Blockchain-based Auction Management

Membros do Trabalho: Catarina Silva, 76399; António Silva, 76678
Professor: André Zúquete
Segurança

3 de Fevereiro de 2019

Departamento de Eletrónica, Telecomunicações e Informática

Conteúdo

1 Introdução

No âmbito da unidade curricular Segurança, inserida no mestrado integrado em Engenharia de Computadores e Telemática realizou-se o projeto Blockchain-based auction management.

O objetivo do projeto é o desenvolvimento de um sistema que forneça aos utilizadores a capacidade de criar e participar em leilões. Como em qualquer outro sistema, a componente de segurança assume um papel fundamental, visto que um dado sistema independentemente da sua especificação se não possuir mecanismos de segurança torna-se vulnerável a muitos ataques conhecidos ou não, o que faz com que se corram riscos muitos deles ao nível da privacidade dos utilizadores ou situações ainda piores. Existe portanto uma necessidade de defesa contra atividades não autorizadas, isto é tem que haver uma defesa ao nível sistemas computacionais perante iniciativas tomadas por individuos com o intuito de distorcer o normal funcionamento do sistema. Sendo atividades deliberadas que visam a corrupção ou subversão dos sistemas, os mecanismos de segurança implementados tornam-se fundamentais para manter o bom funcionamento. Operações como acesso a informação sensível ou confidencial para um grupo de pessoas, alteração de informação com vista a alterar ou eliminar a mesma mas sem autorização ou sem poderes para tal, utilização exagerada ou abusiva de recursos computacionais, vandalismo e/ou impedimento de prestação de serviços são atividades ilícitas que podem por em causa todo e qualquer um sistema que não esteja preparado para lidar com as vulnerabilidades a que está sujeito. Desta forma, implementou-se um sistema que permite a criação de leilões e a participação nos mesmos mas deu-se uma elevada relevância aos mecanismos de segurança fornecendo um sistema seguro contra os ataques conhecidos, tendo começado por fazer um levantamento dos possíveis riscos e vulnerabilidades. Deu-se prioridade à transmissão de ficheiros seguros, quer seja assinados ou cifrados, independentemente do canal.

O sistema é constituido por um cliente que pode criar um novo leilão ou participar em algum já existente, e dois servidores com funções diferentes. Um dos servidores denominado por Auction Manager é o componente com que o cliente interage para criar um leilão. O outro servidor, o Auction Repository, tem como principal função o armazenamento de leilões e é com este componente que o cliente comunica quando pretende fornecer uma oferta para um dado leilão. A implementação destes dois componentes será descrita de forma mais detalhada nas seguintes secções. Tendo em conta este esquema de interação entre os diferentes componentes interessa referir algumas políticas que tem que ser tidas em conta para o bom funcionamento ao nível da segurança. Como as ofertas contém informação sensível tem que se resolver esta situação de forma a tornar os dados confidenciais, manter a sua integridade e fornecer mecanismos de autenticação. As ofertas são aceites tendo em conta alguns critérios especiais em termos de estrutura pelo que tem que haver um mecanismo de controlo de aceitação e de confirmação. Existem dois tipos de leilões, o English Auction e o Blind Auction, para os quais se podem lançar ofertas e mediante o tipo de leilão no qual se pretende participar existem pontos a considerar mais especificamente

na forma como circulam as informações, quer seja a identidade como a oferta em si, na medida em que alguns destes conteúdos poderão ter que ser cifrados. Uma outra vertente para além da confidencialidade que tem que ser verificada é a existência de uma garantia de honestidade na medida em que não há utilizadores a serem beneficiados em relação a outros ou que não há enganos. Para tal, o Auction Repository pode ser acedido publicamente de tal forma que se consigam saber os leilões e as suas propostas tenha ou não o leilão chegado ao fim, devendo fornecer evidências sobre a sua hosnestidade. Note-se que o Auction Repository não pode ter acesso a qualquer tipo de informação de tal forma que possa permitir agir de forma diferente para determinados clientes. A taxa de ofertas que são enviadas deve ser controlada por um mecanismo denominado cryptopuzzle ou proof-of-work, que constitui quase como que um desafio ao qual o cliente tem que responder para prosseguir.

O Auction Repository como já foi referido contém uma lista de leilões. Introduz-se neste sentido o conceito de BlockChain em que a sua implementação será descrita de forma mais aprofundada no decorrer da elaboração do relatório. Blockchain é uma sequência de blocos em que cada bloco é adicionado e não pode ser alterado, isto é, são imutáveis tanto a nível de conteúdo como na ordem pela qual são inseridos. De uma forma geral, constitui uma série de blocos encadeados em que o primeiro bloco tem o conteúdo e uma hash e o bloco a seguir tem o conteúdo do segundo bloco mais a hash do anterior, e assim sucessivamente de tal forma que o bloco seguinte resulta sempre do conteúdo dele próprio mais a hash do anterior. A função de hash vai assinar o bloco, o que permite que a informação não seja alterada e como o bloco seguinte vai ter o hash do anterior cria uma espécie de selo.

Quando um utilizador pretende usar os serviços do sistema deve autenticar-se com o cartão de cidadão. Este smartcard tem diversos fins como guardar informação privada que pode usar mas que não pode ser partilhada nem conhecida pelo próprio. Esta informação privada relaciona-se com as chaves criptográficas sendo que possui uma chave assimétrica de autenticação do titular, uma chave privada de uma par de chaves assimétricas RSA que serve para autenticar o cliente e uma chave privada de uma pare de chaves assimétricas RSA que serve para produzir assinaturas digitais do titular. Tem também como finalidade, para efetuar operações criptográficas, o uso destas chaves que fazem parte da sua informação privada. No contexto do projeto realizado, a chave simétrica do cartão de cidadão do utilizador não será usada, sendo que o cliente irá usar o cartão de cidadão para autenticação quer seja para criar leilões ou terminar ou para fornecer ofertas.

No presente relatório irá proceder-se à explicação aprofundada sobre a implementação do projeto com breve explicação sobre o código entregue, descrição pormenorizada de todas as decisões tomadas bem como uma justificação para cada uma delas. Serão ainda apresentados diagramas que demonstram de forma direta o funcionamento, nomeadamente ao nível de trocas de mensagens com evidência e demostração do que é transmitidos de forma cifrada ou não, e se os dados são assinados.

2 Tipos de leilões

Para se entender o funcinamento do sistema começou por se definir os dois tipos de leilões e restrições de cada um deles. Como tal, na criação de um leilão o seu criador pode definir o tipo que pretende, sendo que tem à disponibilidade o English Auction e o Blind Auction.

