédio - 2, e o serviço "parameterized *root”* tem um nível de acesso mais alto - 3. Isto significa que um colaborador com um nível de acesso 3 consegue aceder a todos os serviços, enquanto que um colaborador com um nível de acesso 1 apenas pode requisitar o serviço “*square root”*.

O colaborador pode escolher uma de entre as várias opções através do *input* de um inteiro. Como uma das grandes preocupações e possíveis explorações ao nível da segurança está precisamente no *input* dos utilizadores, apenas é admissível ao utilizador introduzir um de três números inteiros, correspondentes à opção escolhida. Assim reduz-se a possibilidade de exploração de ataques que façam uso de *inputs* maliciosos por parte do utilizador.

Para requisitar um serviço ao servidor o cliente envia um pedido composto pelas componentes de identificação - o *username* e o *token* e pelas componentes de encriptação - o IV. Envia ainda a identificação do serviço requerido.

O servidor envia a resposta do serviço solicitado, no caso de o cliente existir, de o *token* ser o correto e de o colaborador possuir o grau de acesso necessário para a requisição daquele serviço. Todas estas comunicações utilizam a encriptação simétrica estabelecida no ponto anterior.

# 5. Mensagens entre Clientes

Os clientes podem trocar mensagens entre si. Para possibilitar esta opção ao colaborador foi implementado um algoritmo de preparação de envio e recepção destas mensagens.

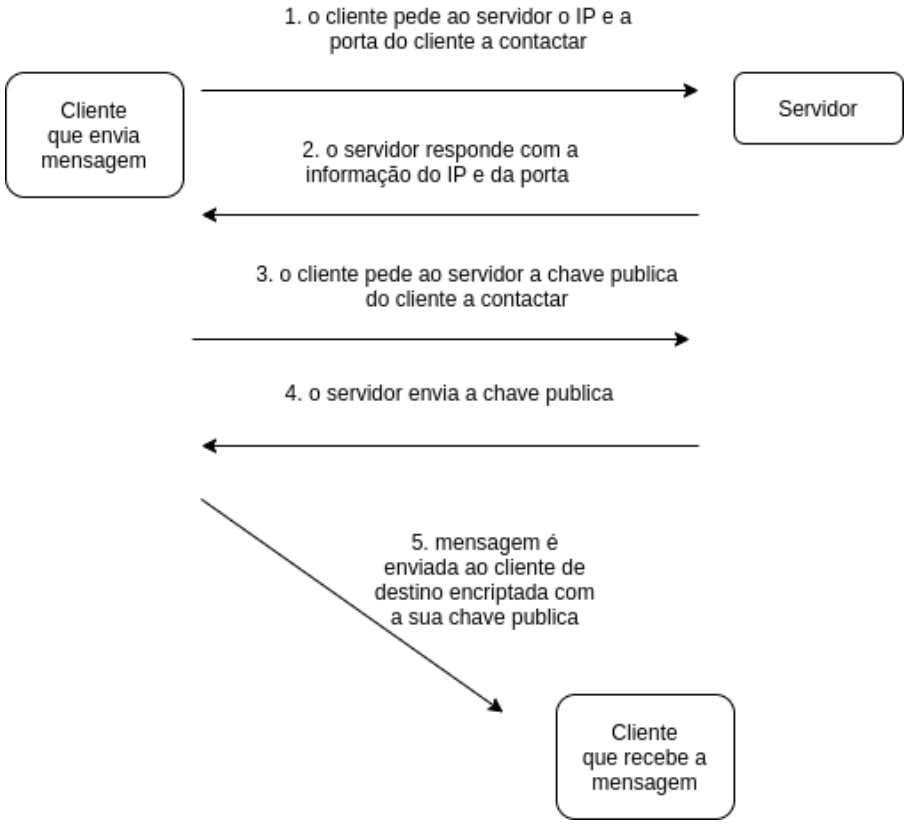


Figura 3: envio de mensagem entre clientes

O cliente começa por pedir ao servidor o IP e a porta do cliente com o *username* a contactar. Se o servidor tiver informação de que este cliente está *online,* devolve a informação ao cliente que a pediu. De seguida o cliente pede ainda a chave pública do cliente de destino da mensagem. Desta forma o cliente de origem da mensagem pode encriptá-la de forma a que só o cliente de destino a possa desencriptar, pois é o único que dispõem da correspondente chave privada.

Desta forma as mensagens trocadas entre clientes circulam na rede de forma encriptada e portanto segura. O algoritmo de encriptação e desencriptação utilizado foi o PKCS1_OAEP. Este algoritmo foi introduzido por Bellare e Rogaway e tem como características principais adicionar um elemento de aleatoriedade, fomentando um esquema probabilístico e prevenir desencriptações parciais da mensagem original. Por outro lado é também enviada uma assinatura da mensagem, realizada com o algoritmo PKCS 1 1.5. Esta encriptação, realizada sobre a *hash* da mensagem utilizando a chave privada do remetente, assegura que a mensagem foi criada pelo cliente de origem e que não foi de qualquer forma editada ou alterada, correspondendo assim à mensagem original. O destinatário verifica a assinatura utilizando a chave pública do remetente e assim é assegurado o não repúdio e limitada a possibilidade de um ataque *man in the middle*.

O cliente de destino da mensagem recebe esta num *thread*, criado especificamente para isso. Esse *thread* fica à escuta de mensagens enviadas e guarda-as quando as recebe, ou as descarta quando a assinatura digital não assegura a sua integridade. O cliente tem ainda a funcionalidade de mostrar as mensagens gravadas que já recebeu até ao momento. Para isso o ficheiro que contém as mensagens é desencriptado e imprimido, de forma a que o colaborador possa ler as suas mensagens.

# Conclusão

Os objetivos propostos para este trabalho foram atingidos, tendo sido possível colocar em prática, de uma forma bastante realista, os conceitos teóricos abordados durante as aulas práticas.

Foi possível compreender de uma forma mais aprofundada como lidar com a segurança de uma aplicação informática, como aplicar os conceitos de criptografia às comunicações em rede e ainda a importância extrema que a segurança informática possui hoje em dia e a sua fragilidade em certos aspetos.

Gostaríamos de ter tido oportunidade de desenvolver um pouco mais a segurança da aplicação implementada, nomeadamente ter acrescentado *message digest* noutros pontos da aplicação para além das mensagens trocadas entre clientes. Estes certificados digitais de autoria e integridade poderiam ter sido aplicados aos ficheiros criados pelo cliente e às mensagens trocadas entre o cliente e o servidor. Isto iria adicionar mais uma camada de segurança à aplicação.