Blockchain-based auction management

Departamento de Electrónica, Telecomunicações e Informática

Segurança

P2G15

Miguel Maia - 76434 maiamiguel@ua.pt

Armando Sousa - 76498 ammsousa@ua.pt

Fevereiro 2019

Conteúdo

1 Introdução **3**

2 Considerações gerais **4**
2.1 Estrutura das mensagens 4
2.2 Funcionamento da aplicação 4
2.3 Autenticação, integridade e confidencialidade 5
2.3.1 Assinaturas 5
2.3.2 Integridade 5
2.3.3 Confidencialidade 5
2.4 Garantia de honestidade 5
2.5 Identificação do autor das apostas 6
2.6 BlockChain 6
2.7 Componente pública dos servidores 6
2.8 Validação do certificado com a cadeia de certificação 7
2.8.1 Lado do cliente 7
2.8.2 Lado do servidor 7

3 Funcionalidades **8**
3.1 Criar um leilão 8
3.2 Terminar um leilão 8
3.3 Listar leilões abertos 9
3.4 Listar leilões terminados 9
3.5 Envio de uma aposta 10
3.6 Mostrar todas as apostas de um leilão 11
3.7 Mostrar todas as apostas de um cliente 11
3.8 Validar um leilão terminado 12
3.9 Decifrar uma aposta 12
3.10 Validar um recibo e consequentemente uma aposta 13
3.11 Verificar resultado de um leilão 14

4 Código dinâmico **15**
4.1 Objectivo 15
4.2 Método eval() 15

4.3 Segurança com o método eval() 15

5 Recibos **17**
5.1 Produção do recibo 17
5.2 Validação do recibo 18

6 Cryptopuzzles **19**
6.1 HashCash 19

7 Problemas **21**

8 Conclusões **22**

Capítulo 1

Introdução

Este relatório tem como objectivo descrever o trabalho prático desenvolvido no âmbito da unidade curricular de Segurança, bem como explicar as medidas implementadas e o seu porquê para garantir que este cumpre todos os requisitos necessários. O objectivo deste trabalho era desenvolver um sistema que permita aos utilizadores a criação e participação segura em leilões. Para tal, o sistema é composto pelo gestor de um leilão, um **Repositório** de leilões e correspondentes apostas e um ou mais clientes. Os requisitos descritos como essenciais seriam:

- Autenticação, integridade e confidencialidade das apostas.
- Controlo das apostas e garantia que as mesmas cumprem certos requisitos para puderem ser aceites.
- As apostas devem ser identificados através do cartão de cidadão. No entanto, devem ser anónimas até ao fim de um leilão.
- O repositório não deve ter acesso a nenhuns dados que lhe permita agir de forma diferente consoante o autor da aposta. Deve haver garantias da sua honestidade.

O relatório começa por descrever o sistema de leilões criado bem como as suas funcionalidades. De seguida fala-se dos mecanismos de autenticidade e integridade usados em geral pelo projecto, seguido de uma descrição mais detalhada e individual das técnicas usadas em cada funcionalidade no projecto.

Os mecanismos de segurança a serem aplicados variam consoante as operações realizadas entre os clientes e os servidores. Informação de carácter público não necessita dos mesmos cuidados que conteúdos sensíveis.

Capítulo 2

Considerações gerais

2.1 Estrutura das mensagens

```
{
  "payload" : {
    "message"      : {
      "typeMSG"       : 'listAuction',
      "genericfields" : Campos extra que podem ser adicionados consoante as
          necessidades,
      "certificate"   : certificado do cliente,
      "chain"         : cadeia do certificado do cliente
    },
    "signature"    : <Mensagem assinada com a chave privada do cartao de cidadao
        do cliente>
  }
}
```

Listing 2.1: Descrição geral da estrutura das mensagens JSON trocadas

2.2 Funcionamento da aplicação

Para que a aplicação funcione de forma correcta, deve ser executado o **Repositório** e o **Manager** antes do cliente.

