Secure, multi-player online Bingo game

Cátia Azevedo, 80316

28 de Setembro de 2020

Conteúdos

1 Introdução

No âmbito da unidade curricular de Segurança foi proposto o desenvolvimento "de uma espécie de jogo do Bingo", em rede e seguro, de modo a ser avaliada a componente prática da UC.

O projeto contém 3 módulos principais, Client, Manager(servidor) e Security.

Em suma, e de uma maneira muito simplista, o jogo do bingo pode ter tantos jogadores quantos aqueles que se tiverem autenticado até ao momento em que o utilizador der ordem de começo do jogo ao Manager. Logo depois, o Manager cria um baralho e de seguida todos os jogadores escolhem(geram) as suas próprias cartas. Podem existir cartas repetidas, e se assim for, em caso de vitória o prémio é divido por todos os jogadores com a mesma carta vencedora. O vencedor será o primeiro jogador a terminar uma das suas cartas.

Todo este processo, desde o início ao fim, é implementado com o auxílio de protocolos de segurança, que serão explicados e esquematizados mais à frente.

O projeto foi desenvolvido em Python 3.8.

2 Funcionalidades

2.1 Privacidade das Mensagens(Segurança)

Numa fase inicial, a privacidade das mensagens trocadas é garantida através de criptografia assimétrica(sistema criptográfico que faz uso de um par de chaves, pública e privada, em que qualquer emissor pode encriptar uma mensagem com recurso à chave pública do recetor, mas essa mensagem só pode ser desencriptada com recurso à chave privada do recetor) usando RSA.

RSA, de todas as outras alternativas de implementação de criptografia assimétrica(Diffie-Hellman, Elliptic Curve, etc), tornou-se a mais vantajosa de implementar tendo em conta que, o algoritmo RSA para além da sua mais simples implementação pode também ser usado para assinar assinaturas digitais(importante no controlo de integridade e autenticação das mensagens) mas, requer a troca de uma chave pública de antemão, o que acaba por ir de encontro ao que é descrito no enunciado(a chave pública do Manager é conhecida).

Ao contrário da criptografia simétrica, em que a chave normalmente é apenas uma série aleatória de bytes, as chaves RSA têm uma estrutura interna com propriedades matemáticas específicas, como se pode ver na Figure 1.

```
def rsaKeyPair():
    privateKey = rsa.generate_private_key(
        public_exponent=65537,
        key_size=4096,
        backend=default_backend()
    )
    publicKey = privateKey.public_key()
    return publicKey, privateKey
```

Figure 1: Geração do par de Chaves Assimétricas

Para a geração da chave privada usando RSA existem dois fatores fulcrais, como se pode ver acima:

- O public exponent, que indica qual será a propriedade matemática da geração da chave. A menos que haja um motivo específico para fazer o contrário, deve usar-se sempre 65537.

- O key size descreve quantos bits de comprimento a chave deve ter. Chaves maiores fornecem mais segurança; atualmente, 1024 e abaixo são considerados quebráveis, enquanto 2048 ou 4096 são tamanhos de chave padrão razoáveis para novas chaves.

A geração da chave pública RSA faz-se através da chave privada.

Sendo assim, podemos chegar à conclusão que todo este processo de encriptação assimétrica usando RSA é implementado através da encriptação da mensagem(enviada pelo Client) com recurso à chave pública do Manager(que já é conhecida). Este irá desencriptar a mensagem com recurso à sua própria chave privada(Figure 2).

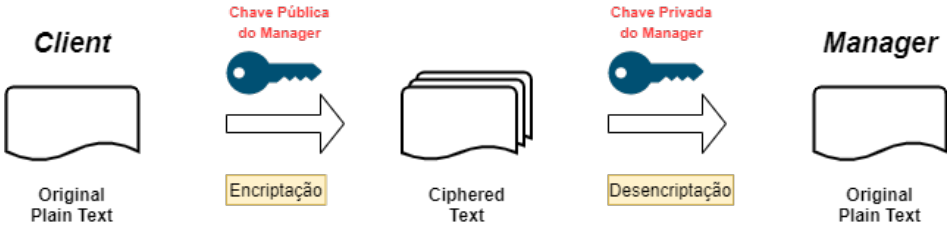

Figure 2: Criptografia Assimétrica

Aquando desta fase inicial, o Client gera uma chave de Sessão, tendo em conta o conceito de criptografia simétrica(sistema criptográfico que faz uso de uma única chave, em que o emissor encripta uma mensagem com recurso a essa chave e o recetor apenas a pode desencriptar se tiver acesso à mesma), usando AES-128.