O English Auction tem como caracteristica o facto de que a identidade de quem fez a oferta tem que estar cifrada mas a oferta em si encontra-se na forma de texto claro (sem estar cifrado).

O Blind Auction diferencia-se deste, visto que o valor da oferta está sempre cifrado mas a identidade pode ou não ser exposta, sendo que isto é decisao de quem cria o leilão e não do utilizador.

Assim, tem que haver formas de controlar a identidade dos autores das ofertas ou o seu anonimato, mas note-se que no final do leilão são reveladas pelo Auction Manager todas as identidades sejam ou não anónimas.

3 Auction Manager

Este servidor surge no contexto do projeto com funções que de forma resumida se identificam e que ao longo do documento se encontram mais detalhadas.

- Criar/Terminar um leilão: Cliente pode iniciar comunicação para a realização destas tarefas.
- Validação de requisitos impostos pelo criador do leilão para as ofertas
- Armazena One-Time-Password e valida as ofertas
- Armazena e executa o código dinâmico

Quando um utilizador pretende criar um leilão pode impor algumas restrições relacionadas com as ofertas, tal como restringir o número de ofertas por utilizador. Mais tarde, para uma dada oferta é necessário verificar se esta cumpre os requisitos estabelecidos segundo o leilão a que pertence. Quem trata desta verificação é este servidor, através do código dinâmico.

De forma a aumentar a segurança existem campos da mensagens que são cifrados, podendo estes ser a identidade e/ou valor da oferta dependendo do tipo de leilão. Para garantir que o Auction Repository não adultera a sequência de ofertas, então os campos sensíveis são cifrados com uma One-Time-Password que por sua vez é cifrada com a chave pública do Auction Manager. Por este motivo, o Auction Manager é responsável pela validação das ofertas, no tipo English para garantir que os valores são crescentes e caso exista código dinâmico verificar que a oferta é válida.

4 Auction Repository

Tal como o nome indica, este servidor tem como principal função o armazenamento de leilões e respetivas ofertas, mantendo uma lista.

Cada leilão é identificado por um nome, número único (ID), tempo limite para aceitação de novas bids e uma descrição. Isto é implementado com BlockChain em que se tem uma oferta por bloco.

O Auction Repository fecha um leilão quando houver um pedido para tal ou se terminar o tempo limite da sua duração.

Os clientes comunicam diretamente com este servidor para o fornecimento de ofertas sendo que quando é lançada um nova oferta, antes de ser aceite o cliente terá que responder a um Cryptopuzzle. Este mecanismo de geração de Cryptopuzzles encontra-se no Auction Repository visto que serve relacionado com as ofertas.

4.1 Blockchain

Como já mencionado o Blockchain é uma sequência de blocos na qual a informação em cada um deles não é mudada, visto estar assinada, e a ordem pela qual aparecem também não é alterada. Funciona quase como que uma base de dados no sentido em que armazena dados mas neste caso de forma segura. Por forma segura entenda-se que se pode verificar se houve algum tipo de infração sobre a cadeia, isto é se foi violada. Quando é inserido um bloco com uma oferta no meio (entre outros blocos) da cadeia, todos os blocos à sua frente são violados.

No contexto do projeto implementou-se um sistema Blockchain, seguindo a ideia básica do mesmo, visto que se pretende garantir a integridade das ofertas e permitir ao cliente uma validação das mesmas garantido o correto funcionamento do processo. Assim, por blocos tem-se uma oferta dada por um utilizador para um determinado leilão.

4.1.1 Implementação

Na Figura 1 encontra-se de forma esquematizada o esquema básico do Blockchain adotado.

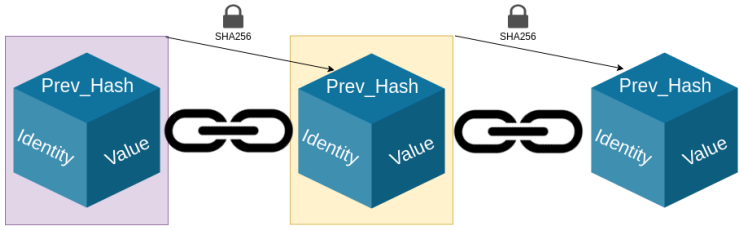

Figura 1: Funcionamento do Blockchain

O Blockchain foi implementado sobre SQLite3 [1] e segue a estrutura da seguinte tabela (Tabela 1):

Colunas	Tipos
auction_id	**INTEGER (PK/FK)**
sequence	**INTEGER (PK)**
prev_hash	TEXT
identity	TEXT
value	TEXT

Tabela 1: Tabela para armazenar as ofertas

Os detalhes mais especificos sobre a criação de ofertas serão mais especificados na secção 8 do presente documento.

Inicia-se o processo, pela criação de uma seed que será usada como prev_hash do primeiro bloco, isto é, é a hash para o primeiro bloco com uma oferta. Nos blocos seguintes, portanto para as ofertas seguintes, o valor do prev_hash será o resultado de um hash da oferta anterior. Estas hashes resultam do algoritmo SHA-256.

Assim, quando um cliente solicita uma listagem das ofertas fornecidas, esta é dada de forma ordenada e com o valor prev_hash presente, de forma a que o cliente possa validar a integridade de todas as ofertas.

4.2 Cryptopuzzle

O cryptopuzzle surge neste contexto como forma de haver algum controlo sobre o fluxo de novas ofertas. Este mecanismo procede ao desenvolvimento de um puzzle ao qual o cliente tem que responder usando brute force para encontrar a solução.

O puzzle é baseado na identidade, que pode estar cifrada dependendo do tipo de leilão, fornecida para a oferta com vista a garantir que o autor do pedido de um cryptopuzzle é o mesmo que apresenta a solução. A identidade que for usada para pedir o cryptopuzzle tem que ser a mesma que será usado depois na oferta o que cria um impedimento para que alguém realize os cryptopuzzles de outra pessoa.

4.2.1 Implementação

Com o algoritmo SHA-256, a identidade é inicialmente hashed e de seguida é gerado um random de 32 bytes que será a futura solução do puzzle. Este random é passado para um XOR junto com a identidade hashed. O resultado do XOR é hashed resultando no puzzle.

Para facilitar a resolução do puzzle, é também fornecido ao cliente um Starts With que representa entre 0% a 70% dos primeiros bytes da solução e o Ends

[1] https://www.sqlite.org/index.html

With que representa o resto dos bytes, ficando apenas dois bytes entre o Starts With e o Ends With.