O projecto foi desenvolvido em python3 e faz uso das seguintes bibliotecas:

- PyKCS11 versão 1.5.3
- cryptography versão 2.4.1
- termcolor 1.1.0
- datetime 4.3
- pycrypto 2.6.1
- pyopenssl 18.0.0

2.3 Autenticação, integridade e confidencialidade

2.3.1 Assinaturas

Todas as mensagens trocadas entre os vários intervenientes, sejam eles quem forem, são sempre assinadas para garantir que se está a comunicar com os intervenientes correctos e não com um impostor. Um dos campos da mensagem é o certificado, no caso do cliente, que é anexado à mensagem bem como a respectiva cadeia de certificação para que o **Manager** e o **Repositório** possam verificar a veracidade e validade do seu certificado.

2.3.2 Integridade

As assinaturas permitem também garantir que o conteúdo das mensagens não é alterado, pois bastaria alterar algo para que a assinatura não fosse válida. Desta forma, há garantia de integridade das mensagens.

2.3.3 Confidencialidade

Para garantir a confidencialidade de certos campos, quer seja da identidade de um apostador ou do valor de uma aposta, foi usada uma cifra simétrica AES. Este algoritmo foi o escolhido pois garante um bom compromisso entre desempenho e segurança. Quando se pretende ocultar o valor duma aposta, cifra-se esse campo da aposta com uma chave simétrica gerada, bem como um dado IV gerado para cada **cliente**. Essa chave usada para ocultar esse campo é guardada pelo **cliente** e não é enviada para os servidores. Somente, no momento em que um **cliente** pretende tornar a sua aposta pública é que envia essa chave simétrica usada para o **Manager** para que este possa decifrar a sua aposta.

No caso em que a identidade é cifrada, essa chave é anexada no momento em que um cliente faz uma aposta e enviada para o **Manager** para que este possa decifrar o certificado do cliente e fazer a validação da assinatura da mensagem.

2.4 Garantia de honestidade

Como no caso em que a identidade deve ser escondida, o certificado do cliente é cifrado e a chave não é enviada, então é impossível para o **Repositório** a decifra desse campo, o que faz com que este não saiba quem fez a aposta, não permitindo assim comportamentos diferentes consoante a identidade de quem apostou.

Cada vez que um cliente faz uma aposta é-lhe emitido um recibo, pelo que caso haja fraude por parte de alguma das entidades, o cliente pode aperceber-se desse facto devido ao seu recibo. É impossível para o **Repositório** alterar

os valores das apostas, pois estas estão guardadas na blockchain o que as torna imutáveis, devido à estrutura da mesma e às hashes de cada bloco.

O **Manager** não tem acesso às apostas que se encontram na blockchain pelo que não consegue alterar os valores das mesmas.

2.5 Identificação do autor das apostas

O cartão de cidadão é usado nesta aplicação para a criação de assinaturas digitais em todas as mensagens geradas pelo cliente. Para a identificação do autor das apostas é também usado o cartão de cidadão através do seu certificado que é enviado juntamente na aposta e que contém o nome do utilizador responsável por essa mesma aposta.

2.6 BlockChain

A blockchain é uma estrutura existente em todos os leilões e usada para guardar as apostas de cada cliente. Esta é composta por blocos que são criados e adicionados à mesma sempre que uma nova aposta é feita. O primeiro bloco é criado e adicionado pela próprio **Repositório** quando um novo leilão é criado.

Os blocos são criados fazendo uso da classe Block.py e possuem 6 variáveis diferentes:

- Index - Número único atribuido a cada aposta para as diferenciar
- Author - Onde é guardado o certificado do individuo que criou o leilão
- Bid - O valor da oferta do cliente
- SerialNb - O número de identificação do leilão
- Prev_Hash - A hash do bloco anterior na blockchain
- Hash - A hash do bloco actual feita com o sha256 e com as outras variáveis de bloco.

A criação da hash de um bloco usando a hash do bloco anterior garante que quaisquer alterações posteriores à blockchain pelo **Repositório** com o intuito de fraude são detectáveis pelo simples facto das hash não estarem correctas.

2.7 Componente pública dos servidores

Optou-se por assumir que a componente pública de ambos os servidores, quer do **Manager** como do **Repositório** são bem conhecidas, pelo que desta forma não é necessário o envio de certificados por parte dos servidores.

