Milestone de Segurança

Universidade de Aveiro

Departamento de Electrónica,
Telecomunicações e Informática

Segurança

P2G15

Miguel Maia - 76434 maiamiguel@ua.pt

Armando Sousa - 76498 ammsousa@ua.pt

Novembro 2018

Conteúdo

Capítulo 1

Introdução

Este relatório tem como objetivo descrever os mecanismos de segurança estudados e escolhidos para aplicar na segunda parte do projeto de 'Blockchain-based auction management', no âmbito da cadeira de Segurança.

O relatório começa por falar nos mecanismos de autenticidade e integridade usados em geral, seguido de uma descrição mais detalhada e individual das técnicas usadas para cada funcionalidade presente no projeto.

Os mecanismos de segurança a serem aplicados variam consoante as operações realizadas entre os clientes e os servidores. Informação do servidor pública não necessita dos mesmos cuidados que conteúdos sensíveis relacionados com clientes particulares.

Capítulo 2

Autenticação

Como os clientes lidam com duas entidades distintas fornecedoras de serviços, é da mais alta relevância que quando houver a troca de mensagens entre estes, haja a garantia que o interlocutor do outro lado seja mesmo quem ele afirma ser. É assegurada a autenticidade da outra entidade.

Para este fim todas as mensagens enviadas pelos servidores, **Repositório** e **Manager**, para clientes têm de ser assinadas usando chaves assimétricas. Quando um cliente envia uma mensagem para um servidor, assina a mensagem com a sua chave privada de assinatura do CC. Anexa também à mensagem o seu session ID para que o **Manager** saiba que certificado usar para validar a assinatura.

Quando um servidor quer enviar uma mensagem a um cliente, faz o processo inverso. É necessário que o servidor **Manager** guarde os certificados dos seus clientes e faça uma associação dos mesmos aos IDs de sessão. Assim, quando o servidor pretende enviar uma mensagem para um cliente, cifra a mesma com a chave **RSA** simétrica enviada anteriormente pelo cliente cliente e assinada com a sua chave privada. Na receção da mensagem, o cliente verifica a assinatura do servidor,

Para cifrar as mensagens escolheu-se usar a cifra **AES** pois oferece um bom compromisso entre segurança e tempo de processamento.

Isto não implica, no entanto, a integridade das mensagens recebidas, sendo necessário um mecanismo adicional para tal, discutido no próximo capítulo.

Capítulo 3

Integridade

A implementação de mecanismos de integridade é fundamental para prevenir alterações de mensagens por atacantes. Todas as mensagens trocadas irão conter um campo chamado hash obtido usando o **HMAC** na mensagem, para prevenir que o seu conteúdo seja alterado. A entidade que recebe a mensagem, gera uma hash do payload da mensagem com o **HMAC** e compara-o com a hash contida na mensagem. Se forem iguais a integridade da mensagem foi mantida, se não a mensagem sofreu alterações por uma entidade terceira e esta é descartada.

Para o **HMAC** vai ser usado a função de síntese **SHA256**.

```
{
  "ciphered_payload" : <ciphered message>,
  "hash" : <256 bit value>
}
```

Listing 3.1: Mensagem cifrada e com a hash

Capítulo 4

Confidencialidade

A confidencialidade consiste em garantir que servidores não podem tirar partido da identidade dos utilizadores para enganar os clientes. Os certificados dos clientes usados para validar as assinaturas nas mensagens são guardados inicialmente no **Manager** que depois a pedido do **Repositório** válida os clientes.

O **Repositório** fica assim responsável em guardar as *blockchains* dos leilões sem saber quem são os autores dos blocos e o **Manager** encarregado de guardar as identidades sem ter capacidade de alterar o fluxo da *blockchain*.

Como os dois servidores desconfiam um do outro, e fazem a verificação dos dados trocados entre si, não há possibilidade deles enganarem os clientes e ficarem ilesos.

