universidade de aveiro

deti universidade de aveiro departamento de electrónica, telecomunicações e informática

Secure Domino Game

MIECT - Security 2020-21

Dany Costa 85097
Diogo Daniel 89221
Pedro Alves 88861
Sérgio Gasalho 84760

Janeiro 2021

Conteúdo

Introdução

Para o desenvolvimento do projeto, baseamo-nos no jogo clássico do Dominó. quatro jogadores, cada um com uma mão de sete peças, jogam uma peça por jogada, sendo vencedor o jogador com a pontuação mais baixa, calculada através da soma dos números das peças restantes do jogador no final do jogo.

Cada partida tem um máximo de quatro jogadores e um mínimo de dois, e existe sempre um tableManager (servidor). Todos os clientes estabelecem comunicação com o tableManager, este orquestra as comunicações entre os jogadores, gere a lógica do jogo e declara o vencedor quando não existe batota.

Implementamos nas várias fases do projecto mecanismos de segurança tais como a autenticação dos jogadores perante o servidor, a distribuição inicial de peças de forma secreta e aleatória, o comprometimento em jogar com a mão atribuída por parte do jogador, o correto desenrolar do jogo, comunicações seguras e a detecção de batotas e respectivas penalizações.

Ao longo da próxima seção do relatório, iremos explicar em detalhe o projeto de uma forma sequencial, ilustrando cada etapa de uma partida.

Desenvolvimento

Antes demais é importante referir que todo o código do jogo foi desenvolvido pelo grupo, não tendo sido utilizado código partilhado pelos nossos colegas. Assim sendo, e com o intuito de melhor contextualizar a arquitetura cliente-servidor, alguns detalhes serão aqui explicados.

Em primeiro lugar é de valor mencionar os módulos comuns entre as entidades. Estes ficheiros contam com funções ou classes desenvolvidas com o intuito de separar algumas das operações do servidor e dos clientes. São elas, por exemplo, operações criptográficas simétricas (**"symetric_functions.py"**), operações criptográficas assimétricas (**“criptoAssym.py”**), operações de controlo de integridade(“**integrity_control.py”**), operações com o Cartão de Cidadão (**"cartaocidadao.py"**), operações de bit commitment (**“bit_commitment.py”**) ou mesmo operações de troca de mensagens para cliente/servidor (**“send_receive.py”**).

Inicialmente e após colocar o servidor em funcionamento, os vários clientes vão se conectando ao mesmo, sendo que o jogo é realizado apenas quando todos os clientes que compõem a mesa estiverem conectados ao servidor. Isto é controlado através da Waiting Room que “bloqueia” os clientes enquanto a mesa não é composta.

Quando a mesa estiver completa é iniciada a **serverThread** que irá controlar a distribuição das peças através de um protocolo de distribuição seguro e após todos os jogadores terem 7 peças cada um, começa o jogo, sendo que no final (se não ocorrer batota) serão mostrados os pontos de cada jogador após autenticação com o cartão de cidadão. É ainda possível realizar um novo jogo no final de cada partida respondendo afirmativamente a isso no servidor e nos clientes.

Agora que a arquitetura e a maneira como os módulos se interligam entre si está bem definida, começar-se-á por explicar alguns dos algoritmos e protocolos usados para implementar as funcionalidades pedidas. Começaremos por descrever alguns procedimentos que os clientes devem tomar antes de se conectarem ao servidor.

Setup inicial do cliente

Após o servidor ser posto em execução, o cliente necessita de estabelecer alguma configuração inicial antes do jogo ser iniciado, nomeadamente o username e a probabilidade de o jogador fazer batota. Sendo que este último é dado como argumento aquando da inicialização do jogador(**cliente.py**).

A instância da classe Player fica encarregue de obter os argumentos necessários e de instanciar a thread de execução implementada na classe **playerThread** (que controlará toda a execução do lado do cliente).

Nesta classe serão estabelecidas algumas variáveis e algum conteúdo importante para estabelecer com sucesso a comunicação com o Table Manager (servidor). Será gerado o par de chaves assimétricas usadas para fazer autenticação inicial e estabelecimento de chaves de sessão.

O algoritmo utilizado para gerar o par de chaves assimétricas é **RSA**, o tamanho da chave é de 2048 bits.

As chaves de sessão mencionadas, têm um tamanho de 16 bytes. A cifra de blocos utilizada é **AES** no modo de operação **CBC**.

Para verificar simultaneamente a integridade dos dados e a autenticidade de uma mensagem é utilizado **HMAC**, a função de hash utilizada é **SHA-256**. As chaves usadas para gerar o **HMAC** são as chaves de sessão partilhadas pelos intervenientes.