2.8 Validação do certificado com a cadeia de certificação

2.8.1 Lado do cliente

Como mencionado anteriormente os certificados dos clientes são enviados em todas as mensagens que estes enviam para os servidores. Para garantir que estes certificados são válidos é necessário a cadeia de certificação de cada um. A cadeia de certificação de cada cliente é criada logo no início quando um cliente corre um programa, construindo a cadeia com base nos certificados presentes no seu cartão de cidadão. Este cadeia é depois adicionada a todas as mensagens enviadas para os servidores

2.8.2 Lado do servidor

Após a recepção de uma mensagem o servidor começa por retirar o certificado do cliente da mensagem bem como a cadeia de certificação do mesmo, também presente na mensagem. A cadeia de certificação começa no certificado da entidade emissora e acaba com o certificado de uma entidade raiz.

O servidor começa por verificar se o certificador de um cliente já se encontra na pasta server_trusted_certs, onde são guardados todos os certificados que são confiados pelo servidor. Se o certificado não for encontrado o servidor começa a percorrer a cadeia de certificação até encontrar um certificado em que confia. Num cartão de cidadão nunca usado no sistema, a procura por um certificado confiável normalmente termina quando os servidores chegam aos certificados 'Cartão de Cidadão 00X' , onde X pode ir de 1 até 4, na cadeia de certificação, visto que estes já se encontram por default na pasta. De seguida verifica-se que o certificado não se encontra em nenhuma CRL.

Se não ocorrerem problemas todos os certificados que antes não eram de confiança são adicionados na pasta de certificados confiados pelo servidor e a cadeia de certificação é validada concluindo-se então que o certificado enviado pelo cliente é de confiança.

Capítulo 3

Funcionalidades

3.1 Criar um leilão

Um leilão pode ser do tipo *English* ou *Blind*. No caso de ser do tipo *English*, as suas apostas têm um valor público mas a identidade de quem as fez é cifrada. Cada aposta deve ser superior à anterior.

No caso de ser do tipo *Blind* o valor da aposta é cifrada e a entidade é pública. Assim sendo, é necessário que o cliente na criação de uma leilão especifique o tipo de leilão que pretende, o seu nome, tempo limite de duração, descrição e se pretende usar alguma função de código dinâmico para validações adicionais.

Após a definição destes requisitos para a criação de um leilão, é enviada uma mensagem para o **Manager**, com esses campos que descrevem um leilão. Como já dito anteriormente, e como acontece em todas as mensagens, é assinada e anexado o certificado e cadeia de certificação do cliente para que o **Manager** possa validar a assinatura e o certificado. Uma vez recebida a mensagem no **Manager**, este envia uma mensagem para o **Repositório** com o objectivo desta entidade criar então o leilão.

Sempre que são trocadas mensagens, todos os processos de assinatura, verificação de assinaturas bem como da cadeia de certificação são repetidos.

3.2 Terminar um leilão

Para que seja possível terminar um leilão é apresentada uma lista dos leilões que se encontram ainda a decorrer e o utilizador deve então introduzir o número identificativo do leilão que pretende terminar. Após esta acção, é criada uma mensagem em que é anexado o número identificativo do leilão, o certificado e a cadeia do certificado do utilizador. É também incluída a assinatura do cliente realizada com o cartão de cidadão. Esta mensagem é

enviada para o **Repositório** que após a validação da cadeira de certificação e da assinatura do cliente, termina então o leilão correspondente.

3.3 Listar leilões abertos

O cliente envia uma mensagem ao **Repositório** a fim de este lhe enviar a lista de leilões que ainda se encontram activos. A mensagem enviada tem um tipo que descreve o que deve ser feito, um campo que identifica o tipo de lista que é pedido, o certificado do cliente bem como a respectiva cadeia de certificação para que o **Repositório** possa garantir que o certificado enviado pelo cliente é válido.

```
{
  "payload" : {
    "message"     : {
      "typeMSG"     : 'listAuction',
      "list"        : 'openAuctions',
      "certificate" : certificado do cliente,
      "chain"       : cadeia do certificado do cliente
    },
    "signature"   : <Mensagem assinada com a chave privada do cartao de cidadao
        do cliente>
  }
}
```

Listing 3.1: Mensagem Json criada pelo cliente.

O **Repositório** após receber esta mensagem irá extrair a assinatura da mensagem, obter o certificado e extrair a chave pública para posteriormente validar a assinatura. De seguida reconstrói a assinatura e valida a mesma. Se a mesma for válida, bem com a cadeia de certificação que valida o certificado, então o **Repositório** envia a lista de leilões ainda a decorrer, criando uma mensagem do mesmo género em que envia essa mesma lista, assinada pela sua chave privada.

```
{
  "payload" : {
    "message"     : <Lista de leiloes abertos>
    "signature"   : <Mensagem assinada com a chave privada do Repositorio>
  }
}
```

Listing 3.2: Mensagem Json criada pelo Repositório em que envia a lista dos leilões.