Na figura Figura 2 encontra-se representado o funcionamento do Cryptopuzzle.

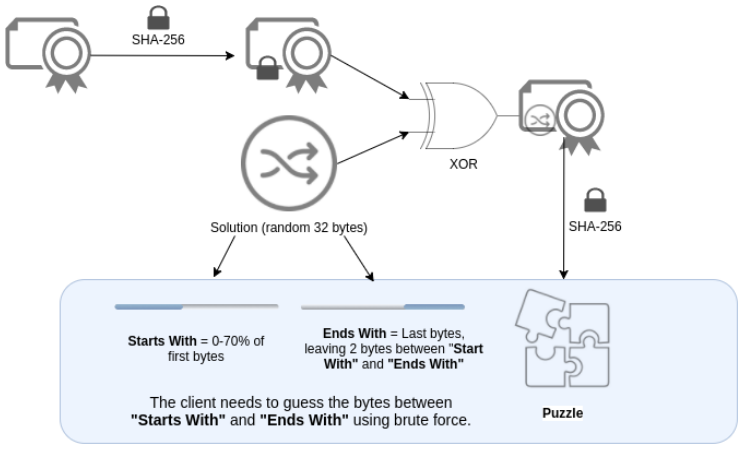

Figura 2: Funcionamento do Cryptopuzzle

Com isto, o cliente tem que usar brute force para encontrar esses dois bytes entre Starts With e Ends With visto que a tranformação resulta no puzzle dado. Dependendo do hardware do cliente, a resolução demora entre 0,5 a 3 segundos para efetuar o processo.

5 Auction Client

O Auction CLient é uma aplicação à qual um utilizador acede para interagir com o sistema de forma a poder usar os recursos disponibilizados pelo mesmo.

O utilizador necessita de apresentar o cartão de cidadão para poder interagir com esta aplicação de maneira a poder autenticar-se no sistema.

Para cada oferta adicionada a um leilão, o Auction Client precisa de armazenar um recibo numa memória não volátil, para posterior validação. Esta parte é muito importante para prevenir que ocorra problemas nos dois servidores alterando a sequência das ofertas no leilão. O recibo terá ainda mais funcionalidades que serão detalhadas mais à frente no documento.

5.1 Cartão de Cidadão

O uso do cartão de cidadão fornece inúmeros padrões de segurança garantindo que não se podem obter os segredos nele armazenados nem por métodos diretos nem indiretos, disponibilizado assim uma segurança para todos os envolvidos numa comunicação.

A proteção proporcionada pelo uso deste smartcard é vital a utilização segura dos pares de chaves assimétricas. As chaves privadas nele guardadas são únicas e até mesmo geradas dentro do próprio smartcard o que as torna ainda mais confidenciais e protegidas. Partindo do principio que mantém os pressupostos de imunidade das cifras RSA e assumindo que os PIN correspondentes não foram partilhados então quando estas chaves forem usadas existe uma garantia de que se está perante o titular do cartão apresentado.

O uso do cartão de cidadão requer a existência de middleware, que neste caso pode ser de dois tipos, e que as aplicações estejam preparadas para o uso deste middleware. Escolheu-se o tipo de middleware que permite usar o cartão de cidadão como um dispositivo criptográfico, sendo disponibilizadas as bibliotecas PKCS11, com um subconjunto da interface Cryptoki, para diversos sistemas operativos.

Considerando o acréscimo de segurança que o uso do smartcard disponibiliza e atendendo às características do projeto implementado, faz todo o sentido e existe a necessidade do seu uso para os utilizadores se autenticarem bem como para realizar assinaturas.

5.1.1 Implementação

Para o uso do cartão de cidadão é usado o package python-pkcs11 de forma a usar o middleware que permite comunicação com o smartcard.

Criou-se um package denominado CartaoDeCidadao e tem como usabilidade a extração tanto do certificado como da identidade ou da chave pública do utilizador. É também usado para realizar assinaturas com a chave privada devido ao facto de esta não poder ser extraída. O cartão de cidadão apenas permite o uso das funções de sintese SHA-256 ou a SHA-1 pelo que usou-se

a SHA-256, por ser mais seguro em relação à SHA-1, para a realização das assinaturas.

6 Recibos

Os recibos são, por norma, usados para provar ou verificar uma dada situação. Neste caso, a considerou-se necessária a sua utilização para que quando o cliente participa num leilão, fornecendo ofertas, possa comprovar que de facto a sua oferta foi aceite e se porventura desconfiar de alguma situação que possa estar a ocorrer ou por qualquer outro motivo queira confirmar se o processo esta a decorrer de forma correta pode solicitar uma listagem das ofertas fornecidas para um leilão e com o seu recibo que contém um número poderá identificar se tudo esta ou não correto. Note-se que os mecanismos de segurança implementados no sistema são capazes de impedir situações indesejáveis, pelo menos aquelas que foram estabelecidas, mas a qualquer momento o cliente poderá verificar essa realidade. Se por caso, algo falhar o cliente poderá também reportar a ocorrência e assim poderão haver melhorias ao sistema constituindo também uma mais valia para melhoramentos ao nível da segurança.

6.1 Armazenamento de recibos

Os recibos podem conter informação sensível pelo que tem que ser protegida de forma a que não seja acessível por terceiros, impedindo que a privacidade do cliente seja exposta.

Para além desta preocupação ao nível da privacidade do cliente, é necessário considerar também que se houver informação na oferta que tem que estar cifrada a chave usada terá de ser armazenada.

Quando se trata da primeira vez que o utilizador armazena um recibo, o Auction Client irá gerar um ficheiro .pwd com random de 128 bytes. Este valor random será usado para gerar a password do utilizador e a sua geração é feita pela assinatura da mesma com o cartão de cidadão do utilizador e posteriormente o resultado da hash é adicionado a um salt, sendo este salt uma porção da chave pública do cartão de cidadão.

Para armazenar um recibo, primeiro é gerado um HMAC para mais tarde ser possível garantir a integridade do recibo. De seguida, a concatenação do recibo com o HMAC é cifrada com o algoritmo CBC, e o resultado obtido é armazenado num ficheiro.

O armazenamento de recibos encontra-se ilustrado na Figura 3.

Para decifrar um recibo é necessário também gerar a password que é usada para decifrar o conteúdo do ficheiro. É, depois, verificada a integridade do conteúdo com o HMAC.