Aceitação do Table Manager

O cliente após gerar o seu par de chaves assimétricas procede a conectar-se com o servidor (WaitingRoom). O WaitingRoom pede aos clientes, quando se conectam, a sua chave pública e depois a sua assinatura e faz a sua validação. Caso seja válida o WaitingRoom guarda a chave pública do cliente e envia a sua chave pública. Após isto é feito um pedido ao cliente para gerar e enviar uma chave de sessão que vai ser utilizada para comunicações futuras entre cliente e servidor. A mensagem com a chave de sessão é cifrada com chave pública do servidor por parte do cliente e é enviada (criptografia híbrida).

Quando todos os clientes estiverem conectados o servidor envia-lhes o array que contém as chaves públicas de cada cliente.

No final deste processo cada cliente vai ter uma chave de sessão com o servidor e todas as chaves públicas dos intervenientes no jogo. As chaves de sessão partilhadas entre clientes serão geradas numa fase posterior.

Troca de mensagens cliente-servidor

Mensagens com Cifra

- Estabelecimento de chaves de sessão
- Lista de clientes
- Processo de randomização
- Mensagens entre clientes na escolha de peças
- Revelação dos mapas
- Mensagens entre clientes na preparação da de-anonimização
- Cálculo de Pontos

Mensagens de estado nunca são cifradas

Mensagens com Integridade

- Processo de randomização
- Escolha de peças
- Bit commitment
- Revelação dos mapas
- Mensagens entre clientes na preparação da desanonimização
- Cálculo de Pontos
- Envio da Lista de Jogadores
- Declaração do Vencedor
- Utilização do baralho

Pseudonomização das peças

No início do jogo o servidor cria um stock de dominós com indexes totalmente aleatórios para cada dominó ficando com o resultado que podemos ver na figura seguinte.

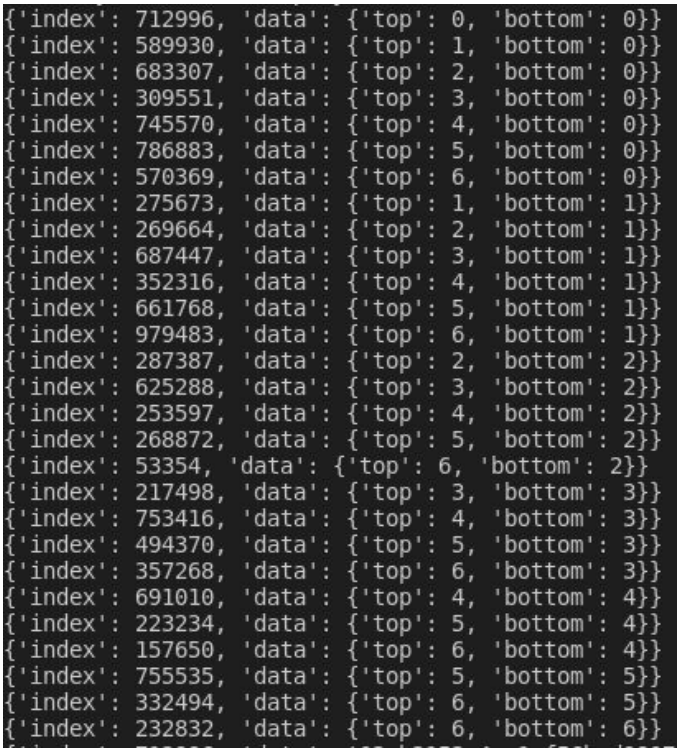

Fig 1 - Stock das 28 peças sem pseudônimos

Depois de terem sido geradas estas 28 peças o servidor inicia o processo de pseudonomização das peças com base na seguinte função fornecida:

Pi = h (i, Ki , Ti)

Na classe table.py, onde podemos observar todo este processo , na função para criar o pseudônimo (**addPseudonymTile()**) começamos por criar uma hash através da biblioteca **hashlib** usando **md5(peça)** que é o nosso **Ki**. De seguida, fazemos um processo análogo para a criação dos pseudônimos , usando a mesma biblioteca, porém em vez de ser apenas para a peça em questão a hash é criada através do index, peça e chave obtendo o pseudônimo **Pi**. No final da função, são guardados num dicionário todos estes valores de forma a que através pseudônimo só se consiga obter a chave e só através desta é que é possível obtermos o valor original da peça juntamente com o index. Este processo é feito para todas as 28 peças.

No fim deste mecanismo o stock é baralhado e o que observámos na figura 1 é "transformado" no que stock que podemos observar na figura 2 de modo a que já se possa inicializar os mecanismos de distribuição de peças.