Capítulo 5

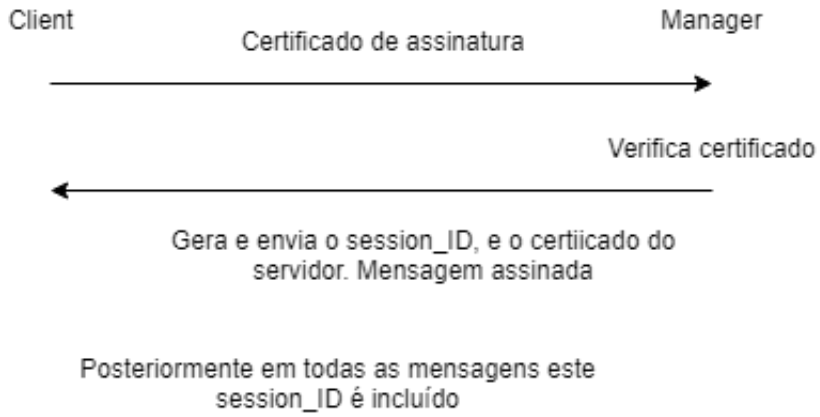
HandShake

Quando um cliente se tenta conectar ao servidor **Manager**, envia uma mensagem do tipo **CONNECT**, assinada, e envia o seu certificado de chave pública de assinatura, bem como uma chave simétrica gerada no lado do cliente e a hash dessa mensagem. Esta chave simétrica irá posteriormente ser usada pelo servidor para enviar mensagens para o cliente.

O servidor **Manager** por sua vez, extraí a chave pública do certificado e verifica a assinatura. Guarda a chave simétrica enviada pelo cliente e faz a associação dessa chave ao cliente.

Se a validação de assinatura e integridade (**HMAC**) forem válidas, o servidor **Manager** envia uma mensagem assinada, contendo o seu certificado de assinatura e chave pública do par de chaves que usa para assinar e ainda um ID de sessão que é um valor de tamanho fixo gerado aleatoriamente e que identifica a sessão. Toda esta mensagem será cifrada com a chave simétrica enviada pelo cliente.

Posteriormente, o cliente em todas as mensagens que envia ao servidor terá que enviar um campo com este ID de sessão.



Capítulo 6

Funcionalidades

6.1 Criar/terminar um leilão

6.1.1 Criar um leilão

Para a criação de um novo leilão o autor envia uma mensagem para o **Manager**, a pedir para criar um novo leilão conforme as especificações dadas.

O autor começa por assinar a mensagem e anexar o seu sessionID à mesma. De seguida a mensagem é cifrada com a chave pública do **Manager** e enviada para ele mesmo. Anexa também uma hash obtida com o **HMAC** da mensagem cifrada para garantir integridade. Assume-se que a chave pública do **Manager** é previamente conhecida pelo cliente, pelo que não há partilha da mesma.

Após chegada da mensagem ao **Manager**, ele primeiro verifica a sua integridade, descartando-a se esta tiver sofrido alterações. No caso de se ter mantido a integridade, o **Manager** decifra a mensagem com a sua chave privada e com o certificado na mensagem, obtido do **Manager** usando o sessionID, verifica a validade da assinatura. Se a assinatura for válida, o **Manager** envia para o **Repositório** a informação pertinente presente na mensagem recebida em diferentes campos.

```
{
  "payload" : {
    "message"       : {
      "typeMSG"      : <type of message value>,
      "name"         : <auction name>,
      "auctType"     : <type of auction: blind or english>,
      "timeLimit"    : <auction's time to die>,
      "description" : <auction's description>
    },
    "signature"    : <message signed with CC's private key>,
    "sessionID" : <session ID>
  }
}
```

Listing 6.1: Mensagem Json criada pelo cliente.

Na imagem anterior é possível ver o esquema da mensagem Json criada pelo

cliente antes do envio. Esta mensagem é depois cifrada e a sua cifra usada para gerar uma hash. É esta cifra e hash que depois são enviadas ao servidor.

6.1.2 Terminar um leilão

O processo inicial para terminar uma leilão é o mesmo que aquele usado quando se cria um leilão, diferindo apenas em alguns componentes. O servidor é o **Repositório** em vez de ser o **Manager** e como tal a chave pública para a cifra também mudas.

No **Repositório**, após a receção da mensagem e a validação do cliente o servidor percorre a lista de leilões abertos e termina aquele com o ID especificado num dos campos da mensagem.