O ReceiptManager tem também uma função para validar um recibo sendo que isto processa-se verificando as várias assinaturas nelas incluídas (Auction Manager/ Auction Repository/Client).

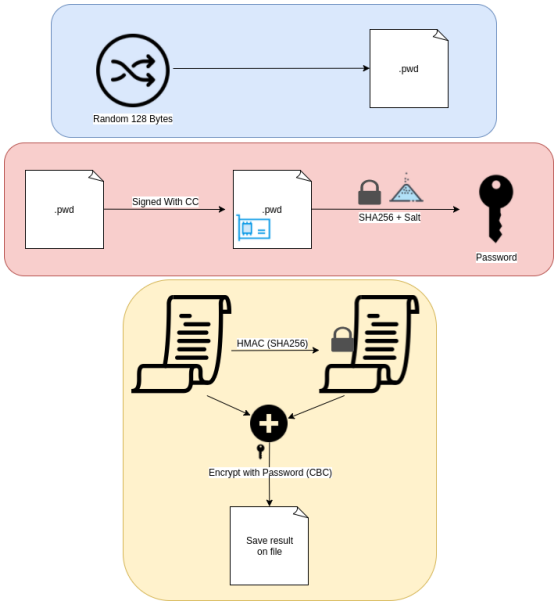

Figura 3: Funcionamento dos Recibos

7 Canais de comunicação

Entre os diferentes componentes que formam o sistema estabeceram-se três canais de comunicação (Figura 4) necessários para a troca de mensagem pelo que se encontram representados no seguinte esquema e no qual se introduziram números para mais facilmente explicar a funcionalidade de cada sistema.

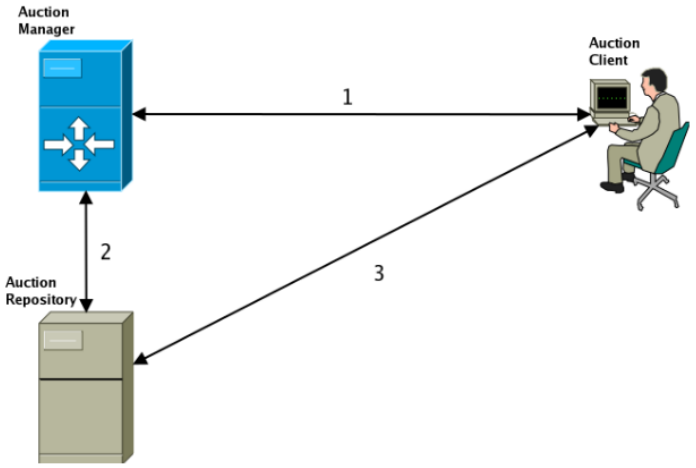

Figura 4: Canais de comunicação

1. Canal Auction Client ↔ Auction Manager
   - Criação e Terminação de leilões ativos
2. Canal Auction Manager ↔ Auction Repository
   - Criação e remoção de instâncias de leilões após pedido de utilizador
   - Validação dinâmica de ofertas
   - Revelação das ofertas
3. Canal Auction Repository ↔ Auction Client
   - Leitura de leilões
   - Criação de ofertas
   - Reclamar prémio

8 Leilões

8.1 Criação de um leilão

A criação de um leilão ocorre entre o cliente e o Auction Manager de tal forma que o cliente interage com o Auction Manager para tal ação (Figura 5). Assim ocorre uma troca de mensagens apresentada no seguinte esquema e apresenta-se também uma explicação para a forma como se processa esta comunicação.

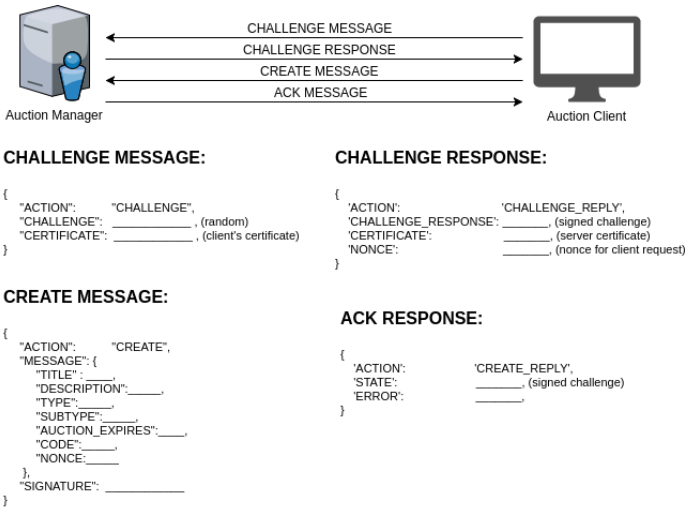

Figura 5: Criação de um leilão: Canal Auction Client ↔ Auction Manager

O utilizador pode interargir com o Auction Manager de forma a requerer a criação de um leilão. A comunicação é inicializada pelo utilizador sendo que para tal, este envia um challenge (random 64 bytes) e o seu certificado através do Cartão de Cidadão para o servidor, neste caso o Auction Manager.

O servidor responde ao challenge enviando o mesmo mas assinado com a sua chave privada e juntamente envia o seu certificado para que o utilizador o possa validar. Na mesma mensagem, o servidor envia também um nonce (random de 16 bytes) ao cliente. A razão pela qual é enviado este nonce é para tornar o request único.

Após esta troca de mensagens o utilizador envia o seu pedido de criação de leilão juntamente com o nonce fornecido. De forma a garantir a integridade dos dados, esta mensagem constituida pelo pedido de criação de leilão e o nonce vai assinada pelo cartão de cidadão garantindo assim a não modificação dos dados e mantendo a garantia de a comunicação estar a ser realizada pelo utilizador correto, isto é com o mesmo que estabeleceu a troca de mensagens anteriores. Esta assinatura é feita com mecanismos RSA SHA256. Como resposta, o servidor devolve feedback com um OK ou NOT OK.

Após a criação do leilão, o Auction Manager comunica com o Auction Repo-

sitory com o objetivo de solicitar o armazenamento do leilão na base de dados (Figura 6).