Fig 2 - Stock das 28 peças com pseudônimos

Distribuição das peças

Processo de Randomização

Inicialmente, o servidor faz circular o baralho com os pseudónimos sequencialmente por cada um dos clientes que fazem parte da mesa, respeitando uma ordem definida. Cada jogador cifra individualmente cada uma das peças que lhe são enviadas usando para isso um algoritmo de chave simétrica, para cada cifragem é utilizada uma chave diferente. Neste passo, se houver uma cifragem que devolva o mesmo resultado de alguma feita anteriormente, essa colisão é detectada, é recalculada a chave e procede-se a uma nova cifragem com a mesma. Para cada peça, o cliente guarda um mapa interno que faz a associação Ci -> Ki , como na nossa implementação o algoritmo utilizado é AES em modo CBC, temos necessidade de para além de guardar a chave(ki), guardar também o IV(IVi), sendo por isso Ki = [ki, IVi], Ci corresponde à peça "i" cifrada, i é um valor unitário sequencial que está no intervalo de [0,27]. O jogador envia para o servidor uma lista com

todos os Ci, devidamente baralhados. O próximo jogador da lista vai receber os Ci enviados pelo jogador anterior e vai repetir o processo mencionado em cima. Após a passagem do baralho por todos os jogadores, este terá as 28 peças cifradas um número de vezes igual ao número de jogadores(se forem 4 jogadores , cifrada 4 vezes). E cada jogador vai ter um mapa interno Ci -> Ki. As comunicações onde o baralho é enviado entre cliente e servidor são cifradas com a chave de sessão que foi trocada inicialmente entre ambos.

O baralho poderá agora circular entre cada jogador para estes poderem escolher as peças, sem qualquer um deles ter acesso ao conteúdo.

Escolha de Peças

Nesta etapa, o baralho randomizado circula secretamente entre os jogadores, e estes vão retirando peças(cifradas) até possuírem 7 peças na mão.

O servidor escolhe inicialmente um jogador aleatório da lista de jogadores e envia o baralho randomizado. O jogador a quem chega o baralho tem duas opções, escolher uma peça ou não escolher, dentro da opção de não escolher o jogador tem a possibilidade de trocar peças que tem na mão com peças do baralho . Foi implementada uma probabilidade de 5% de o jogador escolher uma peça, aumentando assim a confusão no processo fazendo com que seja mais difícil haver seguimento do que está a acontecer. Após a execução desta ação o jogador envia diretamente o baralho sem a peça que escolheu para um jogador escolhido aleatoriamente.

Para que não haja a possibilidade de ocorrer “eavesdropping” de entidades externas e para que esta informação que passa no Table Manager não ser compreendida pelo mesmo, os jogadores sempre que comunicam entre si, cifram a mensagem com a chave de sessão trocada por ambos.

Quando um jogador pretende enviar o baralho para o outro jogador, se ainda não tiver sido estabelecida uma chave de sessão entre ambos, procede-se à troca da mesma. O jogador que envia a mensagem calcula uma chave simétrica(chave de sessão) e cifra com a chave pública do jogador destino, garantindo que apenas ele a pode decifrar. Esse jogador decifra usando a sua chave privada, e obtém a chave de sessão, faz a associação entre a chave e o jogador que a enviou e guarda num mapa, que vai ter as chaves de sessão com cada jogador. O jogador que calculou a chave armazena-a de igual modo. Podendo agora trocar o baralho de uma forma segura.

No caso em que os jogadores já tenham trocado chaves de sessão, não há nova troca de chaves, procede-se apenas ao envio da mensagem cifrada.

Quando o jogador tiver sete peças alerta o servidor que a sua mão já está cheia, quando o servidor receber o número de alertas igual ao número de jogadores, a fase de escolha da peça termina. O servidor pede ao último jogador o baralho.

Bit Commitment

b=h(R1,R2,T)

Antes de haver a revelação das peças e após o jogador já ter as 7 peças na mão (ainda cifradas e com pseudônimos) o jogador cria um bit commitment **b** onde produz um um hash através de dois números random (**R1**,**R2**) e as peças (pseudônimos) cifradas **T** que possui. Este processo pode ser visto na função **bit_commitment()**.

Após ter criado o bit commitment o jogador vai verificar se o bit commitment foi bem feito através da função **check_commit()** em que são dados como argumentos o R1,R2, T e o b que foi gerado e gera-se de novo um bit commitment (b') temporário para verificar se este coincide com o que foi criado (b). Se o bit commitment for bem gerado o jogador guarda numa variável o bit commitment todo, incluindo o b,R1,R2 e o T.

1ª parte

Ainda antes de serem reveladas as peças que cada jogador tem em posse é feito um envio parcial do bit commitment, mais precisamente o **R1,b.** Este processo acontece antes de ser feita a revelação das peças de forma a que o jogador não tenha maneira de conseguir forjar o **T** que foi usado para a geração do bit commitment uma vez que quando se enviar a segunda parte do bit commitment (**R2, T**) facilmente se detecta que o **T** não corresponde ao **T** que foi usado na geração de **b**.

Revelação da mão do jogador

O servidor pede aos jogadores, por ordem inversa da definida da randomização, o mapa de cifragem de todas as peças, exceto as que estão no baralho e partilha com os outros jogadores.

