Secure, multi-player online card game Report

MIECT: Security 2019-20

André Mourato - 84745
Mariana Pinheiro - 80186

Dezembro 2019

Conteúdo

Capítulo 1

Introdução

Na disciplina Segurança foi proposto desenvolver um jogo de cartas multi-jogadores, em rede e seguro para avaliação final da componente prática da unidade curricular. Decidimos, por isso, escolher as Copas que consiste num jogo para 4 jogadores, em que o objetivo é obter a menor pontuação. O jogo começa pela distribuição de 13 cartas por cada jogador. O jogador que tiver o 2 de paus começa o jogo, com essa mesma carta. Em rondas seguintes o jogador que ganhou o turno anterior pode começar com qualquer carta. Os restantes jogadores devem assistir, sempre que possível. O vencedor de cada ronda será o jogador que assistiu e cuja carta tem o maior valor. Quando os jogadores ficarem sem cartas na mão, é efetuada novamente a distribuição do baralho. O jogo termina quando pelo menos um dos jogadores atingir 100 pontos. Contudo no jogo desenvolvido, um jogo consiste em apenas 1 ronda (e não em várias).

No nosso projeto foram desenvolvidos dois clientes: um cliente usado por um humano com uma interface gráfica e outro cliente que permite criar bots (jogadores não humanos), usado para testar a aplicação, em python e um servidor, ou *croupier*, desenvolvido em NodeJS, devido à sua eficiência para lidar com vários pedidos de utilizadores.

Capítulo 2

Funcionalidades

2.1 Autenticação do Cliente no *Croupier*

2.1.1 Introdução

Tal como referido no enunciado o servidor é uma entidade não certificada, e como tal o cliente não faz qualquer tipo de validação da chain of trust. Contudo, todas as mensagens enviadas do servidor para o cliente são assinadas com a chave privada do servidor e validadas pelo cliente com a chave pública. Não existe qualquer tipo de chave de sessão estabelecida entre o cliente e o servidor, uma vez que um atacante não iria ganhar informação útil ao capturar os pacotes enviados entre ambas as entidades. Toda a informação trocada em claro nesta fase (nome do jogador, lista de jogadores conectados ao croupier, lista de jogos a decorrer, certificados, chaves públicas, etc.) é de conhecimento público e não compromete as mesas de jogo.

2.1.2 Com Cartão de Cidadão (CC)

O cliente começa por gerar um par de chaves RSA (para além do par de chaves do CC). De seguida o cliente autentica-se no servidor, enviando uma mensagem contendo o seu *username* e a informação necessária para autenticação. Esta informação é assinada com a chave privada do CC e contém o certificado do CC juntamente com a chave pública gerada pelo cliente. O servidor obtém a chave pública do CC através do seu certificado e valida o conteúdo da informação assinada. Após esta validação, o servidor usa o certificado do cliente para validar a cadeia de confiança (chain of trust). Após ambas as validações anteriores o servidor guarda a chave pública gerada no inicio pelo cliente.

2.1.3 Sem Cartão de Cidadão (CC)

O cliente começa por gerar um par de chaves RSA. De seguida o cliente autentica-se no servidor, enviando uma mensagem, assinada com a sua chave privada, contendo o seu *username* e a chave pública que gerou. Este método de autenticação não é forte, uma vez que não conseguimos identificar de forma única que um username pertence a um jogador. É apenas usado para permitir o teste da aplicação, sem requerer 4 cartões de cidadão.

2.2 Autenticação das Mensagens e Controlo de Integridade de Dados

O controlo de integridade das mensagens é garantido através da assinatura das mensagens com RSA+SHA256. Todas as mensagens enviadas pelo servidor para o cliente e vice-versa são assinadas com a chave privada do emissor e validadas pelo recetor com a chave pública do emissor. Apesar deste método ser mais lento comparado com outros métodos de validação de integridade de dados como HMAC, foi escolhido pela simplicidade de implementação. O conteúdo das mensagens trocadas têm o seguinte formato <operação>:<argumentos>, onde *operação* é a operação que será invocada no recetor, *argumentos* são os argumentos que vão ser passados para essa função. Quando uma mensagem precisa de ser transmitida, o emissor encapsula o conteúdo da mensagem com a assinatura da mesma, produzindo a mensagem digitalmente assinada <operação>:<argumentos>:<assinatura>. Quando for recebida pelo recetor, este valida a assinatura, garantindo assim a autenticação e integridade de dados. De seguida desencapsula e efetua a operação pretendida.