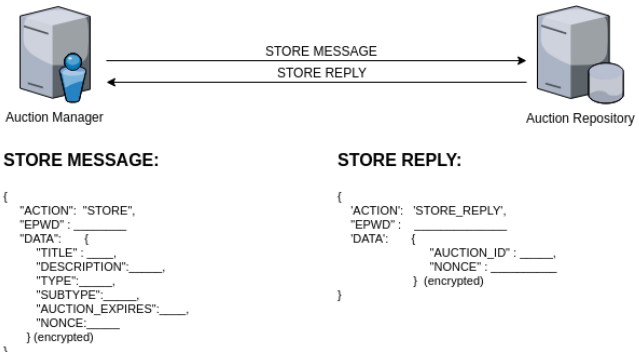

Figura 6: Armazenamento de um leilão: Canal Auction Manager ↔ Auction Repository

Esta operação é suportada pelas seguintes tabelas:

Colunas	Tipos
cc	**TEXT (PK)**
auction_id	**INTEGER (PK)**

(a) Tabela para mapear leilões a utilizadores (Auction Manager)

Colunas	Tipos
auction_id	**INTEGER (PK/FK)**
code	TEXT

(b) Tabela para armazenar o código dinamico(Auction Manager)

Colunas	Tipos
id	**INTEGER (PK)**
title	TEXT
desc	TEXT
type	INTEGER
subtype	INTEGER
duration	INTEGER
start	TIMESTAMP
stop	TIMESTAMP
seed	TEXT
open	INTEGER (1)
claimed	INTEGER (0)

Tabela 3: Tabela para armazenar os leilões (Auction Repository)

8.2 Terminar um leilão

Este procedimento para comunicação é bastante semelhante à criação de uma leilão, seguindo uma execução bastante parecida (Figura 7 e Figura 8).

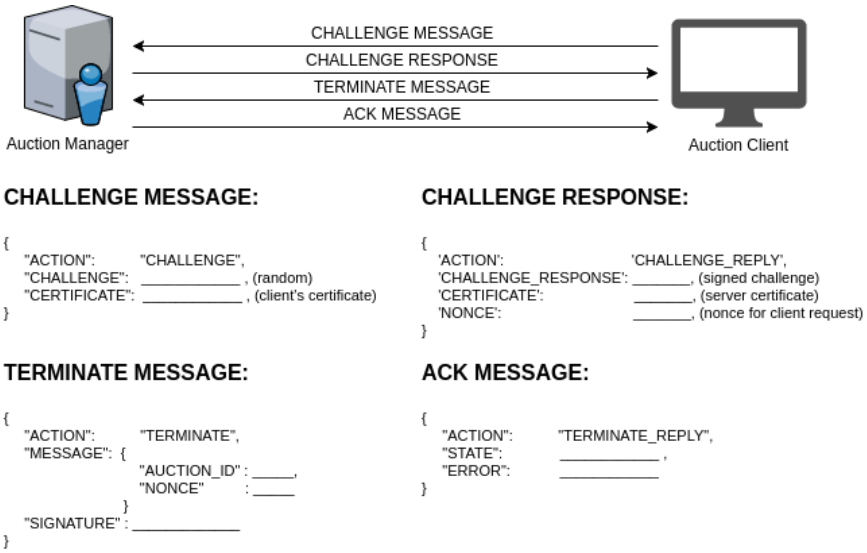

Figura 7: Terminar um leilão (lado do cliente): Canal Auction Client ↔ Auction Manager

Começa por haver uma autenticação do servidor por parte do cliente, com o envio de um challenge.

Como resposta, o servidor responde ao challenge e envia um nonce para o cliente usar. Sendo que de seguida o cliente envia uma mensagem para terminar, TERMINATE MESSAGE, onde se encontra o auction ID e o nonce. Estes elementos são transmitidos assinados pelo cartão de cidadão do cliente. O servidor responde com um ACK MESSAGE com o feedback da ação, terminando assim a comunicação.

O Auction Manager requisita a terminação do leilão ao Auction Repository, tal como esquematizado na Figura 8. O Auction Repository muda o estado de um leilão para fechado e comunica o sucesso da operação ao Auction Manager.

8.3 Listagem de leilões

A qualquer momento um utilizador pode solicitar ao Auction Repository uma lista dos leilões ou informação sobre um dado leilão, e para tal comunica diretamente com o Auction Repository (Figura 9). Esta listagem de leilões não depende do tipo de leilão sendo apenas é identificado por um ID.

De forma a solicitar o pedido mencionado, o utilizador envia uma mensagem a especificar o seu pedido, identificando o tipo de leilão e/ou ID do mesmo, e ainda nesta mensagem envia também um nonce (random de 64 bytes). O nonce neste cenário tem uma utilidade relacionada com autenticidade da resposta do servidor, isto é, o nonce é usado para garantir que a resposta veio do servidor

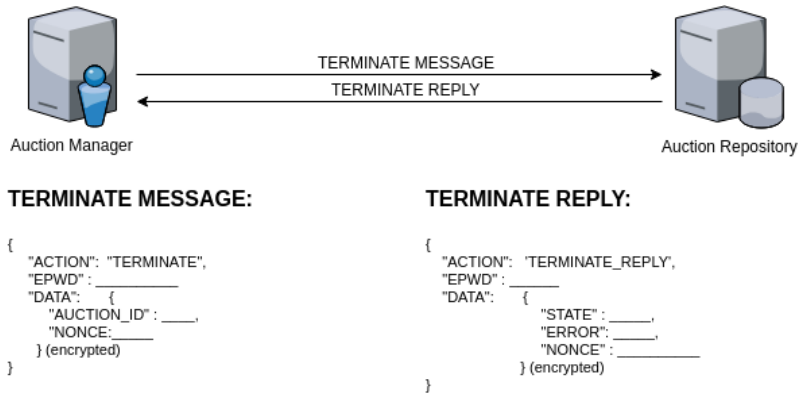

Figura 8: Terminar um leilão (lado do servidor): Canal Auction Manager ↔ Auction Repository

correto visto que sem o nonce o utilizador poderia sujeitar-se a receber uma resposta que não viesse do servidor. Assim, fornece-se uma garantia de que quem enviou a resposta foi o servidor e que portanto a comunicação esta a ser feita com um servidor confiável não havendo portanto riscos acrescidos.

Como resposta ao solicitado, o Auction Repository devolve o conteúdo pedido assinado juntamente com o nonce enviado pelo client. Note-se que se um auction_id for fornecido então o Auction Repository devolve a informação detalhada do mesmo.

8.4 Vencedor do leilão

No final de qualquer leilão, quem ganhou deve ser prosseguir para a fase seguinte de participação no mesmo. Quando um leilão acaba deve ser publicado o resultado, e do ponto de vista do funcionamento do sistema todas as identidades se tornam publicas o que permite que todos tenham conhecimento do que foi apostado e neste caso ja se terá acesso às identidades dos participantes. Surge assim mais uma forma de qualquer pessoa comprovar a honestidade do processo e garantir que tudo decorreu dentro das conformidades. De seguida, procede-se à explicação pormenorizada do processo de tal situação ao nível do sistema (Figura 10 e Figura 11).