Cada jogador quando recebe o mapa, decifra uma camada das suas peças. No final desta fase os jogadores vão ter o index e pseudónimo das suas peças.

As mensagens enviadas que contém os mapas de cifragem são cifradas, para garantir que apenas os intervenientes no jogo conseguem ter acesso aos mesmos. A cifragem é feita através das chaves de sessão trocadas entre os clientes e o servidor.

Preparação da de-anonimização das peças

Nesta fase cada jogador vai preencher um array de forma aleatória, para isso foi usado o mesmo método utilizado na escolha das peças. O servidor escolhe aleatoriamente o primeiro jogador e envia-lhe o array vazio(**self.de_anonymizationPubKey**) que vai ser utilizado para colocar as chaves. O jogador ao receber o array tem duas hipóteses, colocar ou não uma chave. A escolha de colocar é feita de maneira aleatória com probabilidade de 5%. Caso insira, ele gera um par de chaves assimétricas para uma peça da sua mão que ainda não esteja no array. Coloca a chave pública no mesmo na posição do index da peça e a chave privada num array local (**self.de_anonymizationPrivKey**). No fim da inserção ou caso tenha escolhido não colocar a chave, o jogador escolhe alguém aleatoriamente para enviar o array das chaves públicas. O jogador que colocar a última chave no array das chaves públicas, envia(para além do array) um aviso para o servidor saber que o processo acabou. Neste ponto as comunicações entre os clientes são cifradas com as chaves de sessão partilhadas por ambos.

De-anonimização das peças

O servidor por cada pseudónimo cujo index se encontra no array de chaves públicas (**self.de_anonymizationPubKey**) obtém o valor da peça original e a chave que deu origem ao pseudónimo (função **getTileFromPseudonym()** no ficheiro game.py) e consoante o index do pseudónimo usa a chave pública que está no array para cifrar o valor da peça original e a chave do pseudónimo. Desta forma, apenas o proprietário da peça consegue ter acesso ao valor da peça e à chave utilizada no pseudónimo, decifrando com a chave privada que no array (**self.de_anonymizationPrivKey**) mencionado na etapa anterior.

Quando o jogador obtém estes valores, juntamente com o index da peça recalcula o pseudónimo, e faz comparação com o pseudónimo para essa peça. Se os valores forem iguais, o processo foi bem sucedido, senão o servidor enganou o jogador e o jogo deve parar.

Utilização do baralho durante o jogo

Quando um jogador não tem nenhuma peça possível para jogar, recorre ao baralho e tira uma peça do mesmo até obter uma que sirva para jogar ou até que o baralho fique vazio. Se conseguir obter uma peça válida joga-a, se não passa a vez para outro jogador.

Após o envio de pedido por parte do cliente para o servidor de necessidade de ir ao baralho, o servidor responde com o baralho(que ainda se encontra cifrado, resultado da fase de randomização). O cliente escolhe uma peça desse baralho, após isso é enviada para o servidor para que seja feita a decifragem da mesma, o servidor recebe a peça e pela ordem inversa à da randomização envia-a para o primeiro jogador da lista, este devolve a chave + iv que decifram o valor enviado, o servidor guarda a chave+iv num array e utiliza esses valores para decifrar a peça escolhida e envia o resultado para o segundo jogador, e assim sucessivamente até ao último jogador. O servidor envia para o cliente o array de chaves que decifram a peça escolhida. O cliente usa essas chaves e obtém o pseudônimo da peça, pede agora ao servidor uma de anonimização. Este consulta os indexes dos pseudónimos e envia ao cliente o valor da peça e a chave. Com isso, à semelhança do que foi feito na fase anterior, o cliente recalcula o pseudônimo e deteta se o servidor o enganou.

Fase de Jogo

Em termos logísticos, após todos os processos de distribuição terem sido concluídos de forma segura e íntegra , o servidor cria uma lista totalmente random que contém a ordem dos jogadores. De seguida, o servidor entra num ciclo que funciona da seguinte forma:

1. O primeiro jogador entra no processo de jogo onde recebe inicialmente o registo das jogadas e o tabuleiro, que estão vazios pois trata-se do primeiro jogador, e envia a peça que quer jogar, após o pedido do servidor.
    a. O servidor pede ao jogador a peça , o lado onde quer meter a peça e se pretende a peça invertida ou não
    b. O jogador vai ,consoante a mão que tem (se não fizer batota), escolher uma peça totalmente aleatória neste caso pois trata-se do primeiro jogador
    c. O servidor adiciona a peça ao tabuleiro
    d. É de notar que no primeiro jogador e no servidor ,apesar de haver mecanismos onde estes verificam se alguém fez batota ou se a peça jogada é válida ou não, estes são todos ignorados pois trata-se da primeira jogada.
2. O segundo jogador vai entrar no mesmo processo de jogo passando pelos mesmos passos que o primeiro jogador, contudo já pode haver **detecção de batota**.
    a. Quando o jogador recebe o registo das jogadas e o tabuleiro ele vai verificar de imediato se houve alguém que jogou uma peça que este contém na sua mão

