GABRIEL SCHECHTER SALDANHA MARINHO
LUIS GUSTAVO GONÇALVES GALVÃO DE CASTRO
RODRIGO RISKALLA LEAL

# IMPLEMENTAÇÃO E PROVA DE CONCEITO DO PROTOCOLO SECURETCG: PROTOCOLO DE DETECÇÃO DE TRAPAÇAS EM JOGOS DE CARTAS COLECIONÁVEIS MULTIJOGADORES EM AMBIENTE P2P

São Paulo
2015

GABRIEL SCHECHTER SALDANHA MARINHO
LUIS GUSTAVO GONÇALVES GALVÃO DE CASTRO
RODRIGO RISKALLA LEAL

# IMPLEMENTAÇÃO E PROVA DE CONCEITO DO PROTOCOLO SECURETCG: PROTOCOLO DE DETECÇÃO DE TRAPAÇAS EM JOGOS DE CARTAS COLECIONÁVEIS MULTIJOGADORES EM AMBIENTE P2P

Monografia apresentada à Escola Politécnica de Universidade de São Paulo

São Paulo
2015

GABRIEL SCHECHTER SALDANHA MARINHO
LUIS GUSTAVO GONÇALVES GALVÃO DE CASTRO
RODRIGO RISKALLA LEAL

# IMPLEMENTAÇÃO E PROVA DE CONCEITO DO PROTOCOLO SECURETCG: PROTOCOLO DE DETECÇÃO DE TRAPAÇAS EM JOGOS DE CARTAS COLECIONÁVEIS MULTIJOGADORES EM AMBIENTE P2P

Monografia apresentada à Escola Politécnica de Universidade de São Paulo

Área de Concentração: Engenharia Elétrica com ênfase em Computação

Orientador: Prof. Dr. Marcos Antonio Simplicio Junior

São Paulo
2015

# RESUMO

O objetivo desse estudo é avaliar a viabilidade do emprego do protocolo SecureTCG para obtenção de segurança contra trapaças em jogos de cartas colecionáveis em ambientes par-a-par, ou seja, sem a presença de um servidor confiável. Devem ser considerados fatores como eficácia na prevenção e detecção de trapaças, flexibilidade na adaptação a diferentes cenários encontrados em diversos jogos típicos e eficiência na utilização de recursos computacionais. Essa avaliação se realiza através do desenvolvimento de uma Prova de Conceito, que consiste da implementação de um jogo exemplo em um ambiente de rede com arquitetura par-a-par e suporte a múltiplos jogadores, e que oferece mecanismos para realizar tentativas de trapaça. O protocolo mostra-se bastante robusto na prevenção e detecção de trapaças, atuando de forma transparente aos usuários e sem degradar a fluidez do jogo, indicando a versatilidade e eficiência das funções de resumo criptográfico, em que se baseia, para aplicações relativas à segurança da informação.

Palavras-chave: Segurança da informação. Jogos de cartas colecionáveis. Prevenção e detecção de trapaças. Arquitetura par-a-par. Funções de resumo criptográfico. Prova de Conceito.

# ABSTRACT

The aim of this study is to evaluate the feasibility of the SecureTCG protocol for obtaining security against cheating in trading card games in peer-to-peer environments, i.e. without the presence of a trusted server. It must consider factors such as its effectiveness in preventing and detecting cheating, its flexibility in adapting to different scenarios typically found in many games and the efficient use of computing resources. This assessment takes place through the development of a Proof of Concept, which consists of implementing an example game in a network environment with peer-to-peer architecture and support for multiple players, and provides mechanisms to perform cheating attempts. The protocol proved to be quite robust in cheating prevention and detection, acting transparently to the users and without degrading the game flow, showing the versatility and efficiency of hash functions, in which it is based on, for applications related to information security.



Keywords: Information security. Trading card games. Cheating prevention and detection. Peer-to-peer architecture. Hash functions. Proof of Concept.

# LISTA DE ILUSTRAÇÕES

# SUMÁRIO

# 1 INTRODUÇÃO

## 1.1 Motivação

Com a popularização dos aparelhos móveis, existe hoje um mercado bilionário relativo aos aplicativos de jogos, apresentando receita global de US$ 25 bilhões em 2014, e que em 2015 deve atingir a marca de US$ 30.3 bilhões, superando pela primeira vez o mercado de jogos de console, que deve atingir US$ 26.4 bilhões. Esse crescimento não é limitado a um tipo de mercado, sendo observado tanto nos mercados ocidentais mais "maduros" quanto nos emergentes, e vem atraindo investimentos pesados inclusive das empresas de games mais tradicionais (GAUDIOSI, 2015).

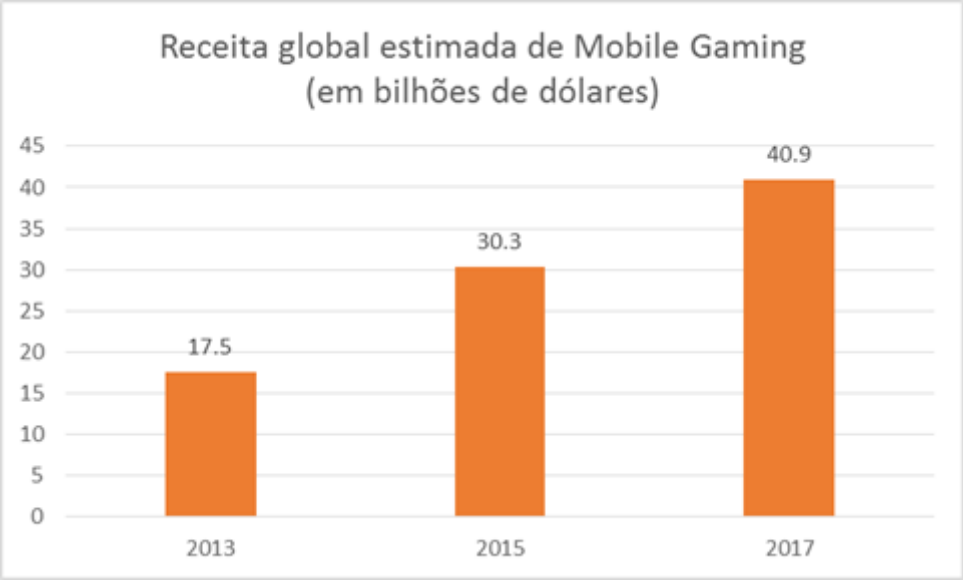


Figura 1 – Receita global estimada de Mobile Gaming

Dentre os jogos dessa nova plataforma, encontram-se categorias anteriormente inexistentes, como os jogos sociais e casuais (ESSENTIAL..., 2014), mas diversos aplicativos de entretenimento são virtualizações para mobile de jogos que já existiam. Dentre estes encontram-se os TCG (Trading Card Games, ou jogos de cartas colecionáveis), assunto esse de escopo do projeto.

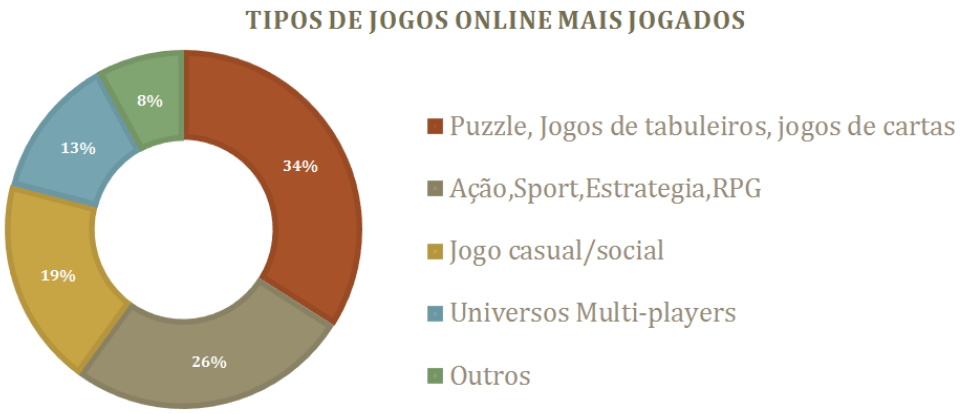


Figura 2 – Tipos de jogos online mais jogados

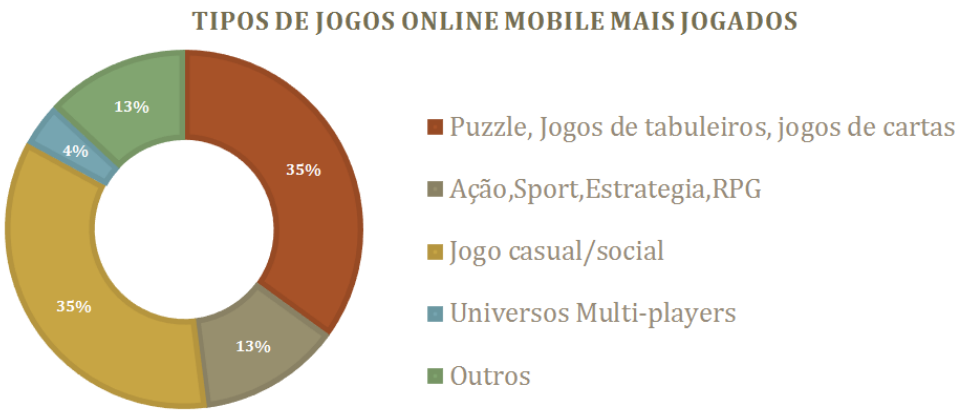


Figura 3 – Tipos de jogos online mobile mais jogados

Os TCGs apareceram a partir de 1993 com o jogo “Magic: The Gathering”, e outros logo foram criados, como “Lord of the Rings CCG”, “Duels Warstorm”, “Yu-Gi-Oh” e “Pokémon”. Esses jogos são similares a jogos de carta tradicionais, mas o número de cartas existentes (que o criador do jogo disponibiliza aos jogadores) não é limitada a 52, como no baralho tradicional. Existe uma variedade imensa de cartas e os jogadores podem comprar essas cartas ou trocar com outros jogadores, montando a sua coleção de cartas, da qual pode selecionar aquelas que desejar para montar um monte de cartas para o jogo (Deck). Essa seleção de cartas por vezes deve seguir algumas regras estabelecidas, como um limite inferior, superior, ou limite de cartas repetidas, entre outras. Existem cartas com diversas habilidades e cada uma tem um

efeito no jogo e a combinação delas pode gerar outros efeitos, assim a escolha de um deck é essencial nesse estilo de jogo.


Figura 4 - Exemplos de jogos de cartas colecionáveis

Assim como o mercado de games online, o mercado dos TCGs é um mercado bilionário, que só no ano de 2013 movimentou US$ 4.1 bilhões, dos quais US$ 1.3 bilhão se deve ao mercado digital (DIGITAL..., 2013). Isso se deve às compras de cartas e subscrições nos jogos digitais e campeonatos (digitais e físicos). Porém, um problema enfrentado nesse tipo de jogo é o método de garantir um jogo honesto entre os jogadores. Por TCGs terem um deck muito diverso do outro e haver certa liberdade da parte do jogador para montar esse deck, identificar trapaças é uma tarefa difícil.

## 1.2 Objetivo

Enquanto em um ambiente com uma TTP (Trusted Third Party ou Terceira Parte Confiável) a detecção de trapaças é simples, em ambientes P2P (peer-to-peer ou par-a-par) ela é bem mais difícil, pois as cartas dos jogadores têm de ser confidenciais e mesmo assim os seus adversários precisam ter certeza que elas estão nos conformes das regras, que a compra de carta do deck seja aleatória, dentre outros requisitos de segurança. Essa abordagem (P2P) requerer soluções mais custosas para a prevenção de trapaças (BETHEA; COCHRAN; REITER, 2011; ROCA, 2005). Essa dificuldade pode acabar levando à insatisfação dos jogadores honesto e consequentemente à sua remoção da subscrição.