Quando o leilão termina, o cliente vencedor pode então apresentar-se e mostrar um comprovativo em como é de facto o vencedor mostrando o seu recibo. Para tal, deve autenticar-se através de um challenge.

O servidor responde ao challenge e envia um nonce. Em resposta, o cliente envia um RECLAIM MESSAGE com o recibo assinado e nonce.

Após recebido o recibo, o servidor compara-o com a oferta vencedora seguindo os passos de verificação seguintes:

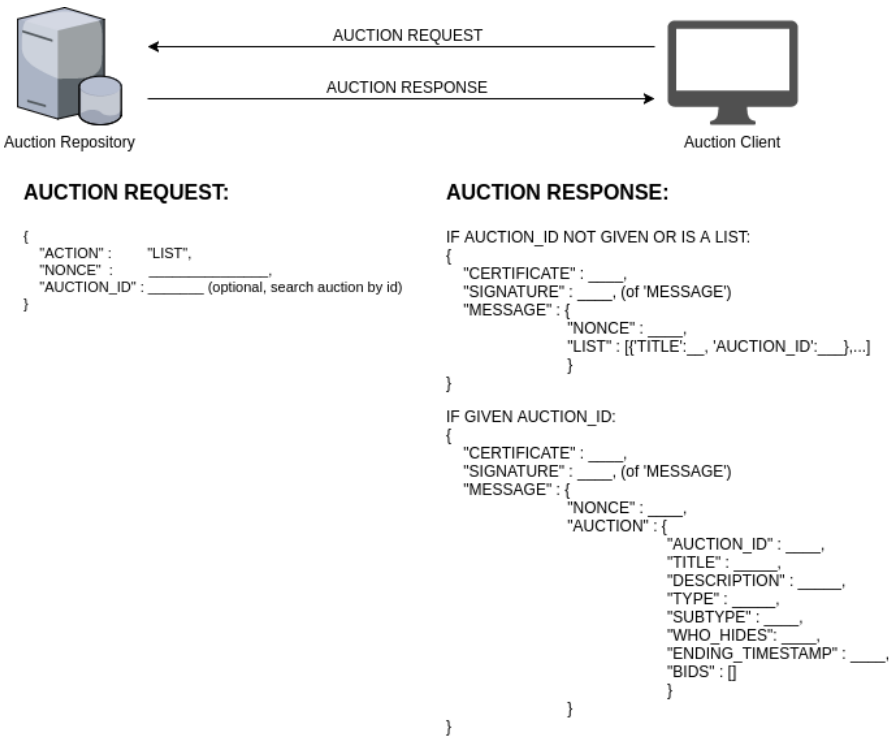

Figura 9: Listagem de leilões: Canal Auction Repository ↔ Auction Client

1. Começa por verificar a assinatura da mensagem;
2. Verifica se o recibo é válido verificando as três assinaturas;
3. Verifica se os valores presentes no recibo são os mesmos presentes no blockchain;
4. A identidade presente no certificado tem de ser a mesma que foi usada na oferta

Por fim, se todas estas verificações sem confirmarem e portanto, se estiver tudo correto, é então devolvida uma mensagem de sucesso para o cliente, confirmando-se de que se trato do verdadeiro vencedor.

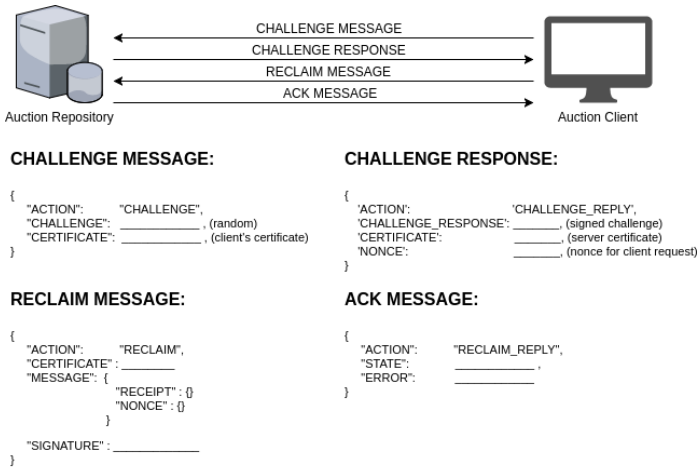

Figura 10: Vencedor do leilão: Canal Auction Client ↔ Auction Repository

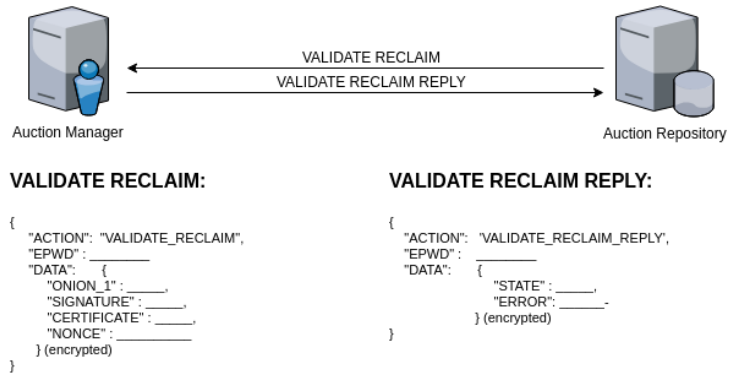

Figura 11: Validação do recibo: Canal Auction Repository ↔ Auction Manager

9 Ofertas

9.1 Criação de uma oferta

Quando existem algum leilão ainda a decorrer e o um dado cliente pretende participar comunica diretamente com o Auction Repository, seguindo o esquema de troca de mensagens seguinte (Figura 12).