Do lado do cliente, uma vez recebida esta mensagem, o mesmo verifica a assinatura da mensagem que foi feita pelo **Repositório**, para garantir que aquela mensagem foi efectivamente enviada e produzida pelo **Repositório**. De seguida extraí a lista incluída na mensagem e disponibiliza essa informação visualmente.

3.4 Listar leilões terminados

Nesta opção o funcionamento é semelhante ao descrito acima, diferindo apenas na mensagem enviada pelo cliente.

```
{
  "payload" : {
    "message"      : {
      "typeMSG"     : 'listAuction',
      "list"        : 'closedAuctions',
      "certificate" : certificado do cliente,
      "chain"       : cadeia do certificado do cliente
    },
    "signature"    : <Mensagem assinada com a chave privada do cartao de cidadao
        do cliente>
  }
}
```

Listing 3.3: Mensagem Json criada pelo cliente.

Neste caso o parâmetro *list* muda para **closedAuctions** para especificar que se pretende a lista de leilões já terminados.

A resposta do **Repositório** é igual, mudando apenas a filtragem que neste caso pesquisa por leilões já terminados.

3.5 Envio de uma aposta

Para que um cliente possa apostar num dado leilão é necessário primeiramente especificar qual o leilão que quer apostar. Depois de definido este parâmetro, o cliente envia uma mensagem ao **Repositório** de forma a saber de que tipo é o leilão. Na mensagem de resposta, o **Repositório** envia junto com a resposta o número de zeros bem como uma string aleatoriamente gerada para o **cryptopuzzle**. De seguida, o cliente calcula o resultado do cryptopuzzle e envia de novo para o **Repositório**. Este processo encontra-se melhor explicado no capítulo 6. Após a recepção da aposta por parte do **Repositório**, este envia uma mensagem para o **Manager** para validar esta aposta. O **Manager** envia o resultado da verificação da aposta para o **Repositório**. Este, após verificar que a aposta é válida, adiciona a mesma à blockchain do leilão correspondente. Caso não seja válida, a aposta é descartada, e é enviada uma mensagem de erro para o cliente por parte do **Repositório** para informar essa situação. Se a aposta for aceite, é igualmente enviada uma mensagem de sucesso para o cliente.

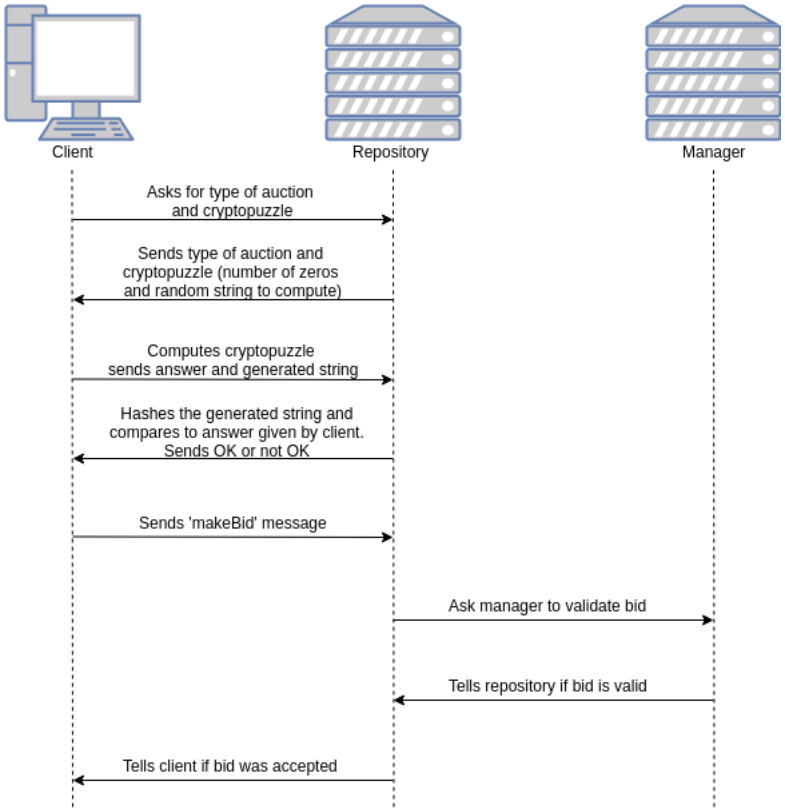


Figura 3.1: Flowchart das mensagens trocadas entre o cliente e servidores quando o cliente faz uma oferta