A forte participação de jogos TCGs no mercado de jogos digitais fomenta oportunidades que visem garantir maior segurança, eficiência e flexibilidade aos produtos deste setor. Desta forma, este trabalho visa consubstanciar uma prova de conceito do **Protocolo *SecureTCG*** que tem por objetivo garantir que não haja trapaças em um cenário de jogo de cartas colecionáveis em ambiente P2P. Para isto, este trabalho descreve os conceitos necessários para o entendimento do protocolo, bem como as ferramentas a serem utilizadas para a construção da prova de conceito. Essa deve ser desenvolvida de forma modular para fácil integração com sistemas de terceiros (jogos já desenvolvidos).

# 2 DEFINIÇÃO DO CENÁRIO

## 2.1 Cartas colecionáveis e Decks – os “baralhos” dos TCGs

Em meados dos anos 90, surgiu um novo tipo de jogo de cartas, diferente dos jogos de cartas convencionais – que utilizam o baralho tradicional de 52 cartas, por exemplo. Ele era diferente porque o conjunto de cartas existentes no jogo é enorme – e ainda é comum que seja periodicamente expandido – portanto o jogador não pode adquirir todas as cartas de uma vez. Em vez disso, ele deve começar adquirindo baralhos básicos, e depois incrementar sua coleção através de novos pacotes de cartas, ou mesmo através de compra ou troca de cartas individuais. Além disso, essas cartas possuem características individuais, como por exemplo as ações que podem realizar, sua “força” e até mesmo sua raridade. Essas características são utilizadas como critério pelos jogadores para escolher quais cartas irão incrementar suas coleções – normalmente visando adquirir maior competitividade no jogo ou maior exclusividade na coleção.

O que surgiu com isso foi um tipo de jogo que possui cartas colecionáveis, produzidas em massa para viabilizar as coleções e trocas, e que possuam regras para que sejam utilizáveis em um jogo de estratégia (WILLIAMS, 2006). O primeiro jogo de cartas colecionáveis criado – e também o mais bem sucedido – foi “Magic: The Gathering”, inventado por Richard Garfield, e patenteado pela Wizards of the Coast em 1993 (FIRST..., 1993; ROTHAERMEL; KOTHAAND; MOXON, 1998).

Também diferentemente dos jogos de cartas convencionais – como pôquer, por exemplo – o “baralho” utilizado nos TCGs, chamado de deck, deve ser construído por seus jogadores. Cada um utiliza seu próprio deck, construído a partir de um grande conjunto de cartas disponibilizado pelo provedor do jogo, e essa construção está normalmente sujeita a um conjunto de regras, específicas para cada jogo ou mesmo para diferentes contextos – regras diferentes das convencionais para um jogo podem existir em um campeonato, por exemplo.

Seguindo a notação de Pittman e GauthierDickey (2013), chamamos o conjunto de todas as cartas disponíveis de Deck Universal (Du), o conjunto de cartas que o jogador possui de Deck Base (Db) e o conjunto de cartas que o jogador pode usar numa

determinada partida de Deck de Jogo (Dp), tal que Dp ⊂ Db ⊂ Du, conforme pode ser visto no diagrama abaixo:

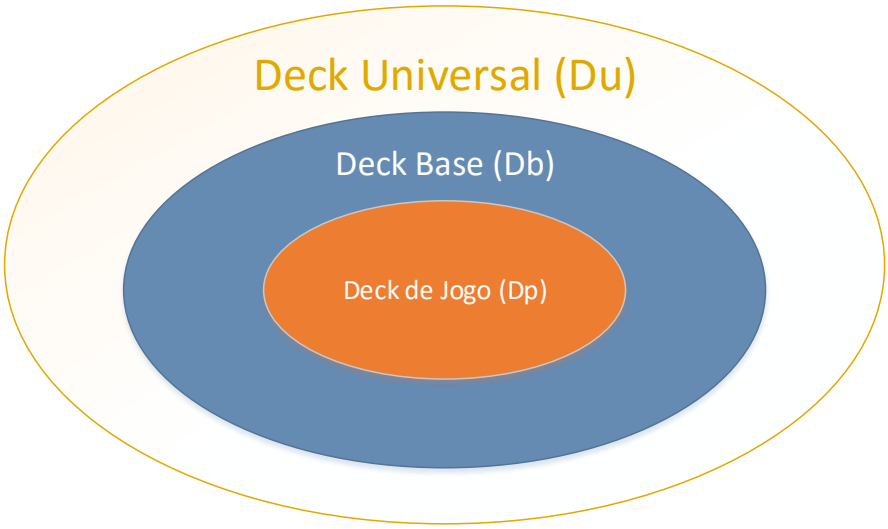


Figura 5 – Hierarquia dos decks em um TCG

O tamanho desses decks pode variar de jogo para jogo, podendo inclusive ser arbitrário em alguns casos, dependendo unicamente da vontade do jogador. Além disso, a construção dos decks pode ter que obedecer a outras regras, como por exemplo limitações sobre o número de cartas repetidas que um deck pode conter. Alguns estilos de jogo comuns que apresentam conjuntos de regras específicas incluem (PITTMAN; GAUTHIERDICKEY, 2013):

- Deck Construído: os jogadores constroem seus Decks de Jogo escolhendo as cartas dentre as que eles possuem, isto é, a partir de seus Decks Base pessoais;
- Deck Uniforme: os jogadores recebem exatamente o mesmo Deck Base, a partir do qual eles podem escolher livremente as cartas para seus Decks de Jogo;
- Deck Selado: são fornecidos aos jogadores conjuntos aleatórios de cartas como seus Deck Base, no lugar das cartas que eles possuem em suas próprias coleções. Eles devem então construir seus Decks de Jogo a partir desses Decks Base. Este estilo razoavelmente comum em torneios, uma vez que leva a partidas que dependem mais de habilidades estratégicas e de jogo do que de poder aquisitivo;
- Deck Selecionado: cada jogador seleciona cartas de um conjunto aleatório de uma forma cíclica, revezando-se a respeito de quem começa a pegar. Mais especificamente, no caso de um jogo com cartas físicas, um jogador recebe um

pequeno conjunto de cartas aleatórias, escolhe uma e depois passa as cartas restantes para o próximo jogador; o processo é repetido até que todas as cartas sejam selecionadas, e então um novo conjunto aleatório é oferecido a todos os jogadores. Obviamente, o primeiro jogador a escolher tem uma vantagem sobre os seus adversários, mas essa vantagem é igualmente oferecida a todos os jogadores nas diferentes rodadas de escolha. As cartas selecionadas desta maneira constituem os Decks Base dos jogadores, a partir do qual os Decks de Jogo podem ser construídos como desejado.

A análise destes estilos mostra que existem basicamente dois tipos de Decks Base: os determinísticos e os aleatórios. Os Decks Construídos e os Uniformes se encaixam na primeira categoria, em que só é necessário assegurar que o jogador está autorizado a usar as cartas no Deck de Jogo empregado em uma partida. Os Decks Selados e Selecionados pertencem à segunda categoria, em que é necessário assegurar a aleatoriedade dos Decks Base de cada jogador. A principal peculiaridade dos Decks Selecionados é que o processo de geração aleatória deve ser repetido algumas vezes e exige que, depois que um jogador escolhe uma carta, o próximo jogador só tome conhecimento das cartas restantes.

## 2.2 Dinâmica típica de uma partida de TCG

A primeira fase de um jogo de Cartas Colecionáveis é a montagem do Deck de Jogo. Isso pode ser feito de diversas formas, como descrito no tópico anterior. Depois de construído o deck de jogo, pode se iniciar a partida propriamente dita. Como em diversos jogos tradicionais de cartas, os TCGs são jogados em turnos. Por isso, quando se diz que são jogos com suporte a grande número de usuários, significa que é suportado um grande número de jogadores participando de partidas diferentes, e não de uma única, pois quanto maior o número de jogadores na mesma partida, mais tempo cada um deve esperar por sua chance de jogar.

A dinâmica dos turnos pode apresentar diversas variações, de acordo com as regras específicas de cada TCG. Apesar dessa variabilidade, algumas ações típicas permitidas a um jogador em seu turno podem ser definidas:

- Comprar cartas, que consiste em pegar determinado número de cartas do próprio deck de jogo e colocá-las na mão, sem revelá-las aos outros jogadores;

- Revelar cartas, que consiste em revelar algumas cartas presentes na mão, colocando-as em jogo se seu efeito for permanente, ou numa pilha de descarte se for temporário;
- Usar as cartas de efeito permanente que já estejam em jogo repetidamente, até que elas saiam de jogo de acordo com as regras do jogo.

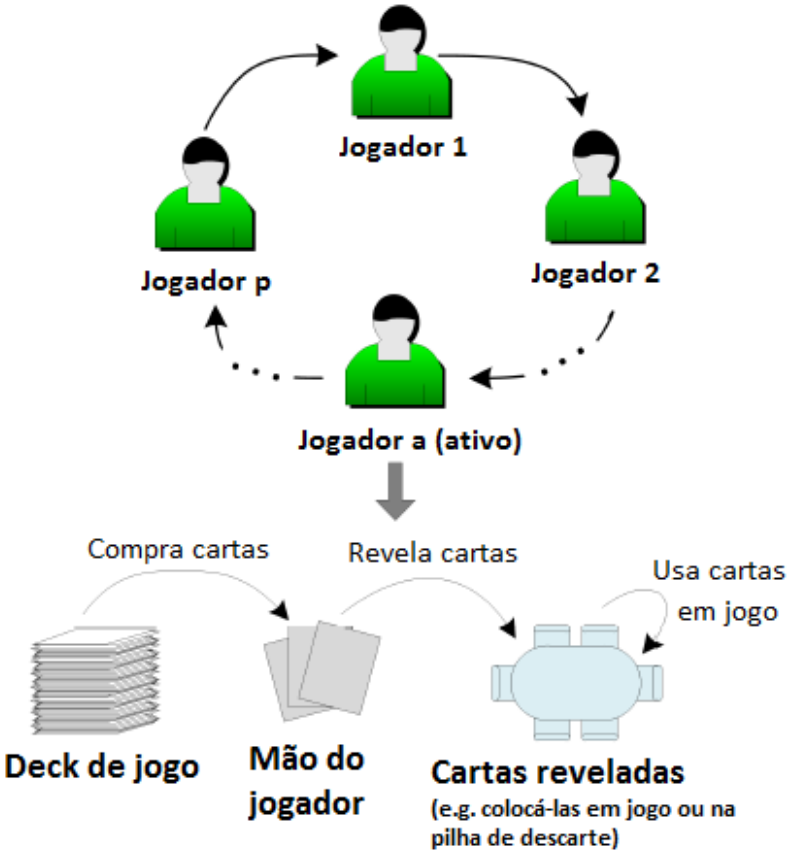


Figura 6 – Dinâmica típica de um turno de uma partida de TCG

A partida termina quando um jogador atinge determinado objetivo, sendo um dos mais comuns reduzir a zero o número de “pontos de vida” de todos os outros jogadores da partida ou ser o último jogador a ainda possuir cartas fora da pilha de descarte.

## 2.3 Ambientes virtuais de TCG

Jogos multijogador em rede podem ser implementados usando várias abordagens diferentes, que podem ser categorizadas em dois grupos: impositivas, que pressupõem a existência de uma entidade central (servidor) para controlar o estado

do jogo entre as entidades finais (clientes, nesse caso); e não impositivas, onde não há entidade central e cada entidade final (peer, nesse caso) é responsável por controlar seu estado de jogo (BEVILACQUA, 2013).

Para a implementação de um TCG, as abordagens mais comuns são, no grupo impositivo, a TTP utilizando a arquitetura cliente-servidor, enquanto no grupo não impositivo é a P2P.