2.3 Juntar um Jogador a uma *Game Table*

Quando um jogador se junta à mesa, tem a opção de aceitar ou recusar jogar com os seus oponentes. Para isso, o servidor anuncia o certificado, a assinatura e a chave pública de todos os jogadores com Cartão de Cidadão. De seguida, cada jogador valida a assinatura e chain of trust dos seus oponentes através do certificado do CC. Por fim é enviada, para o servidor, uma mensagem assinada com a lista dos jogadores da mesa. Este irá proceder para a fase de estabelecimento de chaves de sessão entre os jogadores.

2.4 Estabelecer sessão entre os jogadores de uma *Game Table*

Após todos jogadores aceitarem permanecer na mesa, são estabelecidos canais seguros entre cada jogador. Para estabelecer uma sessão segura entre

os jogadores, o servidor começa por enviar a cada jogador as chaves públicas dos seus oponentes. A figura 2.1 demonstra o estabelecimento de chaves de

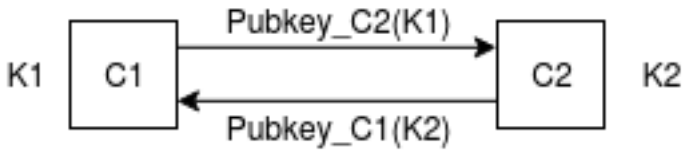

Figura 2.1: Estabelecimento de uma chave de sessão entre o jogador 1 e o jogador 2

sessão entre os jogadores 1 e 2. Cada jogador gera uma sequência de 128 bits, K1 e K2, cifra com a chave pública do oponente e envia o resultado para o adversário. Este resultado é decifrado pela chave privada do recetor.

A chave de sessão entre os dois jogadores será o bitwise XOR das duas chaves. Para o caso dos jogadores 1 e 2 a chave de sessão entre eles será $K1 \oplus K2$ que é equivalente a $K2 \oplus K1$, não sendo por isso necessário determinar uma ordem para calcular a chave de sessão. Isto é aplicado para todas as ligações entre jogadores e no total são estabelecidos 12 canais seguros por cada *game table*.

Uma solução alternativa seria usar Curva Elíptica ou Diffie-Hellman para estabelecer uma chave de sessão entre 2 jogadores, no entanto, optamos por esta abordagem, apesar da simplicidade de implementação de ambas as soluções.

2.5 Protocolo Seguro de Distribuição o Baralho

Esta fase do jogo é complexa e está, por isso, dividida em 4 etapas: cifra conjunta do baralho, seleção das cartas, comprometimento com a mão selecionada, revelação da mão.

2.5.1 Seleção do Algoritmo e Modo de Cifra Simétrica

Para esta distribuição é usada criptografia simétrica, cujo algoritmo e modo pode ser escolhido pelo utilizador antes de se dar inicio à distribuição. O cliente com menor nome de utilizador por ordem alfabética pode escolher entre os algoritmos AES, Camellia, TripleDES, CAST5, SEED, Blowfish e IDEA e os modos CBC, OFB, CFB.

Existiam imensas alternativas para a seleção do jogador que escolherá o algoritmo e modo de cifra, mas esta foi considerada a mais simples.

2.5.2 Cifra Conjunta do Baralho (Randomization Stage)

Uma lista, D, com 52 cartas circulará pela mesa, por ordem alfabética do nome de utilizador. Quando cada jogador recebe o baralho, baralha-o aleatoriamente, gera uma sequência de 128 bits, conhecida por K, e cifra por criptografia simétrica cada carta com K. O algoritmo e modo a usar foram previamente determinados e partilhados com a mesa. Quando o baralho tiver passado por todos os jogadores, este é devolvido ao servidor.