6.2 Listar leilões abertos/terminados

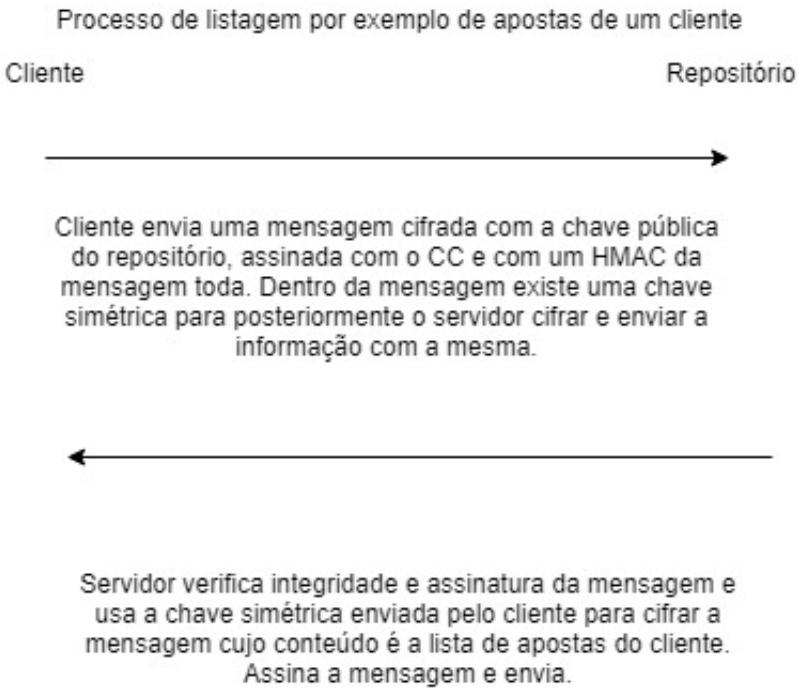
O cliente pede ao **Repositório** para este lhe enviar a lista de leilões a decorrer ou então a lista daqueles que já terminaram. Para isto os mecanismos usados para a troca de mensagens com o intuito de garantir a integridade e autenticidade são os mesmos já discutidos anteriormente para a criação e termino de leilões. O cliente usa uma cifra híbrida para enviar uma chave simétrica que o servidor usará para enviar a lista de leilões. Esta chave é cifrada com a chave pública do servidor, pelo que só o próprio terá acesso.

Quando o **Repositório** responde com a lista ao cliente, este cifra a mensagem onde a informação é armazenada com os valores usando uma chave simétrica enviada pelo cliente na mensagem a pedir a lista de leilões, mensagem que depois só pode ser decifrada pelo proprietário desta chave.

6.3 Mostrar todas as apostas de um leilão

Todas as apostas feitas num leilão são listadas. Só o criador e os participantes de um leilão é que podem receber informação sobre este. Esta informação estará cifrada se o leilão ainda estiver a decorrer ou se no final os clientes tiverem escolhido não decifrar a sua aposta para o público. Para que seja possível ver todas as apostas, todos os clientes têm de se ter conectado e decidido tornar a informação pública. A *blockchain* não é alterada, apenas é anexado a cada bloco a chave simétrica usada para cifrar o conteúdo de cada bloco. Assim sendo, é necessário que todos os clientes se conectem e enviem a sua chave simétrica usada para codificar a informação da aposta, caso contrário poderão ter sido vencedores e não seram considerados como tal.

6.4 Mostrar todas as apostas de um cliente



Todas as apostas feitas por um determinado cliente são listadas. Incluem apostas de leilões abertos e fechados. Quando um cliente envia uma aposta para o **Repositório**, este gera um digest do seu certificado de chave pública e associa as mensagens desse cliente a este digest para ser mais fácil a sua identificação. Assim, quando um cliente faz o pedido das suas apostas, o **Repositório** tem de percorrer a *blockchain* e retornar as mensagens cuja síntese é o mesmo que o da chave pública do cliente correspondente.