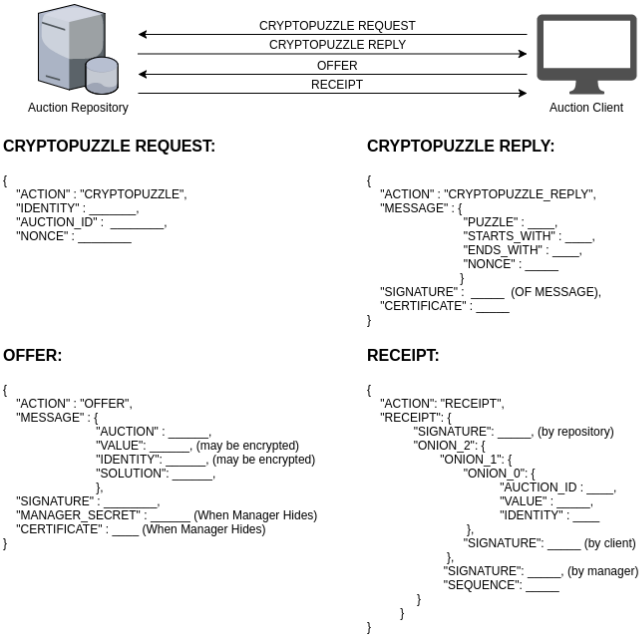

Figura 12: Criação de uma oferta: Canal Auction Client ↔ Auction Repository

Quando o cliente pretende realizar uma oferta inicia a comunicação com o Auction Repository de forma a requisitar um cryptopuzzle. Para tal, envia a sua identidade e o ID do leilão em que pretende participar. É importante referir um aspeto importante referente à forma como a identidade é enviada dependendo do tipo de leilão. No caso em que a identidade está escondida então esta tem que estar cifrada e terá que ser igual à usada na oferta em si.

Com base na identidade do cliente, o cryptopuzzle é gerado e enviado ao cliente. Para garantir integridade e autenticidade, toda a informação do puzzle está assinada pela chave privada do repositório. Havendo uma garantia de que a informação do mesmo não é alterada.

Quando o cliente recebe o cryptopuzzle, resolve-o e envia a solução juntamente com a oferta a realizar para o Auction Repository. Dependendo das especificações do leilão, a identidade do cliente e/ou valor da oferta podem ser transmitidos de forma cifrada. A chave usada para tal terá que ser enviada na

oferta de forma cifrada com a chave pública do manager para não ser revelada ao repositório, mas esta situação depende novamente do tipo de leilão. Desta forma, apenas o Auction Manager pode validar uma oferta (Figura 13), como mencionado na Seção 10.

O Auction Repository envia um recibo assinado por todas as entidades. E por fim, o cliente valida todas as assinaturas e armazena numa memória não volátil com o ReceiptManager.

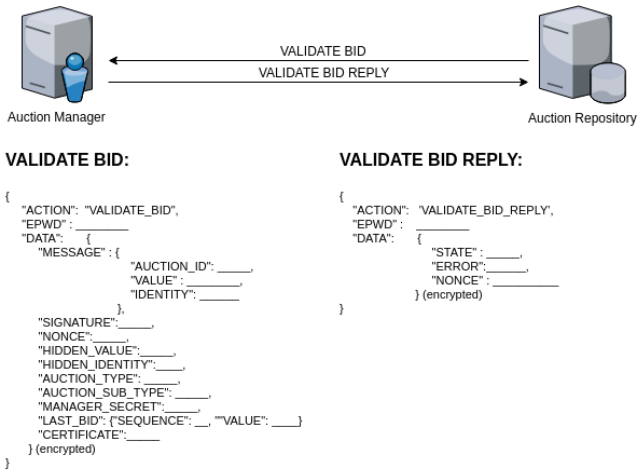

Figura 13: Validação de uma oferta: Canal Auction Repository ↔ Auction Manager

Esta operação é suportada pelas seguintes tabelas:

Colunas	Tipos
auction_id	**INTEGER (PK/FK)**
sequence	**INTEGER (PK)**
identity	TEXT
secret	TEXT

(a) Tabela para armazenar as chaves dos clientes (Auction Manager)

Colunas	Tipos
auction_id	**INTEGER (PK/FK)**
sequence	**INTEGER (PK)**
prev_hash	TEXT
identity	TEXT
value	TEXT

(b) Tabela para armazenar as ofertas (Auction Repository)

9.2 Divulgação das ofertas

Sempre que um leilão é terminado, quer seja por terminar o tempo da duração estabelecida ao início quer seja por ordem do seu autor, o Auction Repository requisita a divulgação das One-Time-Passwords usadas para cifrar as ofertas do leilão em específico (Figura 14). O Auction Manager envia as OTPs correspon-

dentes em seguida o Auction Repository armazena-as numa tabela própria para não alterar a sequência de hashes geradas durante o leilão em sim (Blockchain).

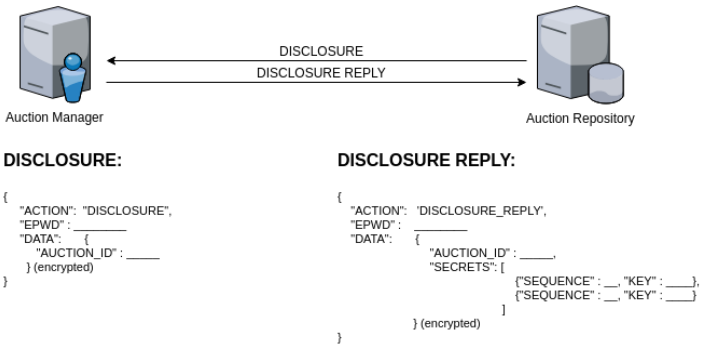

Figura 14: Divulgação das ofertas: Canal Auction Repository ↔ Auction Manager

Esta operação é suportada pelas seguinte tabela:

Colunas	Tipos
auction_id	**INTEGER (PK/FK)**
sequence	**INTEGER (PK/FK)**
secret	TEXT

Tabela 5: Tabela para armazenar as chaves das ofertas (secretas)

Quando o Auction Manager divulga as chaves secretas o Auction Repository procura automaticamente o vencedor do leilão e armazena-o numa tabela concebida para essa tarefa. Desta forma, tornou-se mais eficiente o processo de reclamar o prémio.

Colunas	Tipos
auction_id	**INTEGER (PK/FK)**
sequence	INTEGER (FK)

Tabela 6: Tabela para armazenar os vencedores (Auction Repository)

10 Código dinâmico

O código dinâmico permite que haja uma validação das ofertas sendo essa a sua principal função. Para o código dinâmico é usada o package RestrictedPython que fornece um ambiente restrito para execução de código não confiável. O código dinâmico é testado pela sua syntax e execução antes de ser descarregado para o Auction Manager.

O package RestrictedPython não autoriza o utilizador a realizar qualquer tipo de imports, com vista a impedir o cliente de usar algo que possa pôr em risco o funcionamento correto do servidor.