   i. caso isto aconteça o jogador indica ao servidor que **detetou um batoteiro** e já não avança mais para a escolha das peças
b. Se não for encontrado nenhum batoteiro então este vai para o processo de escolha da melhor peça. Para escolher uma peça válida o jogador tem que verificar todas as peças disponíveis que tem válidas para jogar e entre elas escolhe uma aleatoriamente e de seguida envia para o servidor.
c. O servidor recebe a peça proveniente do jogador e verifica se a peça é válida e adiciona ao tabuleiro
d. Após adicionar ao tabuleiro o servidor vai verificar se existe alguma peça repetida no tabuleiro
   i. caso haja alguma peça repetida são acionados os métodos de **detecção de batota**

3. O próximo jogador,se houver, faz o mesmo que foi feito em 2 e assim sucessivamente enquanto o jogo permanecer ativo.

O jogo permanece ativo até que umas das situações aconteça:

- um jogador fica sem peças na mão
- os jogadores mesmo com peças não mão não conseguem fazer nenhuma jogada válida
  - depois de já terem ido ao stock buscar todas as peças que sobram
- é detectada uma batota e inicia-se o processo de reconhecimento do batoteiro

Envio das Jogadas

Os jogadores para jogarem uma peça que seja válida devem analisar primeiro as peças que estão na mesa para poderem obter peças válidas consoante as que tem em sua mão (caso não faça batota).

Para escolher a peça, independentemente se escolhe peças provenientes da sua mão ou peças vindas do stock ilegal, o jogador cria uma lista que vai conter todas as peças possíveis de jogar , ou seja , peças válidas para serem introduzidas no tabuleiro quer seja no início ou no fim do mesmo e, de seguida , escolhe aleatoriamente uma dessas peças para jogar e , por ultimo, envia para o servidor. O conteúdo dessa lista vai depender sempre se são peças provenientes da mão ou se são peças vindas de um stock “ilegal”.

Como foi mencionado acima, quando é lançado o jogador é dado sempre como argumento a probabilidade do jogador fazer batota sendo que , se o argumento for 100 o jogador vai ser 100% batoteiro e caso seja 0 o jogador não vai fazer batota nenhuma.

Envio sem batota

Caso o jogador não faça batota nenhuma, ele cria a lista e vai adicionar todas as peças possíveis de jogar que estão na sua mão e dentro dessas peças vai escolher uma aleatoriamente e envia ao servidor.

Envio com batota

Para explicar este processo é preciso realçar que o jogador tem em posse um stock considerado ilegal que está no ficheiro dominoTiles.txt onde este vai buscar as peças ilegais caso pretenda fazer batota. Sendo assim, neste processo, o jogador em vez de adicionar à lista peças válidas com base nas que tem na sua mão, vai adicionar peças provenientes desse stock ilegal.

Para explicar melhor este mecanismo vejamos o seguinte exemplo:

1. Jogador A foi lançado com um probabilidade de fazer batota de 40%
2. Jogador A vai escolher uma peça
    a. Antes de escolher uma peça é gerado um número inteiro entre 1 a 100
        i. Se o número for menor ou igual a 40% siginifica que o jogador vai adicionar peças á lista com base no stock ilegal.
        ii. Se o número for maior ou igual a 40% o jogador vai adicionar à lista peças pertencentes á sua mão
3. Depois de obter a lista (**possibile_choices**) o jogador vai jogar:
    a. uma peça escolhida aleatoriamente dessa lista
    b. uma peça “falsa” com um index = -1 ,caso a lista esteja vazia, para indicar ao servidor que este não possui nenhuma peça possível para jogar
        i. Neste caso , se existirem peças no stock , o servidor vai pedir ao jogador para tirar peças do stock até que encontre uma peça válida ou até que o stock esteja vazio.

Neste mecanismo todo o papel do servidor baseia-se em verificar se a peça escolhida pelo utilizador é válida de forma a poder acrescentá-la ao tabuleiro ou ,caso o jogador não tenha peças para jogar, indicar a este para escolher peças do stock até que encontre uma válida ou até que o stock fique vazio.