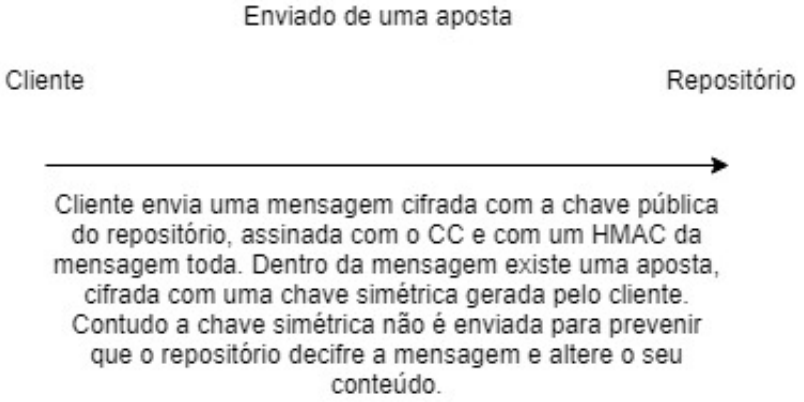
O cliente constroi uma mensagem, com o valor da aposta e o leilão onde vai apostar. De seguida, a mensagem é assinada com o cartão de cidadão do cliente. Com a mensagem assinada, o cliente cifra tudo com a chave pública do servidor e gera uma hash usando o **HMAC**. A hash é anexada à mensagem cifrada formando uma nova mensagem, que é enviada para o **Repositório**. Depois do cliente ter a sua lista de apostas, tem na sua posse a chave que usou para cifrar as mesmas e pode assim decifrar e ver o seu conteúdo.

6.5 Verificar resultado de um leilão

No final de um leilão, um participante tem a possibilidade de verificar o resultado para saber se ganhou. Para que este processo funcione corretamente

é necessário que todos os clientes se conectem. Um cliente que queira verificar o resultado dum leilão terminado, envia a chave simétrica que usou para cifrar as apostas, e o servidor anexa a mesma às apostas correspondentes àquele utilizador. Desta forma a informação torna-se pública pois como a chave está correspondida aos blocos correspondentes, qualquer pessoa pode decifrar a informação.

6.6 Envio de uma aposta



Quando um cliente quer apostar num leilão, o valor da aposta vai cifrada para um leilão *blind* em claro para os *english*. A identidade pode ou não ir cifrada, consoante a vontade do cliente.

Para isso, será usada uma chave simétrica **AES** gerada pelo cliente. Esta chave ficará por agora apenas na pose do cliente e não será enviada para o servidor de modo a evitar que este decifre a mensagem e consiga ler os campos cifrados. Assim sendo, a mensagem cifrada por **AES** é posteriormente assinada com o cartão de cidadão, cifrada com a chave pública do servidor **Repositório**, e anexada com a hash criada usando o **HMAC**. É no fim enviada para o **Repositório**.

É usada cifra híbrida, mas a chave simétrica não é incluída nesta mensagem, sendo que assim evita que o servidor faça batota. No fim dum leilão é necessário que todos os clientes que participaram no mesmo, se conectem e enviem as suas chaves simétricas, para que o servidor possa decifrar as mensagens e assim saber o valor das apostas e consequentemente o vencedor.

Se no fim de um período de tempo todas as apostas não tiverem sido decifradas ganha o cliente com a aposta mais alta das que foram.

Capítulo 7

Código dinâmico

Será enviado para o auction **Manager** código para verificação e validação de apostas. Este código será enviado pelos clientes e depois executado pelo auction **Manager**.

Capítulo 8

Produção e validação de recibos

Um recibo será produzido como o hash duma aposta. Assim sendo, o recibo de uma aposta, será enviado ao cliente que enviou a aposta e este guarda o mesmo. O recibo contém um id identificador da mensagem e do cliente, bem como a assinatura do **Repositório**.

No processo de validação dos recibos, o cliente envia um pedido ao **Manager** de validação de um recibo, e nesse pedido envia o recibo. O **Manager** irá percorrer a blockchain e ver as apostas cujo id é igual ao id do recibo. Irá computar o hash e se corresponderem então o recibo é válido, pois não houve alteração/fraude da aposta.

Capítulo 9

Cryptopuzzles

Em relação aos cryptopuzzles, percebemos a sua importância para garantir prevenção de ataques DoS e controlar o ritmo de apostas. No entanto, ainda não sabemos qual o challenge a usar,

Capítulo 10

Conclusões

Neste relatório explicou-se a implementação pensada para este projecto. Poderá haver alterações consoante feedback do professor e caso seja necessário.