Apesar de em algoritmos de criptografia simétrica a chave de Sessão ser compartilhada e consequentemente, o risco de comprometimento ser maior (enquanto que em algoritmos de criptografia assimétrica a chave privada não é compartilhada e o processo geral é mais seguro), a decisão de passar a usar criptografia simétrica adveio do facto de, algoritmos de criptografia assimétrica serem computacionalmente mais intensivos e, portanto, muito mais lentos. Normal-

mente são também usados em transações menores, principalmente para autenticar e estabelecer um canal de comunicação seguro antes da transferência de dados real, que é o que acaba por acontecer no projeto.

AES-128, de todas as outras alternativas de implementação de criptografia simétrica(RC4, DES, etc) e com o auxílio de Fernet, que se encontra incluído na biblioteca cryptography do Python, acaba por ser a alternativa de mais simples implementação(Figure 3).

AES usa a mesma chave de 128, 192 ou 256 bits para encriptar e desencriptar. Como Fernet apenas faz uso de AES-128, acaba por ser o utilizado. A segurança de um algoritmo de chave simétrica não aumenta necessariamente com o aumento do comprimento da chave, isso é importante mas para algoritmos de chave assimétrica.

```
def aesKey():
    key = Fernet.generate_key()
    return key.decode()
```

Figure 3: Geração da Chave Simétrica de Sessão

Sendo assim, e a partir da primeira mensagem, em que o Client envia ao Manager a chave de Sessão, a privacidade das mensagens passa a ser garantido pela encriptação e consequente desencriptação de todas as mensagens fazendo uso da chave de Sessão (tanto pelo Manager como pelo Client, tendo em conta que ambos tem a posse da mesma), tal como se encontra esquematizado na Figure 4.

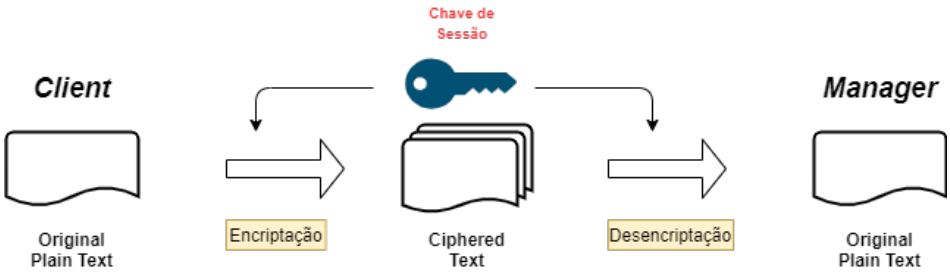

Figure 4: Criptografia Simétrica

2.2 Controlo de Integridade dos Dados e Autenticação das Mensagens

As assinaturas digitais neste projeto são usadas principalmente para atender a dois objetivos principais: controlo de integridade dos dados e autenticação das mensagens.

O controlo de integridade dos dados garente que o conteúdo de determinado dado não foi alterado desde o momento em que foi enviado pelo emissor, até ao momento em que foi recebido pelo recetor. Como veremos mais à frente, o controlo de integridade dos dados acaba por se tornar importante, tendo em conta que no caso específico do jogo do Bingo é necessário garantir que, por exemplo, o baralho final enviado pelo Manager aos Clients não foi alterado.

A autenticação garante que a informação recebida através de uma mensagem realmente veio da entidade que é suposto ter enviado a mensagem. Por outras palavras, é um mecanismo que permite a autenticação do emissor.

Sendo assim a o controlo de integridade dos dados e a autenticação das mensagens, neste caso, são feitos utilizando assinaturas RSA. Uma assinatura RSA é um hash(SHA-256) da mensagem encriptado com recurso à chave privada do emissor, e posteriormente o recetor verifica essa mesma assinatura (a verificação RSA de uma assinatura consiste em desencriptar a cifra com recuros à chave pública do emissor e dar hash(SHA-256) à orginal, comparando assim as duas e verificando a assinatura)(Figure 5).