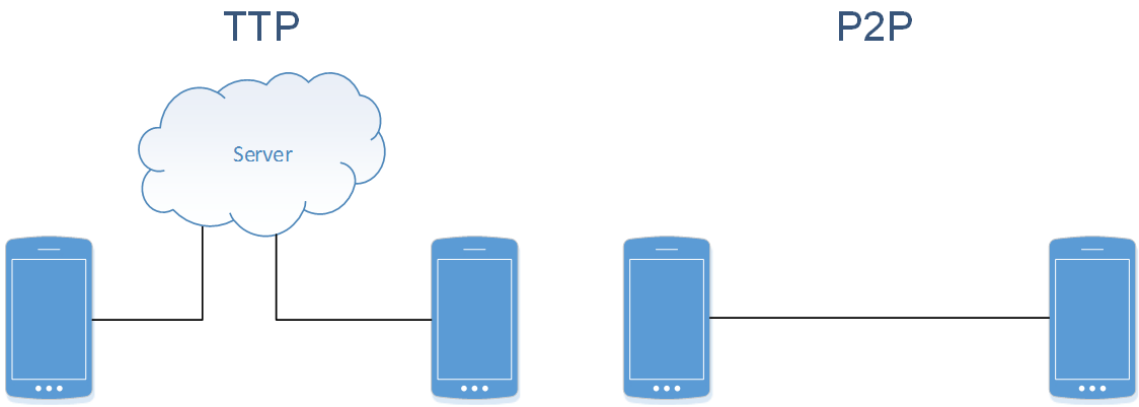


Figura 7 – Ambientes de jogo dependente de um TTP (servidor) e P2P

O cenário TTP é definido pela presença de um servidor que atua como intermediário entre os clientes que participam como usuários do jogo, e cada cliente conectado ao servidor recebe dados dele constantemente, criando localmente uma representação do estado do jogo. Se um cliente executa uma ação, essa informação, antes de ser efetivada, é enviada para o servidor. O servidor verifica se as informações estão corretas e, em seguida, atualiza seu estado jogo. Depois, ele propaga a informação a todos os clientes – inclusive para o que realizou a ação – para que eles possam atualizar o seu estado de jogo (BEVILACQUA, 2013). Nessa situação, fica evidente que o servidor é capaz de garantir, de maneira relativamente simples, dado o controle que possui sobre o desenrolar do jogo, a honestidade do jogo.

Já no caso do cenário P2P, como não há a presença de um servidor, cada peer deve mandar dados para todos os outros peers e receber dados deles. Portanto, nessa abordagem, cada peer deve controlar seu próprio estado de jogo, comunicando cada mudança e ação importante aos outros. Consequentemente, o jogador vê um cenário de jogo composto pelas suas entradas no jogo e por uma simulação de todas as entradas controladas pelos outros jogadores. Diz-se uma simulação, pois as ações

dos demais jogadores, diferentemente das do próprio jogador, podem representar não o estado real, mas uma simplificação. Isso porque não é desejável que todos saibam as ações dos outros jogadores, apenas que elas ocorreram (BEVILACQUA, 2013).

Levando-se em conta esses fatores, nota-se que os jogos de TCG apresentam ótima aptidão a uma implementação P2P, pois a questão do controle do estado de jogo neles é bastante simplificada, devido à dinâmica de turnos e ao fato de as mensagens que precisam ser trocadas entre os peers serem em geral bastante enxutas.

O uso da arquitetura P2P é motivado principalmente pelos menores custos, menores dificuldades de implantação, pela disponibilidade menos dependente de um ponto único (servidor) e pela maior escalabilidade em relação à arquitetura cliente-servidor com TTP. As evoluções dos jogos online, que usuários agora jogam em dispositivos heterogêneos – seus PCs e consoles, mas também em seus smartphones, apenas com uma conectividade intermitente – motivam cada vez mais a utilização de arquiteturas descentralizadas. Além disso, fora os jogos online clássicos, jogos de “realidade mista", onde a localização física dos jogadores impacta no jogo, parecem perfeitamente adequados para rodar sobre esse tipo de arquitetura (NEUMANN, 2007).

Por outro lado, a descentralização do controle do jogo faz com que a abordagem P2P apresente, entre outros, maiores desafios relativos a detecção e prevenção de trapaças, que é uma questão de crescente importância, à medida que os jogos vão se tornando mais populares e complexos, e fundamental para assegurar que jogadores honestos continuem interessados em jogar. Torna-se necessária, portanto, a presença de um protocolo executado nos peers de forma a garantir que as trapaças não ocorram ao longo da partida.

## 2.4 Arquitetura básica de um TCG em ambiente P2P

Como é discutido por Pittman e GauthierDickey (2013), a arquitetura básica de jogos de cartas colecionáveis P2P é composta por duas entidades básicas: servidor de jogo e jogadores.

### 2.4.1 Servidor de jogo

O servidor do jogo é responsável pela definição de quais cartas estão disponíveis no jogo, informando os usuários sobre novas edições quando são liberadas, e também pelo gerenciamento de contas de usuários. No entanto, as interações do servidor com os jogadores devem idealmente ocorrer apenas entre partidas, enquanto os jogos em si devem ser tratados de forma puramente P2P. Mais especificamente, o servidor de jogo em um TCG P2P pode assumir as seguintes responsabilidades:

- Geração de IDs únicos: toda carta e jogador está associada a um identificador único. Um novo identificador do usuário deve ser gerado sempre que uma nova conta é criada – poderia ser, por exemplo, o nome de usuário do jogador no sistema. O identificador de cartas, por outro lado, deve identificar univocamente uma carta dentre todas as existentes e também distinguir duas cartas idênticas pertencentes a diferentes jogadores – poderia ser composto, por exemplo, de um número único de 128 bits concatenados com um contador de 128 bits incrementado toda vez que uma cópia dessa carta é comprada por algum jogador. Estas identificações são utilizadas em todas as operações que requerem a identificação de cartas ou jogadores, tais como quando uma carta é jogada durante uma partida;
- Assinatura de identidades de jogadores: cada usuário recebe um certificado digital assinado pelo servidor – portanto, eles são capazes de se identificar de forma segura perante outros usuários que utilizam seus pares de chaves público/privadas;
- Assinatura de propriedade de cartas: quando um jogador compra cartas do provedor do jogo, o servidor é responsável por adicionar as cartas correspondentes à conta do jogador e por fornecer uma assinatura que comprove esta propriedade e a data em que a carta foi comprada. Um jogador está autorizado a jogar apenas com cartas para os quais exista uma assinatura digital válida que associa a carta com o ID do jogador;
- Intermediação de negociação de cartas: Quando os jogadores querem trocar cartas, o servidor deve atuar como um intermediário confiável, provendo uma assinatura ao novo proprietário com o instante da negociação. Um grande problema que surge neste cenário de negociação é que o antigo proprietário

pode manter uma cópia da carta original e sua assinatura, utilizando-a posteriormente apesar de não ser mais o seu proprietário;

- Estabelecimento de conexão: o servidor pode agir como um ponto de encontro central, ajudando jogadores a conectar-se uns aos outros antes de a partida começar e permitindo-lhes superar inconveniências da rede, tais como NAT e firewalls;
- Auditoria do jogo: depois que uma partida termina, os jogadores podem querer fornecer informações sobre o seu resultado para o servidor para, por exemplo, resgate de prêmios ou para fins de classificação (ranking). Em alguns casos, também pode ser desejável provar a terceiros que um jogador tentou trapacear durante a partida, o que afetaria a sua reputação no sistema.

Em um cenário de jogo real, a existência de servidores que sejam capazes de cumprir suas responsabilidades de maneira satisfatória é crucial para o sucesso da plataforma e a manutenção de sua atratividade aos seus jogadores. Para os efeitos do estudo dos mecanismos de detecção de trapaças durante as partidas propostos pelo protocolo SecureTCG, que são o foco da Prova de Conceito desenvolvida nesse projeto, no entanto, a modelagem de um servidor e seus mecanismos foge do escopo e será, portanto, deixada como tópico para trabalhos futuros.

### 2.4.2 Jogadores

Os TCGs online P2P são disputados por dois ou mais jogadores. Numa partida, os jogadores estabelecem uma conexão usando qualquer canal de comunicação disponível, como a Internet ou uma conexão ad hoc (e.g., Bluetooth), que pode ter o servidor de jogo atuando como intermediador na sua inicialização. O servidor pode, opcionalmente, atuar como ponto de encontro e como intermediador para estabelecimento da conexão inicial entre os jogadores.

Em cenários P2P, os jogadores devem ser capazes de controlar conjuntamente toda a dinâmica da uma partida, mandando e recebendo dados constantemente entre si, podendo assim controlar o estado de jogo, comunicando cada mudança e ação importante aos outros. Consequentemente, a detecção de trapaças também fica sob responsabilidade de todos os jogadores. Os mecanismos propostos pelo protocolo

SecureTCG para isso, que são o alvo do estudo da Prova de Conceito desenvolvida, estão descritos na seção “Protocolo Selecionado”.

## 2.5 Definições do TCG exemplo

Para a Prova de Conceito desenvolvida neste projeto, viu-se a necessidade do desenvolvimento de um jogo TCG exemplo, que apresentasse um conjunto de regras e uma dinâmica que o tornassem minimamente jogável, e ao mesmo tempo fossem suficientes para permitir a simulação de trapaças para exercitar os mecanismos de detecção propostos pelo protocolo SecureTCG.

As entidades básicas do TCG exemplo são:

- Cartas: existem no total 60 cartas definidas no jogo (constituindo portanto o Deck Universo), e todas são do mesmo tipo, e apresentam duas características fundamentais: pontos de ataque e pontos de vida. A primeira determina o quanto um ataque dessa carta remove dos pontos de vida de outra carta ou de um jogador adversário atacados. A segunda, obviamente, determina quantos pontos de vida a carta possui, valor que deve ser decrementado a cada ataque sofrido por outra carta;
- Decks: conforme visto na especificação das cartas, o Deck Universo definido é composto por 60 cartas diferentes. O tamanho do Deck Base estipulado é de 40 cartas, e segue o estilo de Deck Selado, ou seja, deve ser construído aleatoriamente a partir das cartas disponíveis no Deck Universo. Por fim, o Deck de Jogo deve ser composto por 20 cartas que devem ser escolhidas pelo jogador a partir das cartas disponíveis no Deck Base;
- Jogador: a característica fundamental dos jogadores são os pontos de vida. Cada jogador apresenta, ao início da partida, 2000 pontos de vida, que devem ser decrementados a cada vez que eles sofrerem um ataque.

Com isso, tem-se que as etapas de construção dos Decks Base e determinação da ordem dos turnos dos jogadores (aleatória) serão automatizadas e não sofrerão interferência dos jogadores. Disponibilizados os Decks Base, os jogadores devem construir seus Decks de Jogo.

Quando a partida propriamente dita tiver início, ela deverá seguir uma dinâmica próxima da dinâmica de turnos padrão dos TCG. Os turnos, por sua vez, devem

consistir de três operações que só podem ser realizadas no máximo uma vez por turno, na ordem aqui apresentada, e cada carta só pode estar envolvida em uma operação por turno. Essas operações são:

- Compra de carta: o jogador pega uma carta de seu Deck de Jogo e a coloca em sua mão, sem revelá-la aos outros jogadores;
- Revelação de carta: o jogador escolhe uma das cartas em sua mão e a coloca em jogo, revelando-a agora a todos os jogadores. A partir do fim do turno corrente, essa carta poderá ser usada para atacar (no próximo turno do jogador) ou para defender, ou seja, sofrer o dano proveniente de um ataque de uma carta de outro jogador;
- Ataque: o jogador escolhe uma de suas cartas em jogo (já reveladas) para realizar um ataque, e determina qual jogador adversário deseja atacar. O jogador atacado deve, então, escolher uma única de suas cartas em jogo, se tiver, para sofrer o dano proveniente do ataque, ou seja, ter seus pontos de vida decrementados do valor dos pontos de ataque da carta atacante. Caso esse valor seja maior ou igual ao total de pontos de vida da carta atacada, a carta “morre” (sai de jogo e vai para uma pilha de descarte), e qualquer dano excedente deve ser decrementado dos pontos de vida do jogador atacado. Caso o jogador atacado não possua cartas em jogo ao sofrer o ataque, seus pontos de vida devem ser decrementados do valor total dos pontos de ataque da carta atacante;
- Fim de turno: o jogador sinaliza, através de um botão, que deseja encerrar seu turno, ou o turno se encerra automaticamente caso o jogador já tenha realizado todas as operações permitidas.