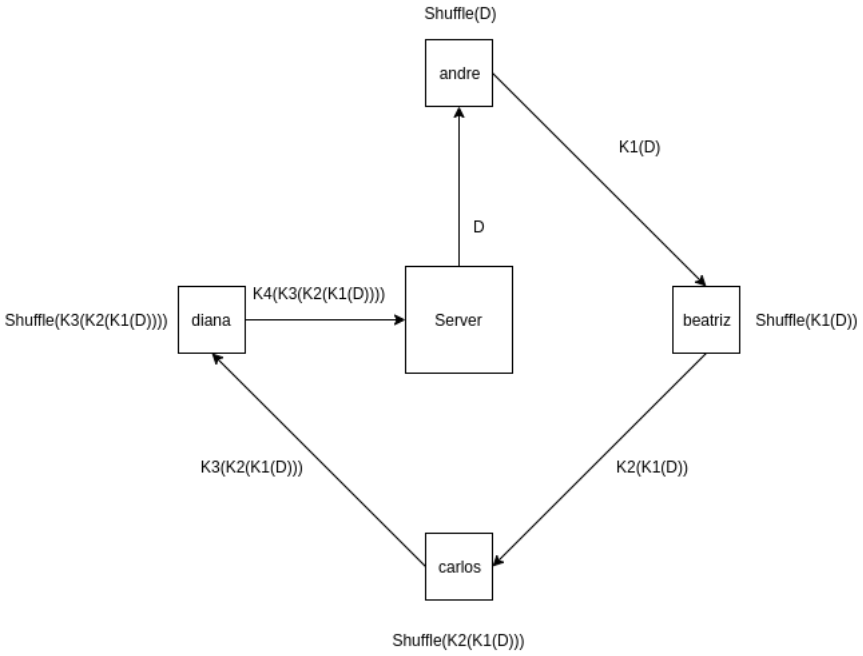

Figura 2.2: Cifra Conjunta do Baralho

2.5.3 Seleção das Cartas (Selection Stage)

Após a etapa de cifra do baralho, o servidor envia a um jogador aleatório o baralho com as 52 cartas cifradas. O jogador recebe o baralho e pode realizar 3 operações: retirar uma carta e colocá-la na sua mão, colocar 1 ou mais cartas da sua mão de volta no baralho, ou não fazer nada. Por fim, envia para outro jogador, escolhido de forma aleatória. Este processo repete-se até cada jogador ter 13 cartas na mão. Como as chaves de cifra do baralho serão reveladas antes do jogo começar, sempre que cada jogador quiser enviar o baralho para outro jogador, deverá cifrar o baralho, com as cartas cifradas, com a chave simétrica do jogador para o qual pretende enviar. Este processo está ilustrado na figura 2.3, sendo A,B,C cartas cifradas na etapa de cifra conjunta do baralho e K12 a chave de sessão entre os jogadores C1 e C2. Isto é necessário porque o servidor serve de intermediário da informação trocada

entre dois jogadores e, por isso, se tivesse acesso ao texto cifrado de cada carta, poderia determinar a mão de cada jogador quando as chaves fossem reveladas.

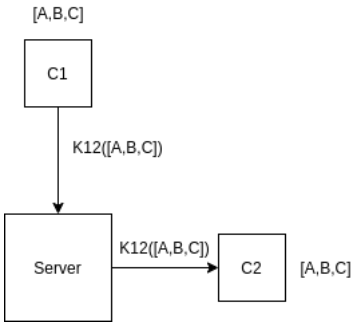

Figura 2.3: Transmissão do baralho restante de um jogador para outro

2.5.4 Comprometimento com a Mão Selecionada (Commitment Stage)

Nesta etapa, cada jogador possui 13 cartas cifradas na sua mão e é necessário que cada jogador se comprometa com a sua mão antes das cartas serem reveladas. Para isso, cada jogador gera duas chaves, R1 e R2, com 128 bits cada. De seguida é calculado o bit commitment, b, de (R1,R2,C), com SHA256, em que C é o baralho de cartas cifradas (a mão do jogador). Após b ser calculado, (R1,b) são enviados para o servidor. O R2 é usado para prevenir que outras entidades adivinhem C (com um ataque da preimagem) e é apenas revelado caso seja necessário verificar se um jogador possui uma determinada carta. Contudo, R2 por si mesmo não é suficiente, uma vez que um ataque de colisão poderia descobrir mais do que um par (R2,C) que produza o mesmo bit commitment. Daí surge a necessidade de introduzir R1, que força o gerador do bit commitment a revelar um dos valores usados na sua computação. Isto garante que só existe um bit commitment que possa ser aceite pelo servidor para validar a mão de um jogador.