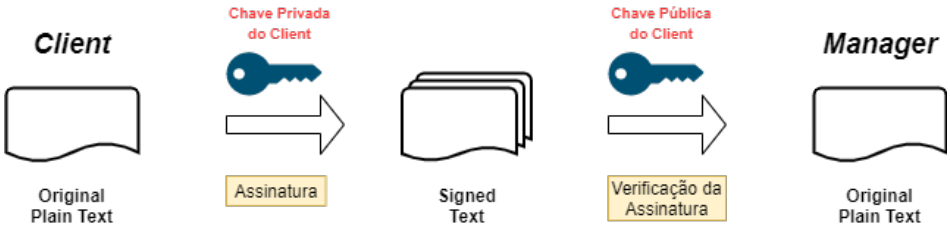

Figure 5: Assinatura e Verificação RSA

2.3 Autenticação do Client

2.3.1 Protocolo de Autenticação Sem Cartão de Cidadão

De modo a proceder à autenticação do Client, foi criado um Protocolo de Autenticação, presente na Figure 6, que consiste no envio de duas mensagens por parte do Client e uma mensagem por parte do Manager. O objetivo no final da última mensagem, que é enviada pelo Client, é o Manager chegar à conclusão que todas as mensagens até então recebidas foram enviadas pelo Client e portanto reunem-se as condições necessárias para estabelecer sessão entre ambos, e assim, o Client participar no jogo que poderá iniciar a qualquer momento.

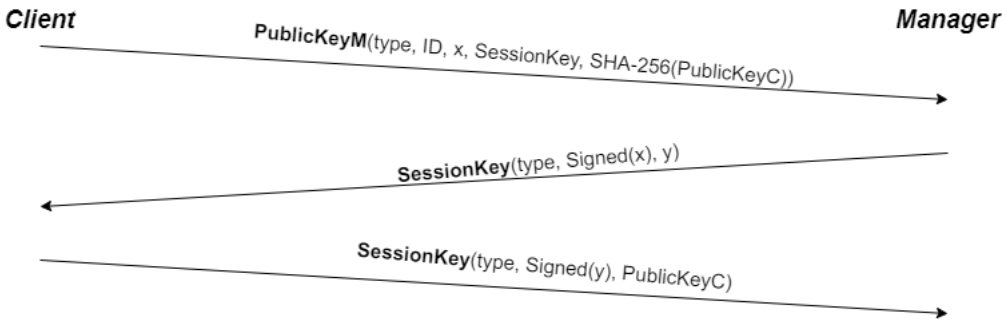

Figure 6: Protocolo de Autenticação

A primeira mensagem, como se pode ver acima(Figure 6), é enviada pelo Client, que fazendo uso da chave pública do Manager, encripta 5 campos de mensagem:

- O primeiro, o tipo da mensagem, que neste caso é 'AUTH0'.

- O segundo, um ID que permite identificar cada Client.

- O terceiro, um número para o Manager assinar e o Client, posteriormente, poder verificar essa assinatura.

- O quarto, a chave de Sessão, criada pelo Client e enviada ao Manager para a partir daí usarem essa chave para estabelecer a troca de mensagens entre ambos de uma maneira segura.

- O quinto, um SHA-256 da sua própria chave pública, tendo em conta que o Manager não tem como ter acesso a esta. Enviá-la numa função de Hash permite que seja mapeado um dado grande e de tamanho variável(chave pública do Client) para um pequeno dado de tamanho fixo(por esse mesmo motivo, as funções Hash são conhecidas por resumirem o dado).

O manager ao receber esta mensagem, desencripta-a com auxílio da sua própria chave privada e no envio da segunda mensagem encripta 3 campos, já fazendo uso da chave de Sessão recebida do Client:

- O primeiro, o tipo da mensagem, que neste caso é 'AUTH1'.

- O segundo campo é a assinatura do número enviado pelo Client para o Manager assinar e assim o Client poder verificar essa assinatura.

- O terceiro campo é um número para o Client assinar e posteriormente o Manager, de igual forma, poder verificar essa assinatura.

O Client recebe esta mensagem, desencripta-a usando a chave de Sessão, e procede ao envio da terceira mensagem, encriptada com a chave de Sessão, composta por 3 campos:

- O primeiro, o tipo da mensagem, que neste caso é 'AUTH2'.

- O segundo campo é a assinatura do número que o Manager enviou para o Client assinar.