3.6 Mostrar todas as apostas de um leilão

Nesta opção é pedido ao utilizador para especificar qual o leilão que quer verificar, sendo disponibilizado uma lista de leilões, pelo que o utilizador deve então introduzir o respectivo número que identifica o leilão pretendido. Após o utilizador escolher o leilão que quer ver as apostas, o cliente envia uma mensagem para o **Repositório** seguindo os moldes já descritos. Nesta mensagem é contido o identificador do leilão, o tipo de lista que se pretende receber o certificado e a cadeia de certificação do cliente. Esta mensagem é assinada com o cartão de cidadão e enviada para o **Repositório** que quando recebe a mesma, verifica a assinatura bem como a cadeia de certificação e envia a lista de apostas daquele leilão.

3.7 Mostrar todas as apostas de um cliente

Nesta opção, é necessário recorrer aos recibos dos clientes. Para cada utilizador é gerada uma pasta com o seu nome onde serão guardados os recibos

das apostas que este faz. Assim sendo, é procurada a pasta correspondente ao utilizador actual e todos os recibos existentes na mesma são carregados e a informação referente a essas apostas mostrada. Como cada utilizador tem uma pasta própria não existe a possibilidade de haver confusão entre recibos de utilizadores diferentes.

3.8 Validar um leilão terminado

Esta opção tem como objectivo verificar a integridade de um leilão, ou seja, garantir que as apostas que se encontram guardadas na blockchain não foram alteradas e se encontram de acordo com o original.

Para garantir que não houve alterações a um bloco basta computar o seu **hash** com os valores que se encontram no mesmo e comparar com o campo **hash** desse bloco. Este campo contém o **hash** dos valores correspondentes a esse bloco e a alteração de qualquer destes campos provocaria por consequência uma mudança à **hash** desse bloco que por sua vez iria alterar a **hash** dos blocos seguintes fazendo com que o resto dos blocos também ficassem inválidos. Inicialmente é pedido ao utilizador para especificar o leilão que pretende que seja verificado. É criada uma mensagem e enviada para o **Repositório** seguindo os moldes já descritos.

Do lado do **Repositório**, caso não se encontre esse leilão ou o mesmo não tenha sido ainda terminado é retornado uma lista vazia que é depois recebida pelo cliente e interpretada como um caso de erro, sendo que é impressa uma mensagem de aviso ao cliente. Se de facto o leilão existir é retornado pelo **Repositório** os blocos correspondentes a esse leilão, que contêm as apostas.

Do lado do cliente, é percorrido bloco a bloco e para cada bloco, é calculada a **hash** desse mesmo bloco (usando os campos de index, autor, valor da aposta, número de série da aposta e a **hash** anterior) e comparada com a guardada proveniente no bloco. Se ambas forem iguais, significa que aquele bloco não foi alterado. Caso sejam diferentes houve fraude pelo que toda a blockchain tem que se invalidada.

3.9 Decifrar uma aposta

Para se saber quem ganhou um leilão *Blind* o **Repositório** tem que esperar que as pessoas que fizeram ofertas decifrem os valores. Sem isto acontecer ele não sabe quem é que apostou o maior valor. Um princípio parecido aplica-se aos leilões *English* onde a identidade tem de ser decifrada para se saber quem é que fez a maior aposta. Enquanto que todas as ofertas não são decifradas o **Repositório** calcula o vencedor usando apenas aquelas que foram decifradas.