Para ocorrer a validação de uma oferta, são fornecidos os seguintes argumentos ao código dinâmico:

- identity: número do cartão de cidadão que fornece a oferta
- value: valor dado pelo cliente
- times: tempo em que o cliente participou o leilão.
- prev_value: maior oferta dada para o leilão; este valor é None se for um Blind Auction

O valores fornecidos para o código dinâmico são sempre os decifrados e este terá que devolver True ou False na sua execução que corresponde às situações em que a oferta é válida ou não.

11 Comunicação segura entre servidores

Inicialmente começou por se usar chaves assimétricas para a troca de mensagens entre servidores, quer fosse para a validação, criação, terminação entre outras operações sobre leilões e ofertas. Devidos ás limitações impostas pelo algoritmo RSA sobre a dimensão dos dados sensíveis, alterou-se esta abordagem. No RSA, o tamanho dos dados de entrada está diretamente correlacionado com o tamanho da chave em si. Nesta implementação usaram-se 2048 bits o que faz um total de 256 bytes de tamanho máximo e visto que para as operações mais complexas, como a validação e a divulgação das ofertas, o tamanho de mensagens trocadas excedia os 256 Bytes teve que se proceder a uma alteração da abordagem a seguir. Desta forma, seguiu-se uma abordagem hibrida, que é recomendada pelo mbed TLS [2]. Assim, os dados são cifrados de forma simétrica com uma One-Time-Password (OTP) e a OTP é cifrada de forma assimétrica com o algoritmo RSA. Esta OTP é sempre uma valor de tamanho 32 Bytes e é obtida através de um gerador aleatório do sistema operativo e por isso apresenta uma segurança mais elevada visto ter um ciclo de repetição muito baixo.

Assumindo que ninguém viola o código dos servidores é bastante difícil simular algum dos dois servidores implementados. Considerando como exemplo, alguém que tenta simular o Auction Manager não consegue decifrar os dados visto que os mesmos são cifrados com a chave pública deste servidor, e esta é conhecida, mas para decifrar é necessário a sua chave privada, e esta apenas é conhecida pelo Auction Manager verdadeiro. Neste sentido, é preciso ter acesso à chave privada do servidor que se esta a tentar simular para conseguir decifrar a resposta, o que é uma operação extremamente complicada.

Assim, para o ataque ser bem sucedido é necessário comprometer os dois servidores em simultâneo. Isto não acrescenta vantagem nenhuma ao atacante visto que assim não estabelece comunicação em ponto algum com o sistema verdadeiro. Ao simular um dos servidores o objetivo seria tentar comunicar com um dos verdadeiros e nesse caso poderia conseguir apanhar alguma mensagem mas como visto não iria conseguir decifrar.

11.1 Implementação – cifras por blocos (Package CryptManager)

São usadas cifras por blocos para esconder informação das ofertas e também para cifrar os recibos armazenados. Usou-se o algoritmo CBC devido à sua simplicidade, eficácia e por não ser necessário a ocorrência de acesso aleatório.

O valor do vetor de inciação é sempre aleatório e armazenado no início do texto cifrado.

[2] https://tls.mbed.org/kb/cryptography/rsa-encryption-maximum-data-size

11.2 Implementação – assimétricas (Package CertManager)

Para gerir o uso de chaves assimétricas foi criada o package CertManager.
Este possui funcionalidades tais como:

- Assinar com chave privada (com recurso a um digest SHA256)
- Verificar uma assinatura dando a public key correspondente
- Cifrar com chave pública (algoritmo RSA OAEP)
- Decifrar texto cifrado com uma public key utilizando a private key (RSA OAEP)
- Verificar Certificado importando uma chain of trust incluída dentro da package
- Obter Identidade através do certificado

A verificação de uma mensagem assinada, por exemplo, é feita verificando a assinatura com a chave pública do assinante e depois verificar o certificado da chave pública do mesmo com base na cadeia de certificação presente.

12 Fases de Desenvolvimento

A realização do projeto dividiu-se em várias fases de desenolviemento pelo que são mencionadas na seguinte lista sendo que a sua execução não seguiu necessáriamente esta ordem à excepção do primeiro item referido.

- Começou por se desenvolver os servidores e o cliente sem nenhum mecanismo de segurança
- Produziu-se um documento sobre as especificações de segurança
- Introduziram-se as funcionalidades referentes ao Cartão de Cidadão e o uso de certificados
- Adicionou-se a emissão de recibos para as bids
- Implementaram-se os procedimentos executados pelo Auction Manager para cifrar uma oferta
- Proteção da comunicação entre todas as aplicações
- Adicionou-se o código dinâmico
- Implementaram-se os cryptopuzzles

13 Funcionalidades Implementadas

- Mecanismos de proteção tais como criptografia, autenticação, assinaturas nas mensagens trocadas
- Proteção das ofertas até ao final do leilão às quais correspondem
- Identificação do autor da oferta através do Cartão de Cidadão
- Revelação das propostas no final do leilão
- Validação das ofertas através do código dinâmico
- Modificação de ofertas validadas pelo código dinâmico
- Construção de um Blockchain por leilão
- Desenvolvimento de cryptopuzzles
- Produção e validação de recibos das ofertas
- Validação de um leilão já fechado

14 Conclusão

A concretização deste projeto serviu para enquandrar muitos dos conceitos aprendidos ao longo do semestre. Embora esta seja uma consideração bastante espectável depois da realização de um projeto no âmbito de uma unidade curricular, neste caso deve destacar-se um importante ponto que se refere ao poder de decisão que muitas vezes foi dado. A abrangência do projeto e o facto de haver uma liberdade considerável para a implementação, bem como nas decisões tomadas para a vertente de segurança levou a que o projeto se tornasse ainda mais interessante.

O levantamento em si dos requisitos de segurança tornou-se um desafio bastante prático e útil visto que leva a pensar em muitos casos e situações que podem acontecer. A própria implementação levou muitas vezes a conversas entre os elementos do grupo bastante produtivas e uma partilha de conhecimentos bastante elevada. A partilha de informação extra em relação ao que foi dado nas aulas, entre os elementos do grupo constituiu também uma mais valia no sentido em que se obteve muitos mais conhecimentos e pontos de utilização de muito o que foi dado nas aulas mas que com o trabalho prático se atingiu esse conhecimento mais rápido.

Neste enquadramento, a realização deste projeto tornou-se uma ótima forma de solificar conhecimentos adquiridos bem como adquirir muitos outros.

15 Bibliografia e Material de apoio

- Segurança em redes informáticas, 5ªedição, André Zúquete
- https://python-pkcs11.readthedocs.io/en/latest/index.html
- https://pypi.org/project/RestrictedPython/4.0b4/