- O terceiro campo, é a sua chave Pública.

Ao receber esta terceira e última mensagem, o Manager consegue autenticar o Client não só através da verificação da assinatura enviada pelo mesmo, como dá Hash à chave Pública do Client comparando-a assim com a chave Pública Hashed enviada pelo Client no primeiro campo da primeira mensagem, e verificando que é a mesma.

2.4 Protocolo de Compremetimento com a Mão Escolhida(Commitment stage and Preparation stage)

Para esta etapa foi desenvolvido o Protocolo de Comprometimento com a Mão Escolhida, que se enontra na Figure 8. É caraterizado pela troca de 3 mensagens, duas enviadas pelo Manager e uma enviada pelo Client. O objetivo pretendido no final do protocolo é que todos os Clients obtenham todos os bit commitments(Figure 7) de todos os outros Clients e, quando chegar a altura da computação do vencedor, poderem validar as cartas dos outros Clients.

BitCommitment = SHA-256(**nonce1**, **nonce2**, **Cards**)

Figure 7: Bit Commitment

O bit commitment, neste caso, é calculado usando uma função hash(SHA-256) de dois valores aleatórios(nonce1, nonce2) e, assumindo que o valor a comprometer é C(conjunto de cartas do Client), de um terceiro valor C. É publicado como (nonce1, BitCommitment) e para provar a exatidão desse compromisso é necessário publicar (nonce2, BitCommitment).

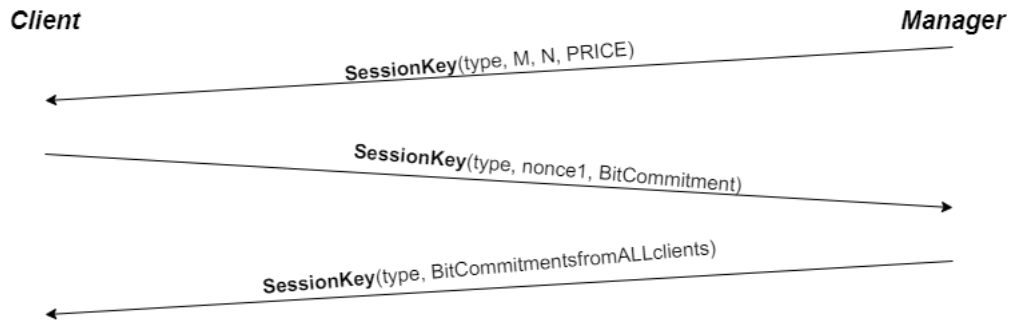

Figure 8: Protocolo de Compremetimento

A primeira mensagem, acima na Figure 8, é enviada pelo Manager e encriptada com a chave de Sessão. Contém 4 campos:

- O primeiro, o tipo da mensagem, que neste caso é 'COMM0'.

- O segundo campo contém M, ou seja, [1,M] são os números possíveis presentes no baralho.

- O terceiro campo contém N, ou seja, cada carta contém apenas N números (cada número entre [1,M]).

- O quarto campo contém PRICE, ou seja, o preço que cada Client deve pagar por cada carta que gerar.

Ao receber a primeira mensagem, o Client desencripta essa mensagem com a chave de Sessão e obtém a informação que necessita para gerar as suas próprias cartas. Depois de o fazer, compromete-se com as mesmas(bit commitment como explicado anteriormente) e é então que envia a segunda mensagem, encriptdada com a chave de Sessão e contendo 3 campos:

- O primeiro, o tipo da mensagem, que neste caso é 'COMM1'.

- O segundo campo é o primeiro valor aleatório presente no Bit Commitment.

- O terceiro campo é o próprio do Bit Commitment.

Ao receber esta mensagem, o Manager desencripta-a com a chave de Sessão e armazena (nonce1, BitCommitment) associados a cada Client. Procede então ao envio da terceira mensagem, depois de obter todos os compremetimentos de todos os Clients, encriptada com a chave de Sessão e contendo 2 campos:

- O primeiro, o tipo da mensagem, que neste caso é 'COMM2'.

- O segundo, um dicionário com todos os primeiros valores aleatórios e bit commitments (nonce1, BitCommitment) associados a cada Client.

O Client ao receber todos os primeiros valores aleatórios e bit commitments dos outros Clients, armazena-os porque, como dito anteriormente, vão ser utilizados posteriormente aquando da computação do vencedor.