Deteção de batotas

Para realizar este processo durante o jogo, tanto os jogadores como o próprio servidor estão constantemente a verificar os mecanismos de detecção de batota antes de jogar qualquer peça (no caso do jogador) ou verificar se a peça é válida (no caso do servidor). Consequentemente, existem dois mecanismos de detecção de batoteiros:

1. Quando se joga uma peça repetida que já está na mesa
    a. O servidor após o jogador jogar cada peça vai verificar se essa peça já está na mesa percorrendo peça a peça
        i. Caso isto se verifique, o servidor passa de imediato ao processo de procura para encontrar qual foi o outro jogador que jogou a peça repetida.
        ii. Depois de encontrar os dois jogadores envolvidos na peça repetida este pede aos jogadores a segunda parte do bit commitment
        iii. Por fim, após pedir aos dois jogadores a segunda parte do bit commitment é iniciado processo de declaração de batoteiros que é explicado em baixo
2. Quando alguém joga uma peça que um jogador possui na sua mão
    a. Neste processo, como foi supramencionado, o jogador antes de jogar qualquer peça que seja vai verificar se alguém jogou uma peça que está na sua mão através das peças que estão na mesa
        i. Caso isto aconteça o jogador envia de imediato uma mensagem ao servidor a queixar-se e nem sequer chega a jogar uma peça pois foi encontrado uma ilegalidade
        ii. O servidor depois de receber o pedido de queixa de um jogador vai procurar , através do registo de jogadas que tem , qual foi o outro jogador que jogou a peça suspeita
        iii. De seguida, quando o servidor ja encontrou os dois jogadores envolvidos na jogada da peça ilegal este pede aos respetivos jogadores que envie o segundo commit.
        iv. Por último, para garantir que o jogador que se queixou não estava a mentir é iniciado o processo de declaração de batoteiros que é explicado em baixo

Utilização das peças sobrantes

Sendo que o dominó é constituído por 28 peças, e inicialmente são sempre distribuídas 7 peças por jogador, quando o jogo é constituído por 2 ou 3 jogadores, irão existir 14 e 7 peças sobrantes (respetivamente). Neste projeto foi assim implementada a utilização de peças sobrantes por parte dos jogadores, quando os mesmos não têm peças válidas para jogar.

Assim, quando um jogador necessita de peças sobrantes, o servidor envia-lhe o Stock, onde o jogador irá escolher uma peça qualquer, sendo que as peças do Stock encontram-se ainda cifradas por todos os jogadores pois passaram pelo processo de randomização, aquando da distribuição das mãos. Após a escolha da peça, o jogador, irá pedir ao servidor para lhe enviar as keys de todos os jogadores necessários para decifrar a peça, para que o jogador possa assim proceder à decifragem. Após este processo, irá requisitar ao servidor uma de anonimização da peça e acrescentar a peça à sua mão. Este processo ocorre até que o jogador tenha uma peça válida, sendo que caso tenha uma peça válida, o jogo procederá o seu fluxo normal.

Bit Commitment 2ª parte

Quando é feita a detecção de um batoteiro o servidor imediatamente pede aos dois jogadores envolvidos na peça suspeita/forjada para enviarem a segunda parte do bit commitment (**R2,T)** e envia esta segunda parte para os restantes jogadores.

É de realçar que o jogador que fez batota pode tentar forjar o **T** (com uma probabilidade pequena de 10%) antes do seu envio de forma a tentar com que ninguém descubra que foi ele que jogou aquela peça ilegalmente. Este método será obviamente o mais fácil de descobrir pois tanto o servidor como os outros jogadores quando forem verificar a veracidade do bit commitment vão reparar que o jogador tentou forjar o T e encontram de imediato o batoteiro pois o bit commitment gerado **b'** será diferente de **b** enviado inicialmente.

Declaração de batoteiros

Após ser feito o envio da segunda parte do bit commitment por parte dos jogadores que jogaram a peça considerada suspeita (ou forjada) é realizado o seguinte processo:

1. O servidor e todos os jogadores verificam se os bit commitments dos dois jogadores estão corretos gerando um bit commitment **b'** através do R1,R2 e T enviados pelo jogador e a seguir verifica se esse bit commitment temporário **b'** é igual ao **b** que foi enviado na primeira parte do bit commitment
    a. Se uma das verificações não bater certo significa que esse jogador tentou forjar o **T** e consequentemente foi encontrado o batoteiro
2. Se (1) não for suficiente para detectar quem realmente fez batota é necessário que o servidor revele o **T** para ver qual dos jogadores é que jogou uma peça que não lhe foi atribuída inicialmente.
3. Decifra-se o **T** e para isso o server utiliza o mapa de chaves que foi usado no processo de revelação das peças para conseguir decifrar **T**
4. Após decifrar o **T** passamos a ter um conjunto de pseudônimos e não de peças e para isso o servidor tem que revelar os pseudônimos acedendo a lista inicial que criou na secção de pseudonomização de peças.
5. Com as peças obtidas o servidor , à medida que vai revelando as peças, vai enviando para todos os jogadores o resultado da revelação para que estes tomem conhecimento.
6. Depois de ser revelada as peças é obtido o jogador batoteiro verificando qual deles não contém a peça suspeita/forjada.