O término do jogo se dá quando restar apenas um jogador com vida diferente de zero ou quando restar apenas um jogador com cartas ativas (no Deck de Jogo, na mão ou em jogo), o que ocorrer primeiro.

# 3 TRAPAÇAS

Trapaças são ações desonestas com o intuito de obter alguma vantagem indevida em alguma situação em que há competição. Especificamente no caso dos TCGs, as trapaças consistem em ações que desobedeçam às regras definidas pelo jogo ou que forneçam informações indevidas a quem as executa, proporcionando-lhe uma vantagem indevida e desleal em relação aos oponentes do jogo. Elas podem ser divididas em duas categorias: Trapaças Sigilosas e Trapaças de jogabilidade.
Trapaças de jogabilidade são intrínsecas do jogo em questão e podem ser facilmente reconhecidas por um jogador que saiba as regras do jogo. Essa categoria de trapaça não será tratada pelo sistema tratado nesse documento. Ou seja, o sistema irá implementar um jogo em conformidade com a seção de Definições do TCG Exemplo sem qualquer tipo de verificação de trapaças desta categoria (alterações no comportamento do sistema acessando o código, ou por alteração de memória durante execução) e não será implementado nenhum módulo de simulação de trapaças dessa Categoria. Alguns exemplos de possíveis trapaças são:

- Aumento da vida do jogador desonesto;
- Redução da vida de um jogador pelo jogador desonesto;
- Alteração dos efeitos e habilidades de uma carta (vida, ataque, magia, entre outros);
- Realização de mais compras do que o máximo permitido pelas regras em um turno;
- Realização de mais ataques do que o máximo permitido pelas regras em um turno.

Trapaças Sigilosas, por outro lado, são difíceis de serem identificadas pelos usuários sem um sistema de segurança. Elas devem ser:

**Prever ou até mesmo controlar a ordem em que ele mesmo ou algum outro jogador retira as cartas do Deck de Jogo**

Um jogador não deve ser capaz de prever ou controlar a ordem das cartas no seu deck. Isso em um jogo real é garantido por todos os jogadores, pois todos se certificam

que os decks foram embaralhados corretamente e que a compra da carta se deu de forma correta. Porém, um sistema digital no qual não existe o deck físico e esse é controlado apenas pelo seu dono é análogo a um jogador embaralhar as cartas e realizar suas compras em baixo de uma mesa. Portanto, se faz necessário um módulo de segurança para garantir a honestidade.

**Adulterar o deck de jogo durante a partida, ou seja, comprar ou revelar cartas que não estavam presentes no deck de jogo no início da partida**

Um jogador não deve ser capaz de inserir cartas em seu deck depois do início do jogo. Isso em um jogo real é garantido por todos os jogadores, pois todos se certificam que os decks foram criados conforme as regras e que durante o jogo ninguém inseriu cartas no deck. Porém, um sistema digital no qual não existe o deck físico é análogo a um jogador ter seu deck em baixo de uma mesa junto de diversas outras cartas que ele poderia pegar. Portanto, se faz necessário um módulo de segurança para garantir a honestidade.

**Tomar conhecimento antes do momento de compra (para o caso das cartas do próprio deck de jogo) ou de revelação (para os outros casos) das cartas presentes em qualquer um dos decks de jogo da partida ou na mão de qualquer oponente**

Um jogador deve descobrir quais cartas existem no deck de seus adversários apenas quando elas forem reveladas na mesa. Isso em um jogo real é garantido, pois todos se certificam que os decks foram embaralhados corretamente, que a compra da carta se deu de forma correta, mas mesmo assim, não sabem qual carta o jogador comprou. Porém, em um sistema digital, os jogadores precisam saber as cartas dos demais jogadores para poderem se certificar de que não houve trapaças na compra e embaralhamento.

**Revelar cartas do seu deck de jogo sem tê-las comprado, ou seja, sem que elas estivessem em sua mão**

Um jogador não deve ser capaz de prever ou controlar a ordem das cartas no seu deck. Isso em um jogo real é garantido por todos os jogadores, pois todos se certificam que os decks foram embaralhados corretamente e que a compra da carta se deu de forma correta. Porém, um sistema digital no qual não existe o deck físico e esse é controlado apenas pelo seu dono é análogo a um jogador embaralhar as cartas e realizar suas compras em baixo de uma mesa. Portanto, se faz necessário um módulo de segurança para garantir a honestidade.

**Entrar em conluio com outros jogadores, em caso de partidas com mais de dois jogadores, para obter informações indevidas ou mesmo influenciar em aspectos do andamento da partida que não deveriam sofrer interferências dos jogadores, como a ordem das cartas do deck de jogo de um adversário**

Um jogador não deve ser capaz de prever ou controlar a ordem das cartas no seu deck. Isso em um jogo real é garantido por todos os jogadores, pois todos se certificam que os decks foram embaralhados corretamente e que a compra da carta se deu de forma correta. Porém, um sistema digital no qual não existe o deck físico e esse é controlado apenas pelo seu dono é análogo a um jogador embaralhar as cartas e realizar suas compras em baixo de uma mesa. Portanto, se faz necessário um módulo de segurança para garantir a honestidade.

Protegendo os jogadores honestos das Trapaças silenciosas permite que o jogo ocorra de forma honesta, pois as demais regras poderão ser garantidas pelos próprios jogadores e pelos sistemas destes (um software que implemente o jogo sabe as regras do jogo e pode identificar trapaças de jogabilidade), possibilitando o reconhecimento e encaminhamento das informações a uma entidade responsável para devidas providencias e punições.

# 4 PROTOCOLO SELECIONADO

O protocolo selecionado para a implementação da prova de conceito é o SecureTCG, pois atende aos requisitos gerais enumerados anteriormente com baixo custo computacional. O protocolo SecureTCG é baseado na associação de um hash único por carta envolvida no jogo e no sequenciamento de cadeias de hash refletindo as ocorrências incrementais do jogo. Dessa forma, o protocolo garante que caso algum dos jogadores participantes cometa alguma trapaça dos tipos enumerados, está será detectada e as medidas pré-estabelecidas pelo sistema serão tomadas. Para a visualização dos eventos gerados pelo protocolo durante a atividade da prova de conceito, será implementada uma interface de Log para a melhor compreensão do funcionamento do protocolo.

## 4.1 Protocolo SecureTCG

O protocolo selecionado pode ser dividido em duas fases independentes: a geração do Deck Base e Proteção contra trapaças. Essa última pode ainda ser dividida em outras duas partes: a fase de difusão inicial de informações e a fase de jogo.
A fase de geração de Deck Base é opcional, pois, como comentado na secção que detalha a dinâmica de um jogo de cartas colecionáveis, o Deck Base existe apenas em alguns campeonatos. Porém, por questão de completude, o protocolo selecionado e a implementação descrita neste documento abrangem essa fase.

### 4.1.1 Geração do Deck Base

Nesta fase os provedores do jogo (organizadores dos campeonatos reais ou os softwares que simulam esses campeonatos) geram um conjunto aleatório de cartas sem beneficiar nenhum dos jogadores que irão montar seus decks de Jogo inserindo cartas melhores para um do que para outro. No mundo real, a honestidade desta distribuição aleatória é garantida por meio de uma comissão organizadora, já em um sistema digital é necessário um mecanismo que garanta a honestidade, ainda mais se essa geração aleatória for realizada localmente por cada aplicação de cada jogador

sem passar por uma terceira parte confiável (TTP). E esse problema se intensifica no cenário que os jogadores não podem saber quais cartas estão sendo selecionadas pelos demais.

**Descrição do mecanismo**

Dado que todos os jogadores concordem em utilizar tamanho do Deck Base sendo *base*:

1. Todos devem gerar uma sequência com *base* números randômicos $v_i^1$, ..., $v_i^{base}$. E mais um número aleatório $u_i$ que irá influenciar na geração dos decks Bases dos demais jogadores;
2. Depois cada jogador calcula o $UCommit_i = Hash(u_i)$ e $VCommit_i = [Hash(v_i^1), ..., Hash(v_i^{base})]$. E encaminha $UCommit_i$ e $VCommit_i$ para todos os demais jogadores;
3. Após o jogador $P_i$ ter recebido o $UCommit_j$ e $Vcommit_j$ de todos os jogadores, este deve enviar aos demais o número aleatório $u_i$. Assim que o jogador recebe um $u_i$ ele verifica se o $UCommit_i$ enviado no segundo passo coincide com o $Hash(u_i)$. Se não coincidir o protocolo já para aqui;
4. Quando o jogador $P_i$ tiver recebido todos os $u_j$ e verificado se o $Hash(u_j)$ batem com o $UCommit_j$, ele irá calcular uma semente randômica $seedB_i = Hash (P_i || u_1 || ... || u_p)$ que será utilizada pelo algoritmo prf para gerar uma sequência *Rand* com *base* números aleatórios $[r_i^1,..., r_i^{base}]$;
5. Então os jogadores utilizam seus valores randômicos privados $v_i^1$, ..., $v_i^{base}$, e realiza uma operação “ou exclusivo” (XOR) com seu número randômico público $r_i^1$, ..., $r_i^{base}$ para gerar um número $ref_i^j$ utilizado para selecionar uma carta do Deck Universo e colocar no Deck Base com $Id = 1 + (ref_i^j \bmod univ)$, onde *univ* é o número de cartas no Deck Universo.

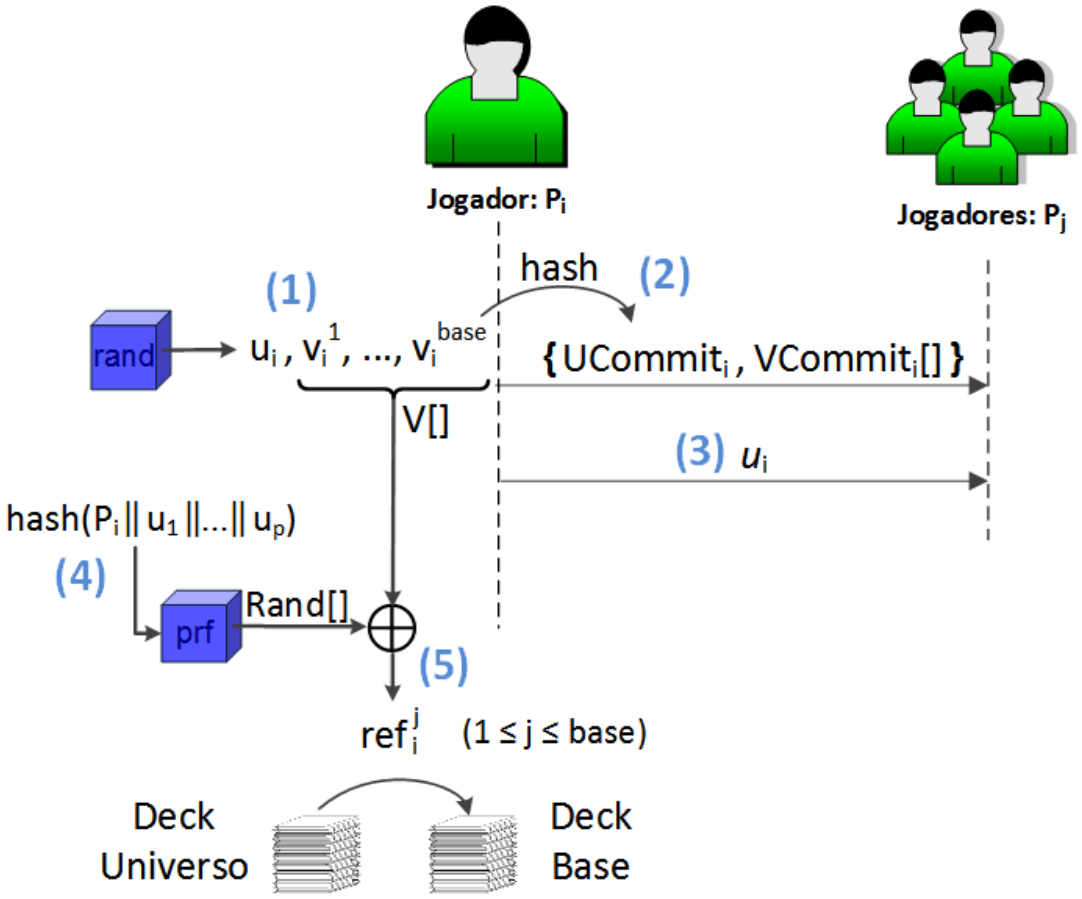


Figura 8 – Geração de Deck Base do protocolo SecureTCG

Depois de terminado o protocolo, cada jogador deve guardar algumas informações para poder verificar posteriormente se não houve trapaças na geração do deck Base. Os valores randômicos privados $v_i^1$, ..., $v_i^{base}$ devem ser guardados junto das cartas selecionadas do deck Base, pois assim, quando o jogador mostrar a carta no jogo $v_i^k$ para os outros jogadores, esses poderão verificar se ($v_i^k \oplus r_i^k$) mod *univ* = Id.