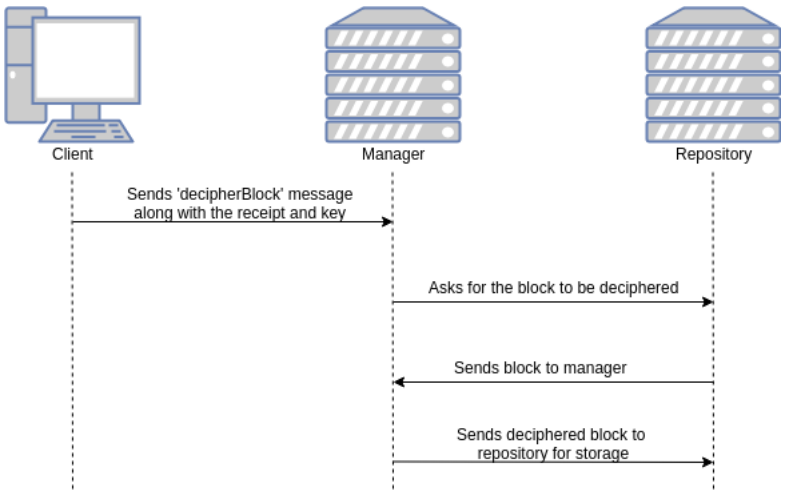


Figura 3.2: Flowchart das mensagens trocadas quando um cliente deseja decifrar uma oferta

Para decifrar as suas ofertas os clientes devem começar por especificar qual o recibo a decifrar indicando o index do bloco e o número do leilão. Este recibo é enviado para o **Manager** juntamente com a chave AES usada para cifrar os campos.

O **Manager** verifica se o recibo é válido e depois pede ao **Repositório** que lhe envie o bloco correspondente ao recibo. Usa a chave enviada pelo cliente para decifrar o campo cifrado no bloco e manda este novo bloco decifrado para o **Repositório** para ser guardado.

3.10 Validar um recibo e consequentemente uma aposta

Neste ponto, é pedido ao utilizador que identifique o recibo que pretende validar. Como os recibos são guardados com o número do leilão e da aposta a que correspondem é fácil carregar os mesmos, desde que o utilizador introduza o número do leilão e da aposta. É possível identificar esse mesmo recibo pois um leilão pode ter várias apostas mas todas têm um número diferente, incremental. A validação do recibo começa com a validação da assinatura do **cliente**, do **Manager** e do **Repositório**. De seguida, é enviada uma mensagem para o **Repositório** para obter o bloco correspondente. Uma vez obtido o bloco, do lado do cliente, é calculado o hash desses valores e comparado com o valor do hash presente no recibo. Se ambos os valores do hash corresponderem, significa que o bloco que se encontra na *blockchain* corresponde efectivamente à aposta feita e não foi alterado. Caso não correspondam, significa que o bloco foi alterado, pelo que a *blockchain* é inválida

e tem de ser desconsiderada. Significa que houve fraude.

3.11 Verificar resultado de um leilão

Nesta opção o utilizador deve introduzir o número identificador da aposta que pretende saber o resultado.

Assim sendo, o cliente envia uma mensagem ao **Repositório** seguindo os padrões já descritos e o **Repositório** pesquisa esse leilão na lista de leilões já terminados. Caso não encontre significa ou que o número introduzido não é válido e não existe nenhum leilão com esse número ou que o leilão ainda não terminou pelo que é impressa uma mensagem de erro ao cliente. Se o leilão já tiver terminado, apenas terá acesso às apostas que foram decifradas. Seguiu-se este raciocínio pois no mundo real, e usando o exemplo do Euro milhões, se uma pessoa não reclamar o prémio então o mesmo não lhe é atribuído e assim a outra pessoa. É, portanto necessário que no final de um leilão os utilizadores decifrem as suas apostas, caso contrário poderão ser os vencedores e não serão considerados como tal.

O **Repositório** retorna uma lista das apostas de um dado leilão que foram decifradas. O cliente, após a recepção desta lista, verifica qual a que contém o maior valor. Após encontrada a mesma, publica a identidade de quem a fez bem como o seu valor.

Capítulo 4

Código dinâmico

4.1 Objectivo

O código dinâmico é um passo extra de validação que é executado quando uma nova aposta é feita por um utilizador. Este é definido inicialmente pelo criador de um leilão e enviado juntamente com os outros parâmetros para o **Manager**. O programa aceita apenas simples verificações matemáticas em formato string e não código mais complexo. Se o criador introduzir uma string vazia, considera-se que não existe código dinâmico para esse leilão.