Resumindo, existem dois processos de declarar um batoteiro, através da verificação dos bit commitments (**b)** dos jogadores ou ,caso o primeiro seja feito sem sucesso, revelando **T**. Por fim, após terminado todo este processo os jogadores mandam para o servidor o resultado da sua análise da descoberta de batoteiros e depois de todos concordarem com a declaração dos mesmos (servidor e jogadores) o jogo é terminado.

Reclamação dos pontos através do Cartão de Cidadão

Para ser possível apresentar os pontos aos clientes, é necessário assinar o respectivo username do cliente e provar que o mesmo não alterou o seu username em nenhuma parte da execução do jogo. Esta garantia de que não houve trocas de identidade é efetuada através de Cartão de Cidadão. No início do jogo, e após ser atribuído um username a um dado cliente, o username foi guardado num dicionário na WaitingRoom.py , em que a chave é o socket do cliente e o valor o respectivo username. No final do jogo, quando o jogador pede para que a sua pontuação seja calculada, envia a sua mão para o servidor, e assina o seu username através do Cartão de Cidadão. O servidor irá por sua vez verificar a assinatura feita pelo cliente recorrendo ao dicionário criado anteriormente, pedindo o username através do socket do cliente que enviou a assinatura. Se a verificação da assinatura do cartão de cidadão for bem sucedida então garantimos que o jogador não trocou a sua identidade sendo então os pontos calculados armazenados num dicionário cuja key é o socket, e os pontos são o respectivo valor e de seguida serão retornados ao cliente.

Declaração do Vencedor

Após o envio dos pontos obtidos por cada jogador para os mesmos, o servidor irá proceder à atribuição do vencedor da partida. Uma vez que no dominó vence aquele que tiver menor número de pontos, é verificado no dicionário criado no passo anterior qual o socket que tem menor número de pontos. Esse será declarado o vencedor e a username do vencedor e respectiva pontuação será enviada a todos os jogadores.

Fim de jogo

O final de uma partida é dado quando um dos jogadores fica sem peças numa mão (sendo que este jogador será o vencedor), caso ocorra uma batota (sendo que o jogo será abortado) ou caso todos os jogadores não tenham peças válidas para jogar (sendo o vencedor o que tiver menor pontuação). Após o final da partida é dada a possibilidade de jogar novamente por parte do servidor aos clientes.

Teste das funcionalidades

O servidor deve ser o primeiro a ser lançado pois é ele que cria a mesa e que estabelece a conexão entre os jogadores , sem ele os jogadores não conseguem jogar. Por outro lado, o cliente pode ser lançado com uma probabilidade X de fazer batota entre 1 a 100 sendo que se for lançado com 0 o jogador não vai fazer batota nenhuma e se for lançado com 100 irá fazer certamente batota.

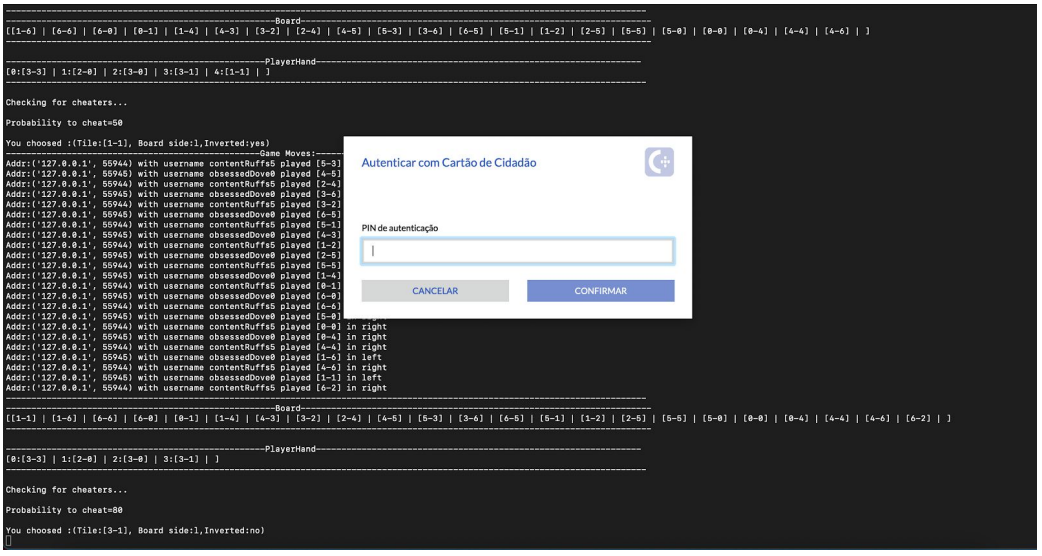

Fig. 3.1: Autenticação com Cartão de Cidadão para anterior ao cálculo da pontuação e declaração do vencedor