2.5 Protocolo do Baralhar do Baralho(Shuffling stage)

Para esta etapa foi desenvolvido o Protocolo de Baralhar o Baralho, que consiste na troca de 3 mensagens, duas por parte do Manager e uma por parte do Client, e que se encontra na Figure 9. O objetivo no final deste protocolo é obter um baralho, baralhado e encriptado(com uma chave simétrica própria para baralhar, criada por cada entidade(shuffling key)) por todos os Clients e pronto para dar início ao jogo propriamente dito.

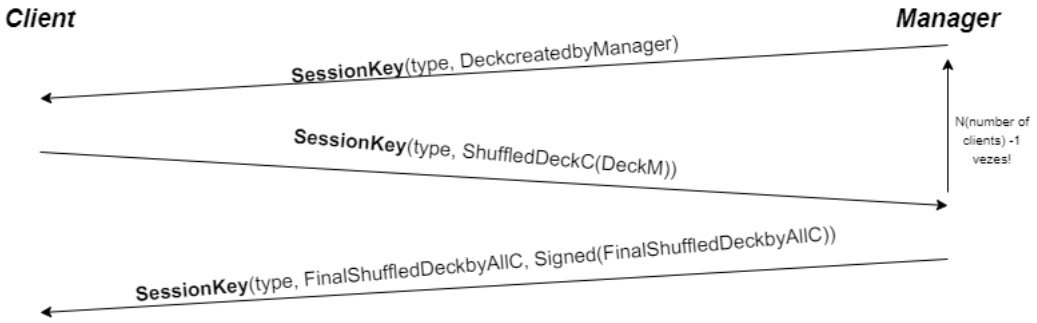

Figure 9: Protocolo de Baralhar

A primeira mensagem, como se pode ver na Figure 9, é enviada pelo Manager e encriptada com a chave de Sessão, e contém 2 campos:

- O primeiro, o tipo da mensagem, que neste caso é 'SHUF0'.

- O segundo campo é o baralho criado pelo Manager.

Cada Client ao receber esta mensagem desencripta-a com a chave de Sessão, deve baralhar o baralho e encriptá-lo(número a número com a sua shuffling key) e voltar a enviar para o Manager. Esta segunda mensagem é encriptada novamente com a chave de Sessão e contém 2 campos:

- O primeiro, o tipo da mensagem, que neste caso é 'SHUF1'.

- O segundo campo é o baralho recebido, baralhado e encrip-

tado(número a número) por si próprio.

O Manager deve enviar a primeira mensagem para todos os Clients, mas apenas envia uma de cada vez, ou seja, só envia a mensagem ao Client2 quando tiver recebido a do Client1. Este processo é demonstrado na Figure 10.