E como é possível deduzir pelo cálculo realizado a cima, os jogadores precisam armazenar os $r_j^k$ (vetor *Rand*) de todos os jogadores, porém se desejar diminuir o espaço necessário para armazenamento em detrimento de um processamento maior, é possível armazenar apenas o $seedB_j$ de cada jogador, pois é possível gerar o vetor *Rand* através da função prf(seedB).

### 4.1.2 Inicialização do jogo e Construção do Deck de Jogo

Anteriormente ao início do jogo, existe uma parte do protocolo de inicialização que garante que, por mais que o deck de jogo escolhido por cada jogador seja de conhecimento apenas deste jogador, caso este revele alguma carta que não estava inicialmente no Deck de Jogo, uma trapaça será detectada. Assim que as cartas que irão constituir o deck de jogo sejam escolhidas por cada jogador, esta etapa do protocolo é iniciada.

**Descrição do mecanismo**

Dado que todos os jogadores concordem em utilizar um tamanho do Deck de Jogo e que as cartas que irão constituir o deck de jogo sejam escolhidas por cada jogador:

1. O jogador $P_i$ associará a cada uma das cartas (com um total de $d_i$ cartas) do Deck de Jogo um valor randômico $mask_i^j$ ($1 < j < d_i$). Na prática, cada carta do Deck de $P_i$ é representada por $c_i^j = (mask_i^j, ID_i^j)$ de forma que $ID_i^j$ é o identificador universal da carta e $Deck_i$ do jogador $P_i$ possa ser descrito como $Deck_i = [c_i^1, .... c_i^{di}]$. Este vetor do $Deck_i$ é armazenado por $P_i$ e mantido em segredo até que alguma carta deva ser revelada. Neste ponto, a ordem em que as cartas se encontram no vetor $Deck_i$ não influenciam a probabilidade de serem sacadas do Deck de Jogo mais tarde;
2. O jogador $P_i$ calcula um hash para cada carta contida no $Deck_i$, de forma que para cada carta $c_i^j$ é calculado seu Hash $h_i^j = Hash(mask_i^j \,||\, ID_i^j)$ gerando uma sequência $Pile_i = [h_i^1, ..., h_i^{di}]$. É importante notar que o tamanho de $h_i^j$ deve ser suficientemente grande para garantir que a probabilidade de duas cartas possuírem hashes iguais seja muito baixa;
3. Cada jogador envia a sequência $Pile_i$ e $d_i$ aos demais jogadores, se comprometendo com a sequência e a ordem do Deck de Jogo;
4. Após a recepção das mensagens de todos os jogadores, $P_i$ produz e guarda um valor randômico $seedC_i$, de tal forma que ele seja secreto durante todo o intervalo do jogo. Após esta etapa, o jogador produz e armazena uma cadeia de hashes com comprimento $l_i$ igual a unidade mais a somatória do total de cartas dos demais jogadores. Esta cadeia de hashes é calculada tomando-se o $link_i^0$ como

seed$C_i$ e calcula os demais hashes da cadeia como $link_i^k = Hash(link_i^{k-1})$ de tal forma que $1 \leq k \leq l_i$;

5. Cada jogador $P_i$ envia o último valor de sua cadeia de hash $tail_i = link_i^{l_i}$;
6. O jogador $P_i$ armazena as seguintes informações acerca dos demais jogadores da partida: a cauda da cadeia de hashes $tail_b$; a sequência de hashes para as cartas do deck de jogo $Pile_b$; os hashes para as cartas na mão do jogador, $Hand_b$ que é inicialmente vazia e finalmente os hashes relativos às cartas já utilizadas pelo jogador $Used_b$, que também é vazia inicialmente.

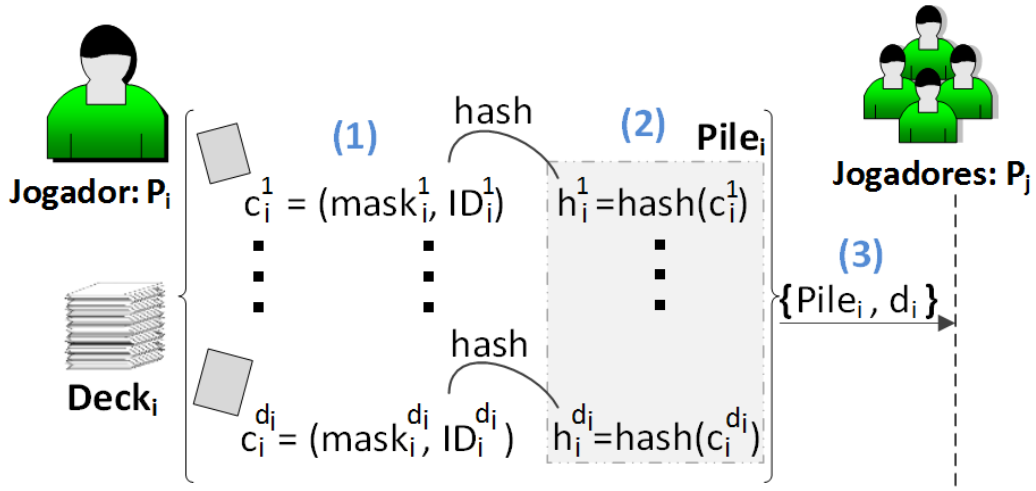


Figura 9 – Etapas 1, 2 e 3 da inicialização do jogo e construção do Deck de Jogo do protocolo SecureTCG

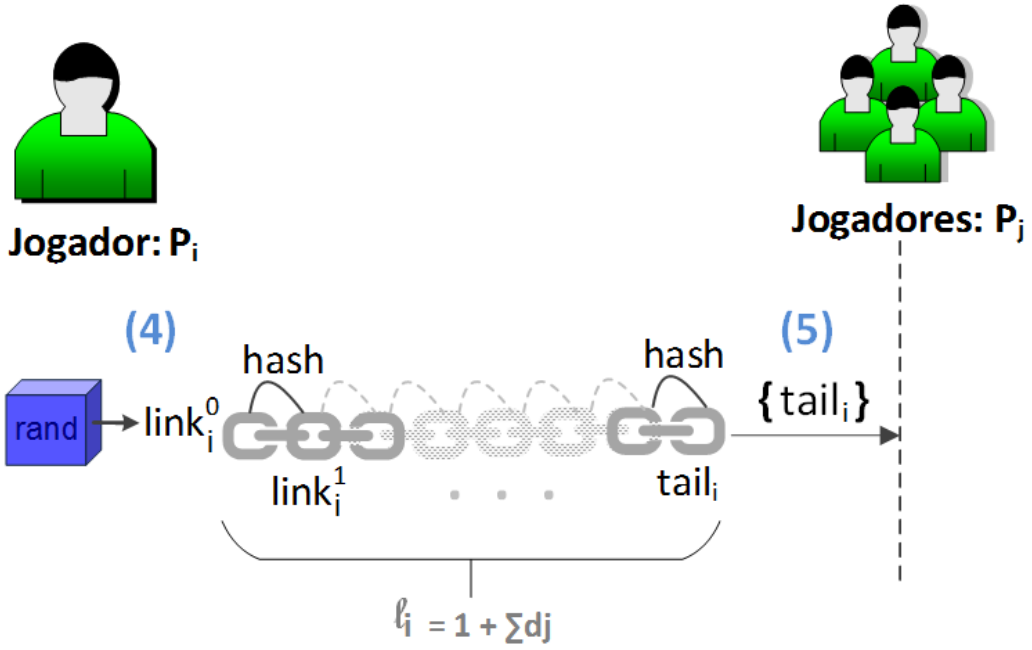


Figura 10 – Etapas 4 e 5 da inicialização do jogo e construção do Deck de Jogo do protocolo SecureTCG

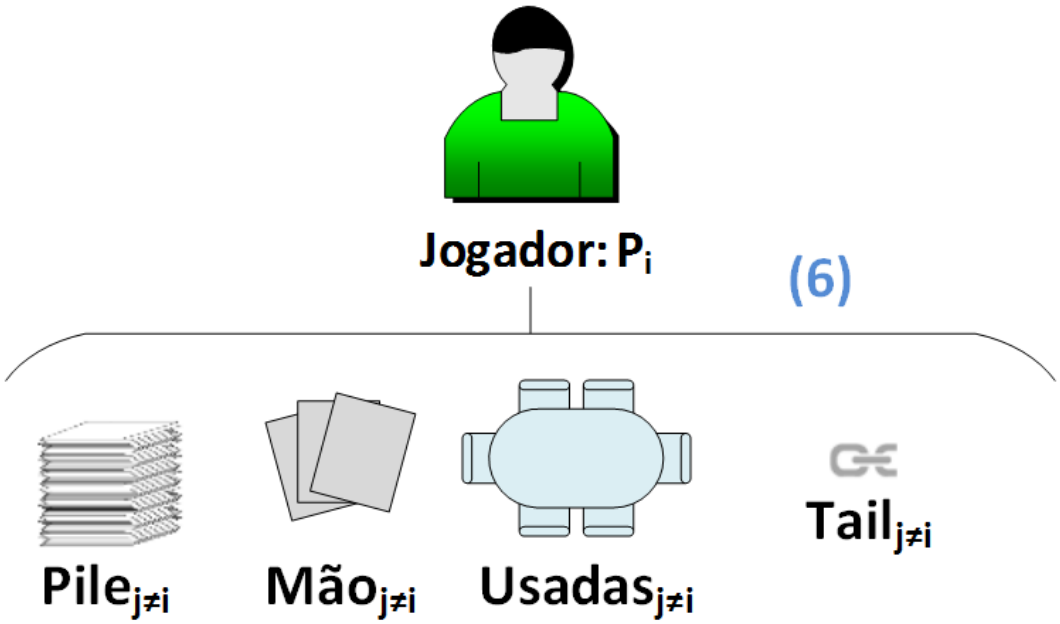


Figura 11 – Etapa 6 da inicialização do jogo e construção do Deck de Jogo do protocolo SecureTCG

### 4.1.3 Retirando uma carta do Deck de Jogo para a Mão

No cenário em que o jogador $P_a$ queira retirar uma carta do Deck de Jogo, não deve ser permitido que este seleciona qualquer carta que queira. Dessa forma, torna-se necessário aplicar o mecanismo descrito abaixo para garantir que a próxima carta sacada pelo jogador $P_a$ tenha probabilidade uniformemente distribuída com relação às demais cartas do Deck de Jogo.