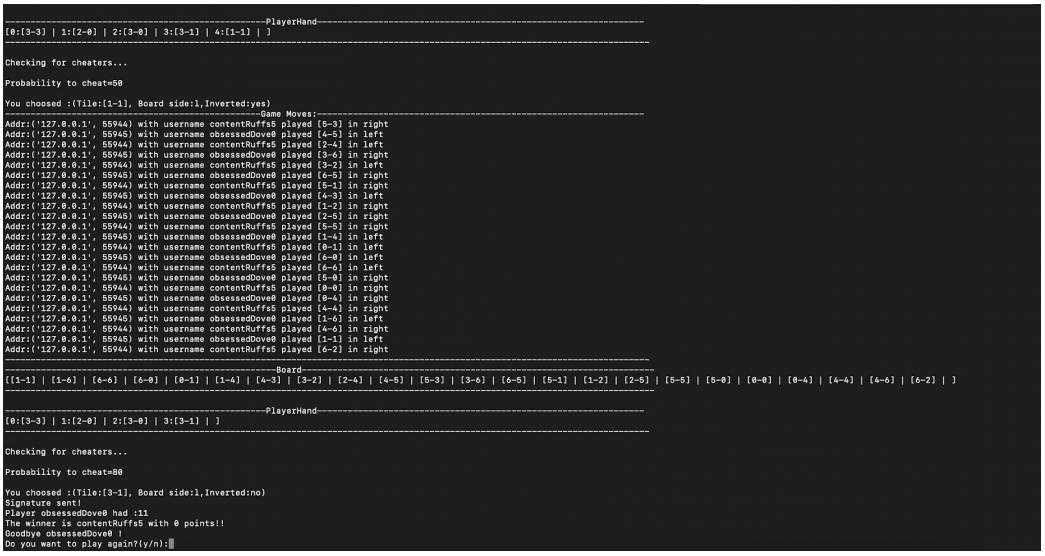

Fig. 3.2: Final do jogo com os pontos do jogador e a declaração do vencedor

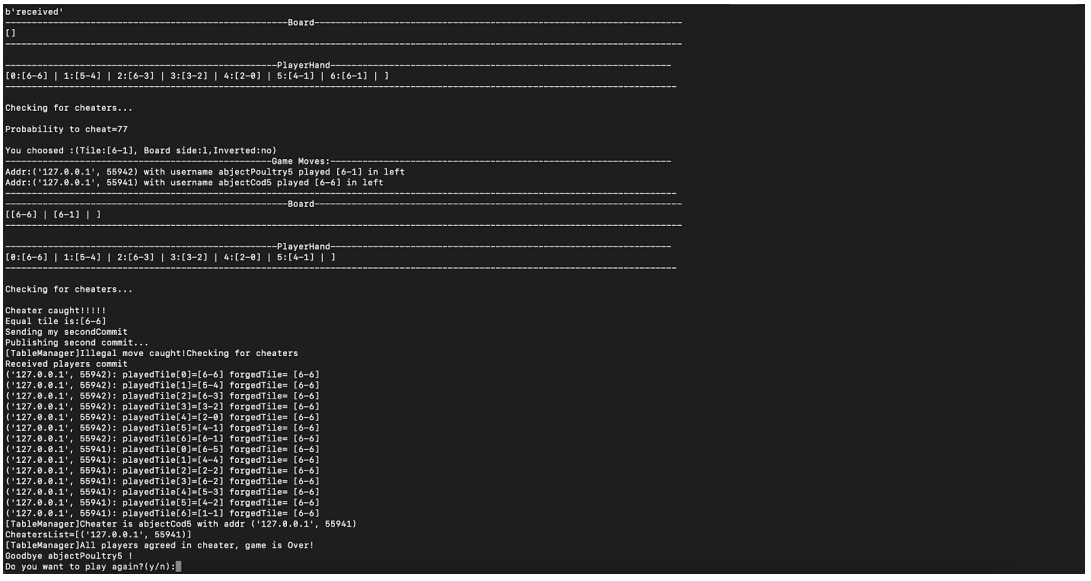

Fig. 4. Exemplo de um jogo que foi abortado devido a batota não havendo lugar a pontuação e vencedor

Conclusões

Apesar das dificuldades enfrentadas, o grupo conseguiu desenvolver e terminar todas as funcionalidades e implementações descritas no enunciado do projeto de forma segura e robusta. O jogo conta não só com comunicações seguras, íntegras, confidenciais e autenticadas, entre entidades mas também com protocolos fiáveis de distribuição de peças para prevenir ataques de ouvintes exteriores, com deteção e reclamação de cheating, utilização de peças sobrantes e ainda reclamação de pontos através do cartão de cidadão.

Contribuições

Dany Costa 85097 - 27%
Diogo Daniel 89221 - 27%
Pedro Alves 88861 - 27%
Sérgio Gasalho 84760 - 19%