2.5.5 Revelação da Mão (Revelation Stage)

Após cada jogador ter-se comprometido com a sua mão, o servidor revela as chaves usadas para cifrar as cartas do baralho. Cada jogador irá, de seguida, decifrar cada carta da sua mão com as chaves reveladas pela ordem inversa pela qual foram cifradas, isto é, para cada carta decifram com a chave do jogador com o último nome por ordem alfabética, depois o terceiro, o segundo e por fim o jogador com o primeiro nome por ordem alfabética. A figura 2.4 demonstra este processo. Assumindo que

a, b, c e d correspondem respetivamente às chaves do António, da Beatriz, do Carlos e do Diogo e que a mão do António, foi cifrada pela ordem $carta_{cifrada} = C_d(C_c(C_b(C_a(carta))))$ então deverá ser decifrada pela ordem inversa $carta = D_a(D_b(D_c(D_d(carta_{cifrada}))))$

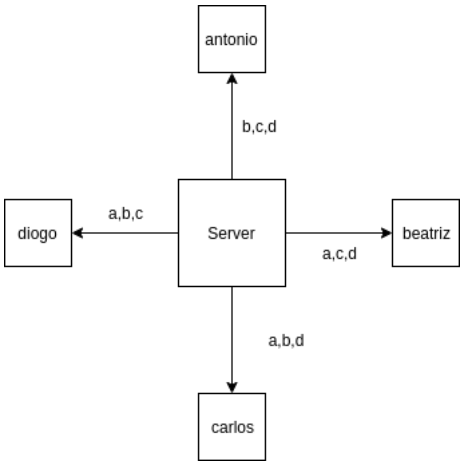

Figura 2.4: O servidor revela as chaves de todos os jogadores

2.6 Jogo das Copas

Nesta secção iremos descrever os mecanismos de segurança implementados para garantir a segurança e a evolução correta do jogo. Para isso, todas as entidades irão colaborar para detetar a existência de batota.

2.6.1 Jogar uma Carta

Para se jogar uma carta, o jogador envia ao servidor uma mensagem assinada que contém a carta que pretende jogar. Inicialmente o servidor aguarda que o jogador com o 2 de paus jogue a carta. Assim que este o fizer, anuncia a todos a carta jogada. O jogador seguinte joga a carta, e o processo repete-se até estarem 4 cartas na mesa e é determinado o vencedor da jogada. Cada entidade é responsável por acompanhar todas as métricas do jogo (cartas jogadas pelos adversários,etc.).

2.6.2 Protestar batota

Durante um jogo, e depois da jogada de cada carta qualquer jogador pode protestar caso considere que o adversário fez batota enviando uma mensagem assinada ao *croupier*, onde identifica a fraude cometida. É possível reclamar 3 tipos de fraude: um adversário jogou a minha carta, um adversário jogou

a mesma carta duas vezes ou um adversário não assistiu ao naipe quando devia (renuncia).

O *croupier* apenas irá verificar se ocorreu batota se receber a acusação de um jogador. O servidor irá de seguida determinar se o acusador (jogador que acusa) tem razão ou se o acusado (jogador que é acusado) está inocente, transmitindo, por fim, a conclusão da sua verificação a todos os jogadores e terminando, de seguida, o jogo.

2.6.3 Validação das cartas jogadas por cada jogador durante o jogo

Como referido anteriormente, nós decidimos que o servidor só ia validar as cartas caso algum jogador se manifesta-se. No nosso jogo existe 3 tipos de batota. Apesar de ser trivial verificar se um jogador jogou uma carta duas vezes, ou fez renuncia, é bastante mais complexo verificar se jogou uma carta de um adversário. As verificações seguintes são feitas pelo servidor com a colaboração de todos os jogadores.