4.2 Método eval()

A execução de código dinâmico em python3 pode ser feita através dos métodos exec() ou eval(), que permitem respectivamente a execução de um objecto de código ou então código no formato string proveniente do utilizador. Das duas opções disponíveis fez-se uso da eval(), que devolve como valor de retorno a solução da expressão string usada como parâmetro de entrada. A função exec() também aceita uma string como parâmetro de entrada, mas diferencia-se no facto de retornar sempre um None tornando-se assim indesejável para o uso no sistema que necessita de uma variável de retorno para saber se uma aposta é ou não válida.

Por causa disto foi usado a função eval() no sistema para conseguir correr código dinâmico introduzido pelos utilizadores para validar as apostas.

4.3 Segurança com o método eval()

No entanto o uso do método eval() requerê medidas adicionais para garantir que esta não é usada por terceiros para atacar o sistema. A aceitação de qualquer string proveniente de fontes não confiáveis poderá resultar em

eventos catastróficos para o sistema. Por exemplo um utilizador malicioso seria capaz de aceder a variáveis privadas, e correr funções de sistema.

É vital então tomar precauções para evitar situações destas. Criou-se uma classe, DynamicCode, com um único método run_dynamic_code que aceita 3 parâmetros de entrada: function, a string onde o código dinâmico do utilizador se encontra, bid, o valor apostado pelo client e highest_bid, o maior valor de aposta feito até ao momento.

O método eval() é chamado dentro desta função para garantir que o código dinâmico só tem acesso a estas 3 variáveis. Para além disso o próprio método permite especificar que variáveis globais e funções builtins do python podem ser chamadas quando a função é corrida, e ao deixar estes campos vazios evita-se que código dinâmico de utilizadores possa fazer uso destas para correr código malicioso.

Capítulo 5

Recibos

5.1 Produção do recibo

O recibo é algo que cada cliente recebe do **Repositório** após fazer uma aposta e que serve como comprovativo em como a sua aposta foi adicionada à blockchain do leilão com sucesso.

O recibo pode ser dividido em 3 partes diferentes. A primeira é escrita pelo cliente quando este faz uma aposta num leilão. É composta por duas partes: o corpo da mensagem denominado 'mensagem', onde se encontra um dicionário com toda a informação sobre a oferta do cliente e a assinatura do cliente.

```
{
  "message"   : <Informacao sobre a oferta>,
  "signature" : <Mensagem assinada com a chave privada do cartao de cidadao do
       cliente>
}
```

Listing 5.1: Recibo no formato inicial.

A segunda parte do recibo é feita pelo **Manager** se a oferta do cliente for aceite. Á mensagem original do cliente é adicionado um novo campo depois da assinatura do cliente que específica se a oferta é válida e uma outra assinatura mas agora do **Manager** feita com os 3 campos anteriores.

```
{
  "message"     : <Informacao sobre a oferta>,
  "signature"   : <Assinatura da mensagem pelo cliente>,
  "bidValidity" : <Informacao sobre se a oferta e valida>,
  "signature"   : <Assinatura do Manager>
}
```

Listing 5.2: Recibo após a oferta ser validada pelo Manager.

Por fim o **Repositório** adiciona mais 2 campo à mensagem depois da assinatura. Um com a informação do bloco criado pela oferta e outro uma assinatura do **Repositório** de todos os campos anteriores.

```
{
  "message"      : <Informacao sobre a oferta>,
  "signature"    : <Assinatura da mensagem pelo cliente>,
  "bidValidity" : <Informacao sobre se a oferta e valida>,
  "signature"    : <Assinatura do Manager>
  "bloco"        : <Informacao sobre o bloco onde a oferta foi guardada>,
  "signature"    : <Assinatura do Repositorio>

}
```

Listing 5.3: Recibo final.

5.2 Validação do recibo

Para validar um recibo é necessário verificar que as assinaturas contidas no mesmo estão correctas. Para tal é reconstruído passo a passa cada assinatura e feita a verificação das mesmas. Se todas estiverem correctas, significa que foram produzidas pelas entidades supostas e que o recibo se manteve inalterado. É portanto um recibo válido. Caso haja alguma que não seja válida, invalida o recibo.

Capítulo 6

Cryptopuzzles

Em relação aos cryptopuzzles, percebemos a sua importância para prevenir ataques DoS (denial of service) e controlar o ritmo de apostas. Isto previne que um utilizador faça aposta sucessivas a um ritmo absurdo o que iria sobrecarregar o sistema e impedir outros utilizadores de terem oportunidade para fazerem as suas apostas.