**Descrição do mecanismo**

1. Cada jogador, exceto $P_a$, envia seu próximo valor da cadeia de hashes para todos os demais jogadores;
2. Todos os jogadores verificam se a mensagem enviada pelos demais jogadores corresponde a um valor válido da cadeia de hashes do outro jogador. Isto é feito verificando se $Hash(link_i^k) = link_i^{k+1}$, de forma que $link_i^{k+1}$ é valor atualmente associado à cauda da cadeia de hashes de $P_i$ ($i \neq a$). Se esta situação for validada, o jogador $P_a$ atualiza a informação acerca do valor da cadeia de hashes de $P_i$ para $link_i^k$. Caso a validação falhe, uma trapaça é detectada e o procedimento adequado é disparado para este cenário;
3. Para que se determine o índice da carta a ser retirada do Deck de Jogo de $P_a$, todos os jogadores executam a seguinte conta para calcular a posição com base nos valores da cadeia de hashes recebidos anteriormente por cada um: $pos = Hash(link_1^k \,||\, ... \,||\, link_{a-1}^k \,||\, link_{a+1}^k \,||\, ... \,||\, link_p^k) \bmod (g_a)$ em que $g_a$ representa a quantidade restante de cartas no Deck de jogo de $P_a$;
4. Todos os jogadores atualizam a informação acerca do jogador de $P_a$ de forma a remover o $h_a^{pos}$ (correspondente à carta retirada do Deck) de $Pile_a$, adicionando este hash a $Hand_a$, referente a mão do jogador $P_a$.

Cabe ressaltar que este mecanismo não impõe restrição na ordem em que os jogadores retiram cartas dos seus respectivos de Deck de Jogo. Por exemplo, existem jogos em que os jogadores retiram cartas dos Decks de jogo um por vez e outros em que cada jogador retira todas as cartas necessárias sequencialmente antes do próximo jogador retirar as suas.

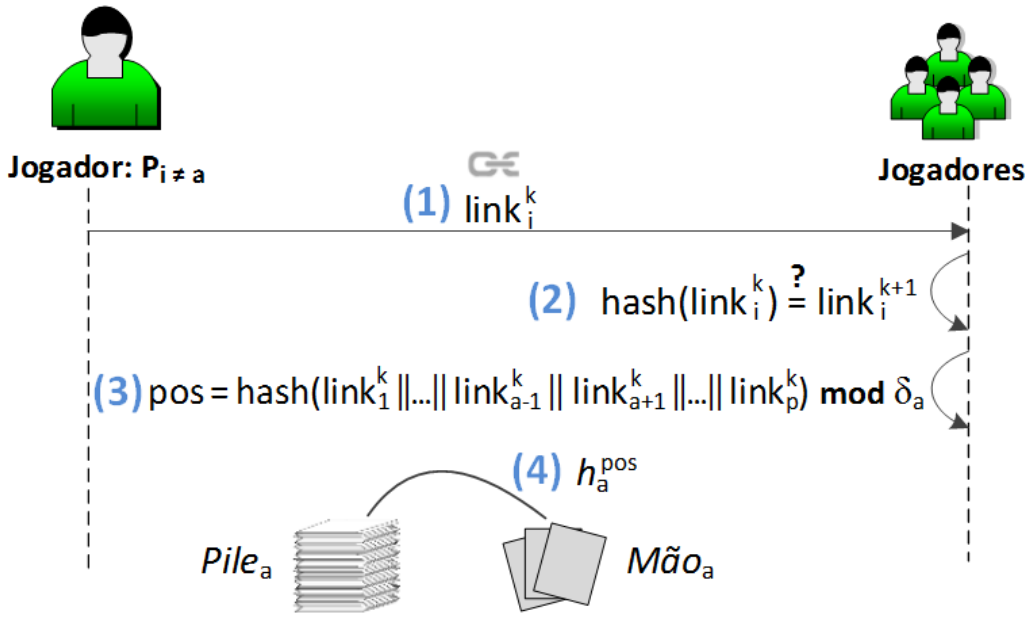


Figura 12 – Compra de carta do protocolo SecureTCG

### 4.1.4 Revelando uma carta na mesa

No cenário em que o jogador $P_a$ queira revelar uma carta e colocá-la no jogo, este só poderá escolher dentre as cartas que estão na sua mão, ou seja, que estão descritas pela estrutura de dados $Hand_a$ armazenada localmente em todos os demais jogadores. Para a validação de que nenhuma trapaça ocorreu, o mecanismo descrito abaixo deve ser seguido.

**Descrição do mecanismo**

1. Supondo que o jogador $P_a$ queira revelar a carta com a identificação $ID_a^j$, o jogador $P_a$ revela o valor de $c_a^j = (mask_a^j, ID_a^j)$ definindo os parâmetros de jogo *params* associados a esta carta para todos os demais jogadores. Se o Deck Base do jogador $P_a$ foi gerado de acordo com o protocolo de geração de Deck Base descrito anteriormente, $P_a$ também revela $v_a^b$ para a carta e o correspondente índice b;
2. Todos os demais jogadores da partida calculam $h_a^j = Hash(mask_a^j, ID_a^j)$ e verificam se o resultado corresponde a uma das cartas armazenadas localmente

em $Hand_a$. Para o caso de jogos com a geração randômica de Deck Base, os jogadores também verificam se $Hash(v_a^b) = VCommit_a[b]$ e se $ref_a^b = v_a^b \oplus r_a^b$ corresponde a $ID_a^j$, o que significaria que $v_a^b$ foi utilizado no protocolo de geração de Deck Base e que a carta revelada foi de fato uma seleção randômica do Deck Universal. Se as verificações anteriores falharem, um alerta de trapaça é disparado e as medidas adequadas para a situação devem ser tomadas;

3. Finalmente, todos os jogadores removem o valor $h_a^j$ de $Hand_a$ e o armazenam na estrutura de $Used_a$ de forma a indicar que esta carta já foi utilizada pelo jogador $P_a$.

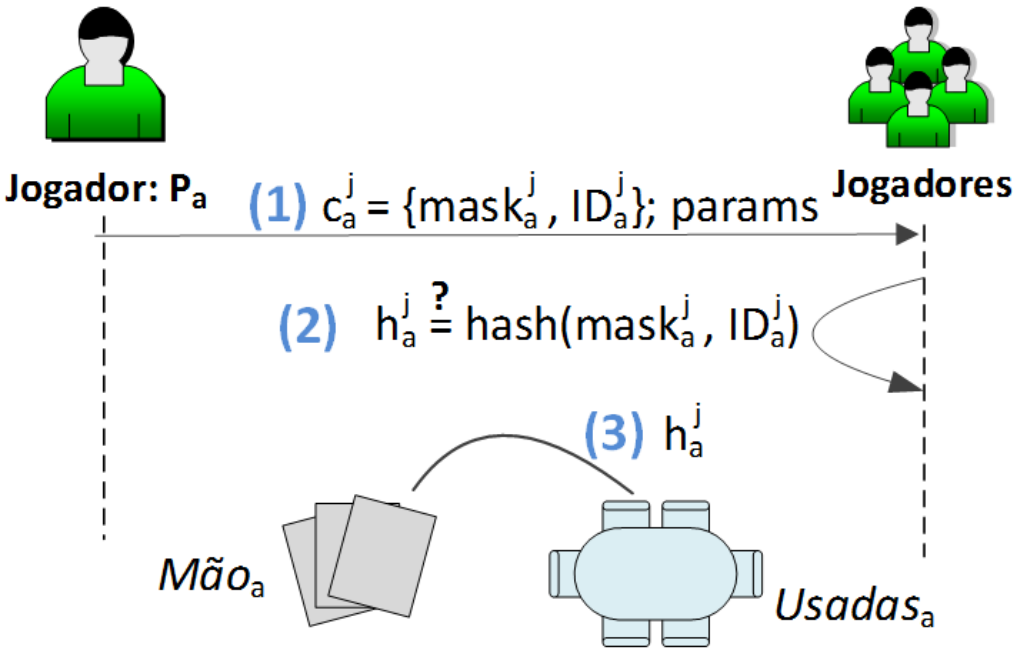


Figura 13 – Revelação de carta do protocolo SecureTCG

## 4.2 Detecção de trapaças em ambiente P2P

A solução proposta não requer a intervenção de uma terceira entidade atuando entre os jogadores durante toda a partida para garantir os requisitos de segurança da aplicação. Mais especificamente, as atividades que envolvem o servidor do jogo são transações restritas que ocorrem antes da partida começar, como a compra de cartas e a troca das mesmas, e após a partida terminar, no caso da auditoria do jogo se fazer necessária.

### 4.2.1 Consistência de Decks

Ao revelar os valores de hash de todas as cartas no protocolo de inicialização, os jogadores se comprometem às cartas dos Decks de jogo bem como a ordenação do mesmo. Dessa forma, a não ser que um jogador desonesto $P_d$ seja capaz de gerar $c_d^{j'}$ que satisfaz $Hash(c_d^{j'}) = Hash(c_d^{j})$ para um dado j, violando as propriedades de segurança da função de hash, o jogador $P_d$ não pode modificar a informação contida em $Deck_d$ e $Pile_d$ sem que apareçam inconsistências nestas estruturas de dados armazenas nos demais jogadores durante o protocolo de revelação de cartas. Isto é, quanto o jogador $P_d$ tenta colocar $c_d^{j'}$ no jogo, os demais jogadores são capaz de detectar que $Hash(c_d^{j'})$ não existe em suas versões de $Hand_d$.
Além disso, no final do jogo, após cada jogador $P_i$ revelar todos seus valores de $mask_i$ randômicos, os demais jogadores poderiam verificar se o Deck foi construído da maneira correta.