Jogou uma Carta de um Adversário

Como a mão de cada jogador é secreta às restantes entidades, esta verificação torna-se mais complexa, contudo é possível verificar de quem é a culpa através do bit commitment feito na distribuição segura do baralho. Quando este tipo de batota é detetado, o acusador envia uma mensagem assinada, a avisar o servidor que a batota foi detetada, que contém a carta em causa e o jogador acusado. De seguida, o jogador acusado envia para o servidor o par (R2,C), assinado, usado para calcular o bit commitment, em que C é o conjunto de cartas cifradas. O servidor calcula o bit commitment *b'* do jogador que foi acusado $b' = h(R1, R2, C)$, em que h representa o hash com SHA256. Após esta etapa, o servidor compara e garante que $b = b'$. Caso sejam iguais, conseguimos determinar a mão inicial do acusado, decifrando C, pela ordem inversa das 4 chaves usadas para cifrar as cartas na primeira etapa do Protocolo de Distribuição Segura do Baralho. Após a mão inicial do jogador ser determinada verifica-se se a carta em causa está presente na mão inicial do acusado. Se isto se verificar, então o acusado é inocente e o culpado é o acusador que terminou o jogo em vão. Se a carta não estiver presente na mão inicial do acusado, então este é o culpado.

Jogou uma Carta Duas Vezes

No caso em que o jogador é acusado de jogar uma carta pela segunda vez o processo de verificação é trivial, passando por ir verificar as cartas que o jogador acusado jogou.

Renuncia

Se o jogador for acusado de não assistir a um naipe, é necessário verificar as jogadas em que esse jogador não assistiu a um naipe, por exemplo espadas, porque significa que já não possui mais cartas com esse naipe. Se em jogadas futuras o jogador jogar uma carta com espadas, então é detetada renuncia. Se isto for detetado, o jogador é acusado de ter feito renuncia e o jogo é terminado.

2.6.4 Possibilidade de batota

Cada jogador tem a possibilidade de fazer batota através do cliente desenvolvido. A carta maliciosa será enviada para o servidor, como qualquer outra, e compete às entidades restantes fazerem a verificação da existência de batota. O cliente com interface gráfica apresenta a lista de cartas na mão, mas também possui a opção **I choose to cheat!** que permite enviar não só qualquer carta, mas também qualquer string para o servidor.

2.6.5 Adição de Jogador à Lista Negra (black list)

Quando é detetada batota, é determinado o culpado e o inocente da ocorrência. O culpado será adicionado à lista negra do servidor, que consiste apenas em guardar num ficheiro do diretório src/server/black_list,que será criada caso não exista, uma mensagem com o ID do jogo, o ID do acusador, o ID do acusado e o resultado da averiguação da culpa. Se um jogador, acusador, acusa outro, acusado, de ter cometido uma fraude, existem dois resultados possíveis. Se o acusador estiver correto e o acusado for o batoteiro, então o acusado é adicionado à lista negra, caso contrário o acusador é adicionado à lista negra.

2.6.6 Apuração do Resultado (game accounting) e Consulta do Histórico de Jogos

O jogador que obteve a menor pontuação é o vencedor. É guardada num ficheiro do diretório src/server/game_accounting uma mensagem que contém o ID do jogo, o ID e a pontuação de cada jogador na mesa. Esta mensagem é assinada com a chave privada do CC (onde o utilizador deverá introduzir o seu PIN). É, também, possível consultar o histórico de jogos de um jogador. Para isso o servidor lê todos os ficheiros do diretório src/server/game_accounting/<nome-do-jogador> e verifica a assinatura com o certificado do CC fornecido pelo jogador na autenticação.

Capítulo 3

Implementação

3.1 Problemas conhecidos e limitações

O software do cartão de cidadão que usamos para este projeto pode ter um comportamento instável e raramente gera um Segmentation Fault, contudo este não é um erro da nossa aplicação. Como o acesso CCs com assinatura digital ativada foi limitado, nem sempre pudemos validar a chain of trust, por isso desativamos temporariamente o mecanismo de segurança que não permite avançar sem que o CC tenha um certificado válido, contudo a verificação é feita de qualquer forma. Apenas é validada a autenticidade dos jogadores com CC na fase em que todos os jogadores devem aceitar jogar com os seus oponentes, antes do jogo começar, uma vez que esses jogadores não possuem certificados e como tal não há forma de realizar uma autenticação forte. Os jogadores sem CC devem, por isso, ser usados apenas para testar a aplicação.

Capítulo 4

Conclusão

Durante a realização do nosso projeto sentimos dificuldades em implementar algumas das funcionalidades pedidas no enunciado, dando destaque à inclusão do Cartão de Cidadão e implementação do bit commitment no projeto, tendo sido solicitado ajuda ao professor em multiplas ocasiões. No entanto, podemos concluir que conseguimos contornar essas dificuldades e alcançar o nosso objetivo que era implementar todas as funcionalidades do enunciado.