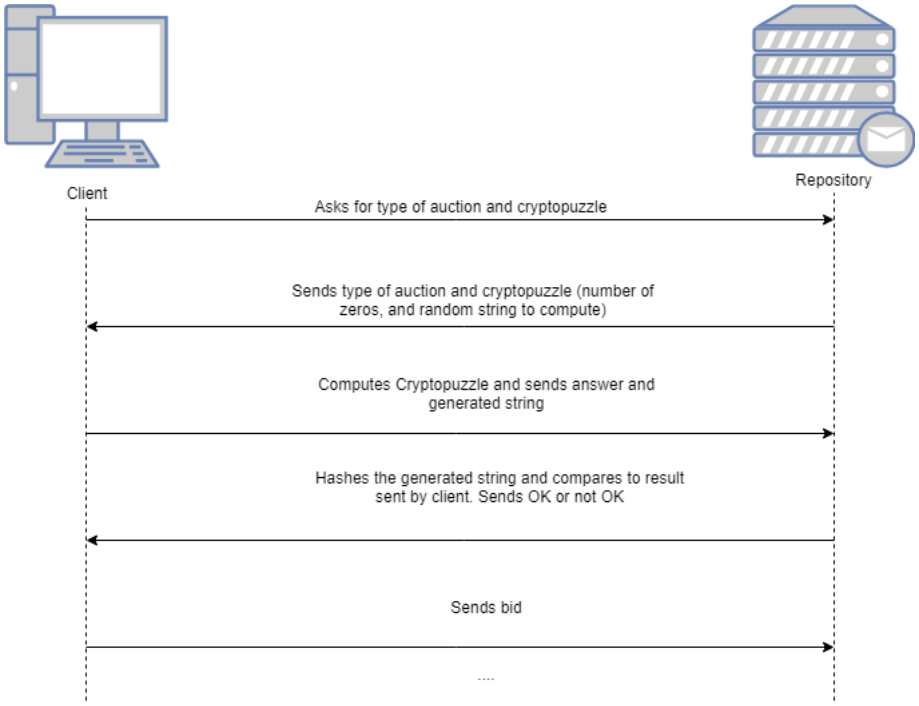
Seguindo a sugestão do Professor André Zúquete utilizamos um algoritmo de **hashcash**, o *proof-of-work* usado pela bitcoin. Um *proof-of-work* consiste em provar que foi computado um dado resultado. Esse resultado é fácil de validar mas difícil de computar e demora tempo a ser calculado.

6.1 HashCash

Este algoritmo consiste na recepção de uma string random, e de um valor de zeros que o resultado final deve ter inicialmente de forma contínua.

A essa string recebida como argumento são acrescentados números random até que o hash da concatenação dessa mesma string com os números e/ou caracteres gerados aleatoriamente (pode também incluir um timestamp além dos caracteres random gerados) contenha o número recebido de zeros contínuos. Este processo exige algum poder de computação visto que é um método de tentativa erro (brute force) e que demora algum tempo, consoante a grandeza da quantidade de zeros desejada e da string inicial.

Segue-se um diagrama de como este processo funciona neste trabalho:


Client
Repository
Asks for type of auction and cryptopuzzle
Sends type of auction and cryptopuzzle (number of zeros, and random string to compute)
Computes Cryptopuzzle and sends answer and generated string
Hashes the generated string and compares to result sent by client. Sends OK or not OK
Sends bid
....

Capítulo 7

Problemas

Apesar de todas as precauções tomadas com o método eval() na parte do código dinâmico, é possível que este não seja completamente seguro, como se pode ver descrito aqui. Como esta vulnerabilidade foi detectada perto do fim do prazo de entrega não foi possível mudá-la a tempo.

Capítulo 8

Conclusões

No que toca aos requisitos deste projecto, todas as funcionalidades foram implementadas, garantindo os objectivos principais deste projecto:

- Integridade, autenticação e confidencialidade das apostas
- Controlo do ritmo de aceitação de apostas e validação das mesmas
- Identificação e anonimato dos autores das apostas
- Garantia de honestidade do repositório

Estes requisitos foram implementados usando como base as sugestões do professor e a nossa interpretação pessoal dos mesmos, implementando consoante o que se achou fazer sentido.

Desta forma, foi possível implementar um sistema de leilões seguro, que permite a criação de leilões e apostas com valor oculto ou identidade dos apostadores oculta.