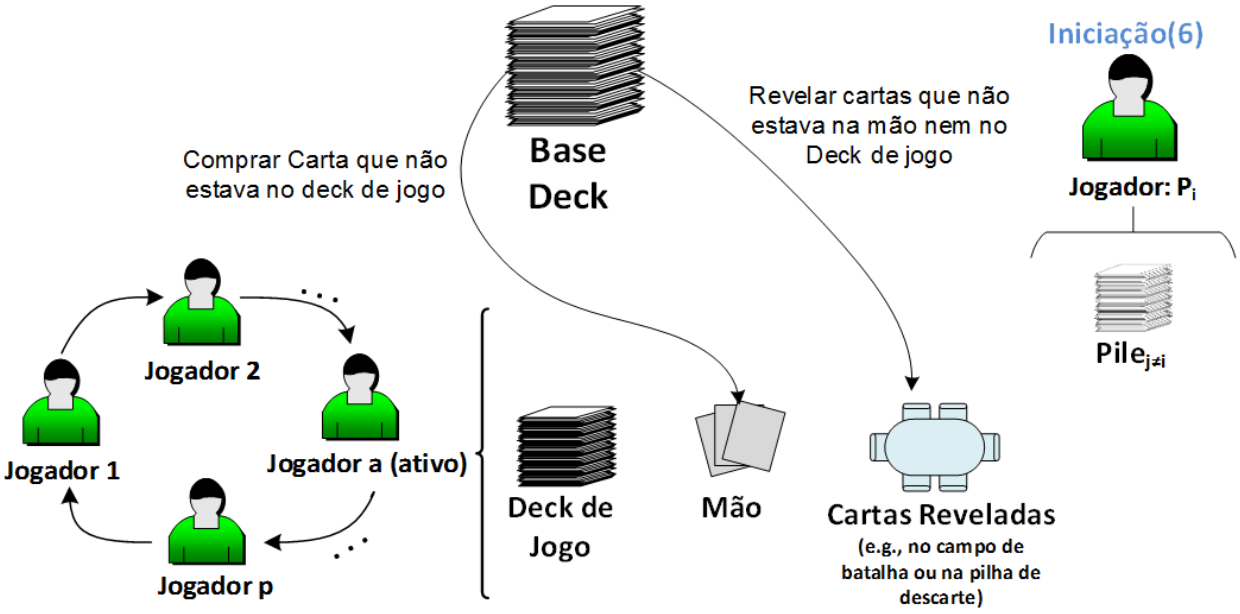


Figura 14 – Esquema de utilização de cartas não autorizadas do Deck Base

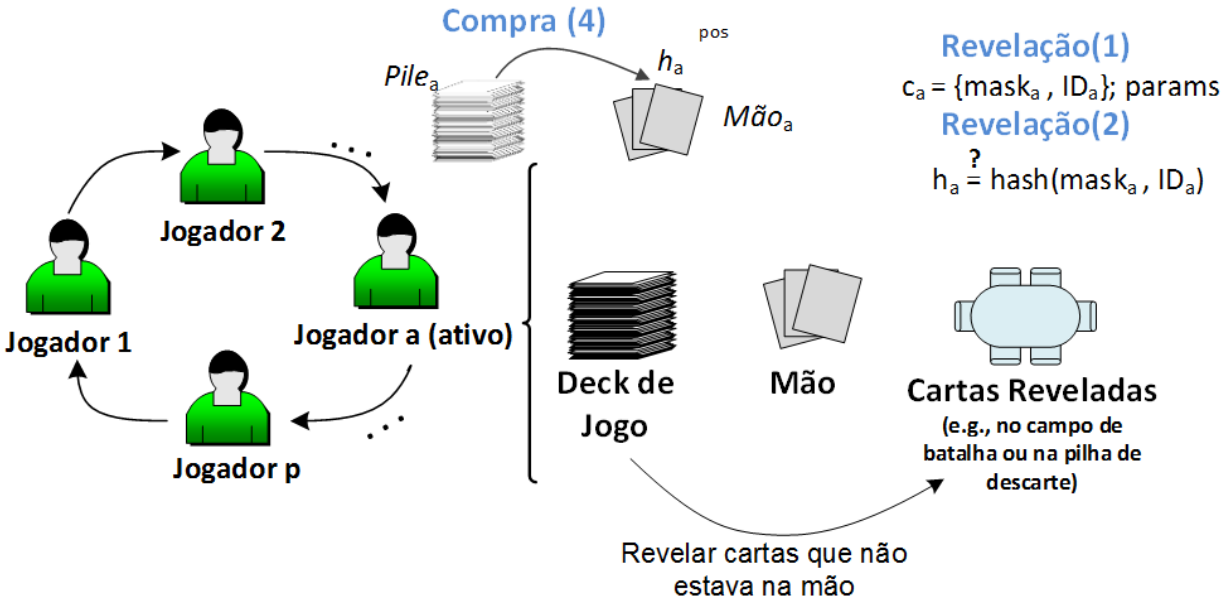


Figura 15 – Esquema de utilização de cartas não autorizadas do Deck de Jogo

### 4.2.2 Distribuição Aleatória das Cartas

A ordem em que as cartas são retiradas pelo jogador $P_a$ depende da variável *pos* calculada no protocolo de retirada de cartas do Deck de Jogo. A variável *pos* segue uma distribuição uniforme, visto que é calculada a partir da aplicação de funções hash na cadeia de links proveniente de todos os jogadores com exceção de $P_a$. Sendo assim, desde que a entrada da função de hash não seja manipulada, cada carta tem a mesma probabilidade de ser retirada. Na solução proposta, a manipulação desta entrada é impraticável devido ao uso de cadeias de hash. Isto é todos os jogadores revelam a variável *tail* durante a fase de inicialização de forma que se comprometem a contribuir com uma sequência de variáveis pseudoaleatórias para o cálculo de *pos*.

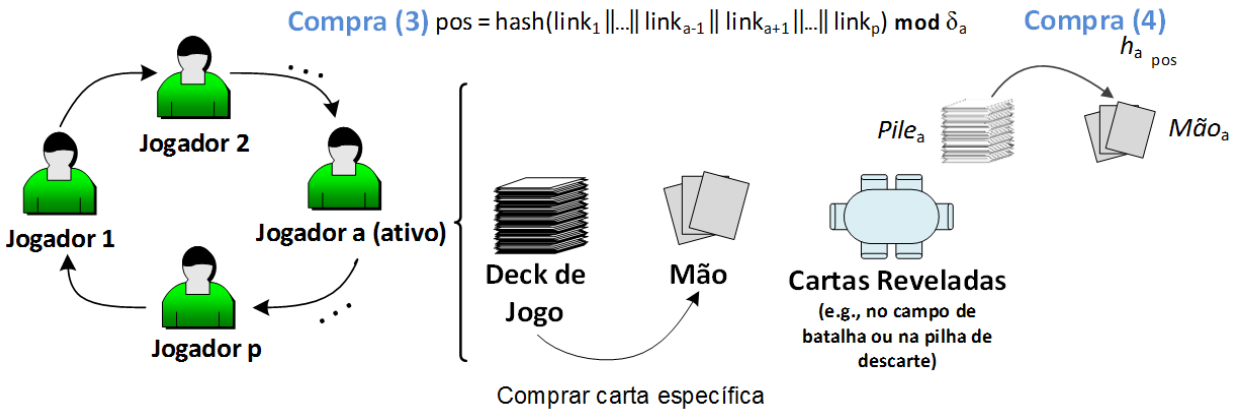


Figura 16 – Esquema de fraude sobre a aleatoriedade de compra de cartas

### 4.2.3 Confidencialidade Completa das Cartas

A solução proposta sugere que concatenar uma variável randômica $mask_i^j$ a uma carta identificada com ID é necessário, visto que no caso em que isso não ocorra algumas situações indesejadas podem ocorrer: duas cartas com o mesmo ID iriam mapear o mesmo $h_i^j$, revelando que a existência de cartas repetidas no mesmo Deck; Um jogador $P_d$ que sabe quais cartas existentes no Deck de Jogo de um outro jogador $P_i$ (numa situação hipotética em que algumas cartas sejam memorizadas após a finalização de um jogo anterior com o mesmo Deck) poderia calcular a função hash de uma carta $ID_i^j$ do jogador $P_d$ e tentar localizar este valor na estrutura de dados $Pile_i$, identificando sua localização. Cabe ressaltar também que a não repetição de $mask_i^j$ para cada carta do jogador $P_i$ em diferentes jogos é necessária, dado que na situação em que isto não aconteça, os demais jogadores poderiam reconhecer os valores de $hash_i^j$ localizados em $Pile_i$.

### 4.2.4 Efeitos Mínimos de Coalisão

Jogadores atuando em coalisão não podem ganhar nenhuma informação útil acerca das cartas de um outro jogador não participante da coalisão, visto que todos os jogadores têm acesso apenas a suas próprias cartas e contribuem com sua parcela de aleatoriedade para o cálculo do índice de retirada de uma carta por outro jogador.

Por outro lado, jogadores em coalisão não podem subverter a natureza aleatória do processo de retirar uma carta do Deck de outro jogador desde que haja pelo menos um jogador honesto no jogo. Por exemplo, supondo que o jogador $P_a$ queira retirar uma carta do Deck de jogo, um outro jogador desonesto $P_d$ poderia esperar até que todos os demais jogadores contribuíssem com o próximo valor de suas respectivas cadeias de hashes para que o índice da carta a ser retirada fosse determinado e, somente então, escolhesse um valor para enviar que combinado com o valor enviado pelos demais jogadores resultaria em um índice desejado. Esta manobra não beneficiaria $P_d$ diretamente, já que as contribuições de cada jogador não são utilizadas para calcular o índice de retirada da carta do próprio jogador. Entretanto, dois jogadores $P_i$ e $P_d$ poderiam formar uma coalisão de forma a tentar colocar uma carta interessante nas mãos um do outro.

Sendo assim, mesmo que existam alguns jogadores atuando em coalisão, estes não poderão subverter a segurança e calcular a priori a posição do índice da carta a ser retirada por um outro jogador. Isto se deve ao fato de que a existência de um único jogador honesto garante uma parcela de aleatoriedade no cálculo do índice *pos* que tornaria inviável o seu descobrimento a priori pelos jogadores atuando em coalisão.

Finalmente, no caso extremo em que todos os jogadores com exceção de $P_a$ estejam atuando em coalisão, por mais que fosse possível calcular a priori o índice de retirada da carta por $P_a$, isto não revelaria nenhuma informação útil acerca da carta a ser retirada por $P_a$ porque nenhum outro jogador a não ser $P_a$ conhece o valor verdadeiro do ID das cartas descritas na estrutura $Pile_a$.

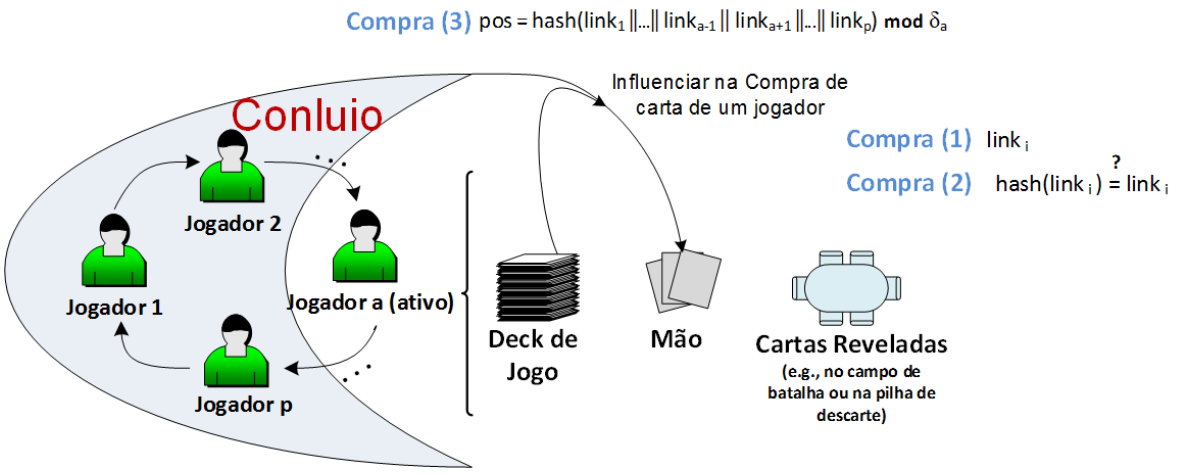


Figura 17 – Esquema de conluio para prejudicar jogador honesto

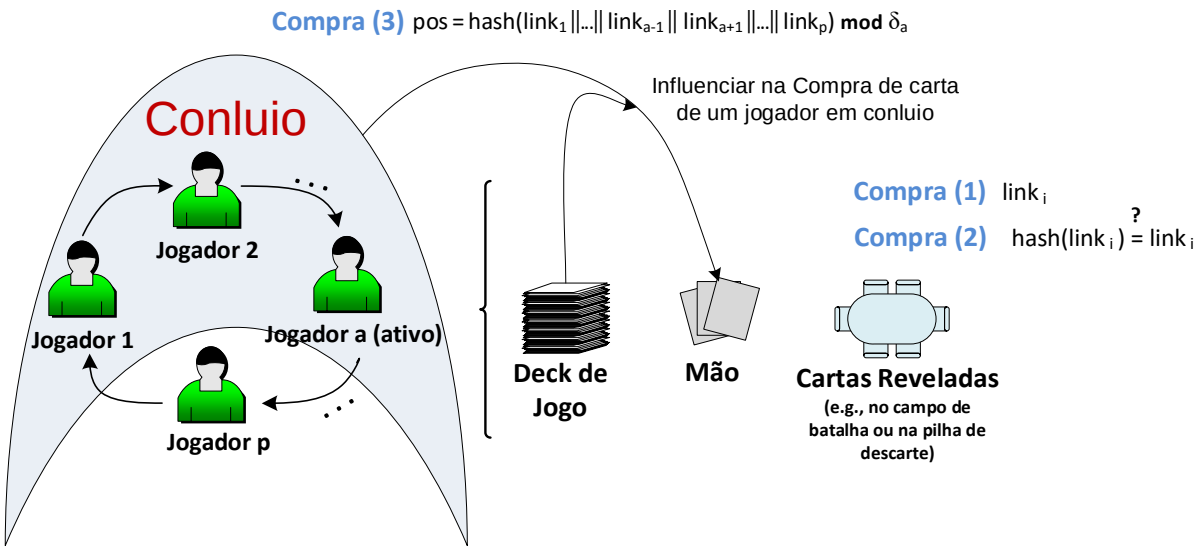


Figura 18 – Esquema de conluio para beneficiar jogador desonesto

### 4.2.5 Detecção de trapaças com alta probabilidade

A solução adotada pelo SecureTCG detecta com alto grau de confiança tentativas de trapaças como a tentativa de manipulação da ordem em que as cartas são retiradas por algum jogador, jogar uma carta que não esteja na mão do jogador em questão e a tentativa de construção de um Deck que vá contra as regras do jogo. Entretanto, está afirmação se baseia na hipótese de que a solução utilize uma implementação da função de hash que garanta suas propriedades de segurança e que sejam utilizados comprimentos suficientemente grandes de hashes, visto que este é o principal ferramental criptográfico em que se baseia o SecureTCG.