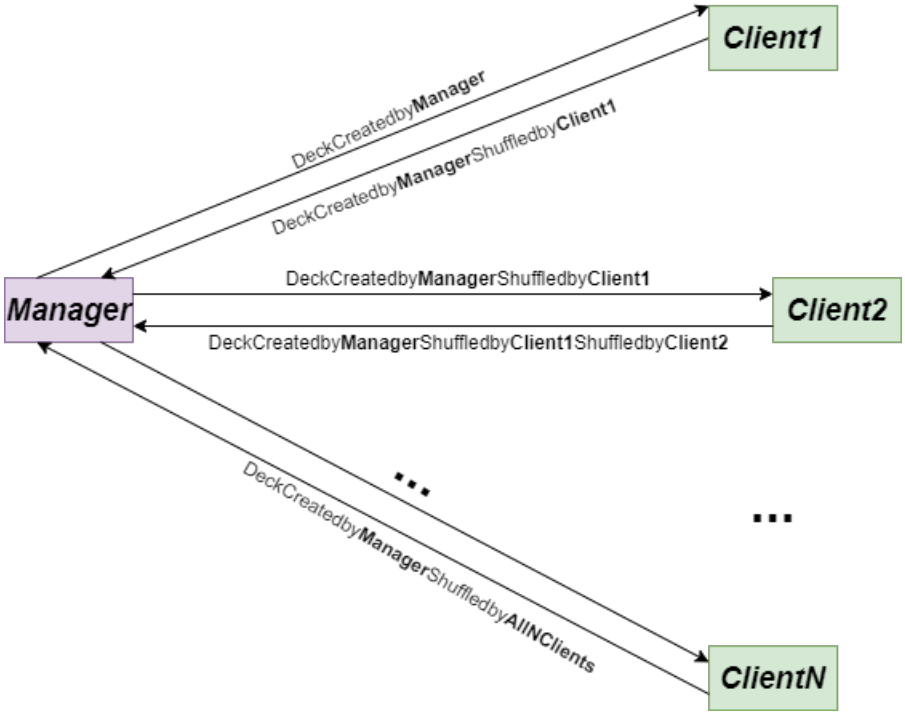

Figure 10: Processo de Baralhar presente nas duas primeiras mensagens

A terceira mensagem só é enviada do Manager para os Clients quando todos tiverem baralhado e encriptado(número a número) o baralho e enviado para o Manager. Esta mensagem é encriptada pela chave de Sessão e contém 3 campos:

- O primeiro, o tipo da mensagem, que neste caso é 'SHUF2'.

- O segundo é o baralho final, baralhado e encriptado por todos os Clients.

- O terceiro, o baralho final assinado pelo Manager.

O Client ao receber esta terceira mensagem desencripta-a com a chave de Sessão, verifica a assinatura do Manager e armazena o baralho, pois mais à frente quando chegar a altura de autenticar a informação recebida e calcular o vencedor irá necessitar do mesmo.

2.6 Protocolo da Revelação dos Segredos(Revelation stage)

Para esta etapa foi desenvolvido o Protocolo da Revelação dos Segredos, que consiste novamente na troca de 3 mensagens, duas enviadas pelo Client e uma enviada pelo Manager. O objetivo deste protocolo para além de todos os Clients obterem os Secrets de todas as outras entidades, é também o Manager chegar à conclusão de quem é o vencedor e posteriormente o Client também.

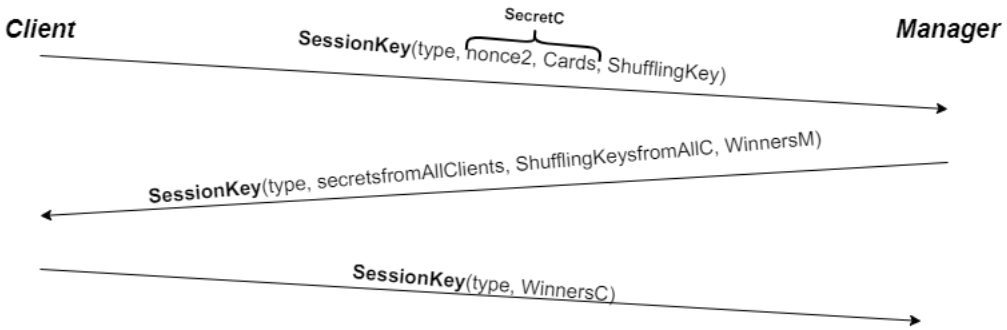

Figure 11: Protocolo de Revelação

A troca de mensagens é visível na Figure 11, e começa com uma primeira mensagem, que parte do Client, é encriptada mais uma vez com a chave de Sessão, e é composta por 4 campos:

- O primeiro, o tipo da mensagem, que neste caso é 'REV0'.

- O segundo campo é o segundo valor aleatório associado ao Bit Commitment.

- O terceiro campo é o conjunto de todas as cartas que cada Client gerou para si mesmo.

- O quarto campo é a chave simétrica usada por cada entidade, individualmente, para encriptar o baralho número a número(shuffling key).

O Manager ao receber esta primeira mensagem, desencripta-a com a chave de Sessão e armazena o segundo valor aleatório e as

próprias cartas associados a cada Client. Esses mesmos atributos, fazem parte do Secret de cada Client. O Manager procede então ao envio da segunda mensagem, encriptada com a chave de Sessão e contendo 4 campos:

- O primeiro, o tipo da mensagem, que neste caso é 'REV1'.

- O segundo campo é um dicionário composto por todos os Secrets a cada Client, ou seja, os Secrets de todos os Clients.

- O terceiro campo é uma lista com todas as shuffling keys de todos os Clients e ainda do Manager.

- O quarto campo contém o vencedor calculado pelo Manager, depois de validar toda a informação recebida anteriormente nos Secrets.

O Client ao receber a segunda mensagem desencripta-a com a chave de Sessão e obtém os Secrets de todos os Clients e as shuffling keys de todos os Clients e Manager, e portanto pode fazer o mesmo que o Manager já fez, ou seja, a computação do vencedor e assim surge o envio da terceira mensagem, que é encriptada com a chave de Sessão e contém dois campos:

- O primeiro, o tipo da mensagem, que neste caso é 'REV2'.

- O segundo campo, o vencedor calculado pelo Client.

2.7 Validação da Informação e Computação do vencedor

Como dito e explicado anteriormente, a validação de toda a informação recebida e a consequente computação do vencedor, quer por parte do Manager, quer por parte de todos os Clients, foi realizada aquando do Protocolo da Revelação dos Segredos(Revelation stage), presente na Figure 11.

A validação da informação é realizada depois de cada entidade obter os valores nonce1, nonce2, BitCommitment, as cartas de cada Client, também as shuffling keys de todos os Clients e o baralho final encriptado pelos Clients. Com base na informação recebida:

- Cada entidade calcula o Bit Commitment de cada Client e compara esse valor com o valor do Bit Commitment recebido, verificando assim a autenticidade(ou não) das cartas de cada Client.

- Com as shuffling keys de todos os Clients, cada entidade desencripta o baralho final baralhado e encriptado por todos os Clients, tantas vezes quantas as shuffling keys que existirem. O baralho é desencriptada pela ordem inversa de encriptação, ou seja, a chave da última entidade a baralhar e a encriptar é a primeira a ser utilizada para desencriptar.

A computação do vencedor é assim realizada tendo em conta que com as cartas de todos os Clients e com o baralho final desencriptado, chega-se ao Client que mais depressa preenche uma das suas cartas tendo em conta a ordenação do baralho.

2.8 Contabilidade correta

Para esta última etapa foi desenvolvido o Protocolo da Contabilidade em que apenas existe o envio de uma mensagem por parte do Manager. O objetivo no final deste protocolo é cada Client vencedor obter um recibo de contabilidade assinado pelo Manager. Um recibo acaba por ser o montante calculado correspondente ao prémio de cada Client vencedor(se cada Client pagou X Euros por cada cartão, o vencedor deve ganhar (P * X)/NR-WINNERS Euros, em que P representa o número de Clients que participaram no jogo e NR-WINNERS o número de vencedores, ficando assim o prémio dividido por todos os vencedores caso haja mais que um).

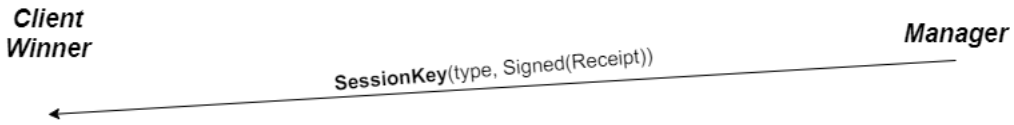

Figure 12: Protocolo de Contabilidade

O envio da mensagem é visível acima, na Figure 12, é encriptada com a chave de Sessão e contém 2 campos:

- O primeiro, o tipo da mensagem, que neste caso é 'ACC0'.

- O segundo, o recibo calculado e assinado pelo Manager.

O Client ao receber esta mensagem, desencripta-a com a chave de Sessão, verifica a assinatura do recibo e armazena-o caso pretenda reclamar o seu valor no futuro.

3 Implementação

3.1 Problemas conhecidos e limitações

Para além de alguma falha de segurança que possa existir e que não tenha sido detetada ou tida em conta, este projeto peca pela não realização de três pontos requeridos no enunciado. São eles a autenticação com cartão de cidadão, a possibilidade de batota e o protesto contra batota. O primeiro deles não foi realizado devido às dificuldades sentidas na tentativa da sua implementação e os outros dois por falta de tempo e consequentemente foi escolhida a estratégia de investir mais noutros pontos.

4 Conclusão

Durante a realização deste projeto foram sentidas algumas dificuldades, como referido anteriormente, tendo várias sido superadas e outras não.

Embora alguns objetivos particulares do projeto não tenham sido atingidos, o objetivo geral foi concluído, pois o jogo do Bingo está a funcionar com implementações de Segurança, como requerido no enunciado. O objetivo geral da UC fica assim também cumprido com alguma satisfação, pois a realização deste projeto permitiu o adquirir de vários conhecimentos ao nível prático.