# 5 ESPECIFICAÇÕES

## 5.1 Requisitos

### 5.1.1 Requisitos do Sistema

A implementação proposta para a Prova de Conceito consiste no desenvolvimento de um software que permita a um ou mais usuários, denominados jogadores, se encontrarem através da rede, estabelecerem um canal de comunicação entre si e disputar uma partida de um jogo exemplo do estilo TCG, cujo andamento deve ser protegido de trapaças pelos mecanismos propostos pelo protocolo SecureTCG. Devem existir também funcionalidades que ofereçam a possibilidade de realizar tentativas de trapaça, de modo a permitir a verificação da eficácia dos mecanismos do protocolo implementados.
Desta forma, foram levantados os seguintes requisitos para a plataforma principal do sistema:

**Requisitos funcionais**

- Servir como ponto de encontro: o sistema básico deve, logo após a inicialização, oferecer ao jogador a possibilidade de encontrar outros jogadores interessados em disputar uma partida de TCG;
- Estabelecer conexão: o sistema deve cuidar do estabelecimento de um canal de comunicação entre os jogadores que entrarem em acordo entre si sobre iniciar uma partida TCG;
- Registrar e apresentar logs: o sistema deve oferecer um mecanismo de log que registre e exiba de maneira legível todas as etapas do seu funcionamento, principalmente as mensagens trocadas entre os jogadores na inicialização e no decorrer de uma partida, tanto relacionadas ao estado da partida em si quanto à execução dos mecanismos do protocolo de detecção e prevenção de trapaças. Esse registro será valioso na avaliação da eficácia do protocolo e da implementação desenvolvida.

**Requisitos não funcionais**

- Usabilidade: apresentar uma experiência de usuário simples e intuitiva, permitindo que em pouco tempo o usuário consigam realizar as tarefas desejadas eficientemente;
- Desempenho: apresentar tempos de resposta reduzidos às interações com o usuário;
- Segurança: assegurar que a conexão estabelecida seja segura e que as informações restritas do usuário sejam protegidas;
- Portabilidade: oferecer suporte a diferentes plataformas e canais de comunicação diretamente ou através de poucas adaptações;
- Escalabilidade: suportar conexões entre múltiplos jogadores, mantendo a estabilidade inclusive em situações críticas, como grande número de jogadores numa mesma partida ou abandono de um jogador durante uma partida em andamento.

### 5.1.2 Requisitos do Jogo TCG exemplo

Como descrito na seção "Definições do TCG exemplo", viu-se a necessidade de implementar para a Prova de Conceito um jogo TCG exemplo, que apresentasse um conjunto de regras e uma dinâmica que o tornassem minimamente jogável, e ao mesmo tempo fossem suficientes para apresentar situações suscetíveis a trapaças, para permitir exercitar os mecanismos de detecção propostos pelo protocolo SecureTCG. Os requisitos a serem atendidos pelo TCG exemplo são:

**Requisitos funcionais**

- Permitir a realização de partidas: a implementação do jogo deve permitir aos jogadores realizar partidas conforme a dinâmica e as regras estabelecidas na seção de "Definições do TCG exemplo" e nos casos de uso, desde as etapas de inicialização até o controle dos turnos e da eliminação de jogadores derrotados;

- Permitir a execução de trapaças: o TCG exemplo deve disponibilizar uma ferramenta que permita a execução deliberada de tentativas de trapaça, para poder exercitar e avaliar a eficácia dos mecanismos de prevenção implementados.

**Requisitos não funcionais**

- Simplicidade de dinâmica e regras: o jogo exemplo deve ser simples de entender e jogar, dado que ele seu intuito principal é testar a implementação do protocolo de segurança;
- Interesse: as regras definidas devem ser suficientes para manter minimamente o interesse dos jogadores ao longo de uma partida;
- Custo computacional reduzido: o jogo deve apresentar reduzida utilização de operações de alto custo computacional, de dados armazenados e de trocas de de mensagens entre os jogadores durante a partida, e as mensagens que inevitavelmente precisem ser enviadas devem ter um tamanho reduzido;
- Escalabilidade: assim como os outros componentes do sistema, a lógica do jogo deve oferecer bom suporte a múltiplos jogadores.

### 5.1.3 Requisitos do módulo de prevenção e detecção de trapaças

Como descrito na seção de "Trapaças", a violação de algumas regras do jogo que envolvem cartas já reveladas pode ser facilmente detectada por outros jogadores, afinal qualquer jogador pode verificar se o seu efeito sobre o jogo está de acordo com o conjunto de regras estabelecido. O mesmo não se aplica, no entanto, à detecção de trapaças relacionadas às cartas na mão ou no Deck de Jogo do jogador, bem como à construção aleatória do Deck Base, nos estilos de jogo em que ele é utilizado. Levando-se em conta esses fatores e as peculiaridades apresentadas pela utilização da arquitetura P2P, é possível identificar um conjunto de requisitos para o módulo de detecção e prevenção dessas Trapaças Sigilosas, que deve ser aplicável a uma grande variedade de TCGs:

**Requisitos funcionais**

- Garantir a consistência do Deck de Jogo: ainda que os jogadores em um TCG geralmente sejam autorizados a construir seus próprios Decks de Jogo, a seleção de cartas que o compõem não pode ser alterada após o início de uma partida. Além disso, deve haver um mecanismo para que os jogadores possam verificar se os decks dos adversários foram construídos de acordo com as regras do jogo (por exemplo, verificar a aleatoriedade de um Deck Base, quando aplicável);
- Garantir a compra aleatória e uniforme de cartas: a composição da mão de cada jogador, ou seja, a ordem das cartas que ele retira do seu Deck de Jogo, deve depender de contribuições de todos os jogadores da partida, de tal forma que nenhum deles seja capaz de prever ou controlar a ordem em que as cartas são compradas por um determinado jogador. Esse requisito deve ser assegurado sempre que um baralho precisar ser embaralhado, o que geralmente ocorre antes do início da partida;
- Manter a confidencialidade completa das cartas: normalmente, os jogadores só são autorizados a saber das cartas que possuem em suas próprias mãos, ou das que estão em jogo ou na pilha de descarte, e não das cartas em nenhum dos Decks de Jogo (que estariam "viradas para baixo", no mundo real) ou na mão dos adversários. Ou seja, a confidencialidade das cartas em relação a cada jogador deve ser mantida até o momento que ele esteja autorizado a conhecê-las;
- Garantir a utilização de cartas autorizadas: os jogadores numa partida devem poder revelar somente cartas presentes em sua mão, que por sua vez tenham sido compradas do seu Deck de Jogo, ou seja, cartas que eles estejam autorizados a utilizar. Deve haver um mecanismo que permita aos adversários perceber se houver violação dessa regra;
- Minimizar os efeitos de conluio: em partidas envolvendo mais de dois jogadores, alguns deles podem decidir estabelecer um canal de comunicação paralelo e, em seguida, trocar informações sobre o jogo (por exemplo, as cartas em suas mãos) ou sobre o protocolo do jogo (por exemplo, uma chave secreta). Se isso acontecer, a quantidade de informação obtida por esses jogadores deve ser

equivalente ao que eles já sabiam separadamente e insuficiente para violar os requisitos anteriores. Por exemplo, eles podem compartilhar informações sobre suas próprias mãos, mas eles não podem conseguir obter nenhuma nova informação sobre as cartas no Deck de Jogo ou na mão de um jogador honesto.

**Requisitos não funcionais**

- Independência de um TTP: a detecção de trapaças não deve depender da intervenção de uma entidade de confiança, seja humana ou máquina, durante a partida. Em outras palavras, não deve haver interferência de um TTP durante uma partida;
- Alta probabilidade de detecção de trapaças: qualquer tentativa de fraude deve ser detectada pela solução, seja no final do jogo ou, de preferência, no momento em que acontece;
- Custo computacional reduzido: o módulo de segurança deve se utilizar pouco de operações de alto custo computacional, tais como as comumente utilizadas por algoritmos de criptografia assimétrica. Além disso, não deve exigir que os jogadores armazenem grandes estruturas de dados durante a partida. Finalmente, não deve envolver trocas de um grande número de mensagens entre os jogadores, e as mensagens que inevitavelmente precisem ser enviadas devem ter um tamanho reduzido;
- Escalabilidade e suporte a múltiplos jogadores: os mecanismos de segurança devem ser flexíveis o suficiente para suportar, sem impactos significativos nos custos computacionais e na performance, dois ou mais jogadores simultâneos na mesma partida;
- Tolerância a abandono de jogador: se um jogador deixa a partida, intencional ou acidentalmente, os jogadores remanescentes devem ser capazes de continuar a jogar. Este requisito é essencial para assegurar a continuidade da partida na ocasião de eliminação de um jogador derrotado, e ainda, no contexto de jogos online, nos casos de problemas de conexão;
- Transparência: os mecanismos de segurança devem operar de forma transparente aos jogadores, exceto quando eles deliberadamente tiverem

interesse em verificá-los (através dos logs) e nos casos em que forem detectadas trapaças.

Deve se ressaltar que grande parte desses requisitos devem ser atingidos utilizando mecanismos baseados no cálculo de funções de hash que, simplificadamente, têm o objetivo de gerar de resumos criptográficos (com tamanho fixo) de mensagens, e de maneira unidirecional (isto é, não invertível). Elas são amplamente empregadas pelo protocolo SecureTCG por permitirem: prevenir que se obtenha o conteúdo da mensagem a partir do seu hash (devido à unidirecionalidade); criar mecanismos de comprometimento dos jogadores com suas mensagens pois, divulgado um hash de uma mensagem, o jogador estará comprometido com ela, dado que é muito improvável que consiga obter outra mensagem que possua o mesmo hash; gerar cadeias de hash, em que o comprometimento apenas com o último elo resulta automaticamente no comprometimento com todos os outros. Além disso, elas ainda podem ser utilizadas na construção de geradores pseudoaleatórios.
Portanto, para o atendimento dos requisitos da solução, é essencial a utilização de uma função de hash ofereça (SIMPLICIO, 2015):

- Resistência a primeira inversão: dado um valor de Hash R, é computacionalmente inviável encontrar uma mensagem M tal que R = H(M);
- Resistência a segunda inversão: dado um valor de Hash R e uma mensagem $M_1$ tal que R = H($M_1$), é computacionalmente inviável encontrar uma outra mensagem $M_2 \neq M_1$ tal que R = H($M_2$);
- Resistência a colisões: é computacionalmente inviável encontrar duas mensagens $M_1$ e $M_2$ tais que H($M_1$) = H($M_2$);
- Boa performance: a ampla utilização das funções de hash pelo protocolo implementado significa que a sua performance possui grande impacto no desempenho e na experiência de usuário da solução final.

## 5.2 Casos de uso

**Caso de uso UC01**

Nome: Construir Decks Base.
Descrição: O Sistema gera, a partir do Deck Universal, diferentes Decks Base para cada um dos Jogadores, cujas cartas são selecionadas aleatoriamente e são desconhecidas pelos Jogadores adversários.
Objetivo: Gerar Decks Base aleatórios e confidenciais para os Jogadores adversários.
Fase: Construção dos Decks.
Ator(es): Sistema, Jogadores.
Pré-condições: Existe um Deck Universal.
Fluxo normal:

1. Os Jogadores informam ao Sistema que pretendem disputar uma partida, para a qual eles precisam de Decks de Jogo, que devem ser gerados a partir de Decks Base.
2. O Sistema, a partir da escolha aleatória das cartas disponíveis no Deck Universal do jogo, constrói diferentes Decks Base e os atribui a cada um dos Jogadores, sem permitir que eles conheçam a composição dos Decks uns dos outros.

Fluxo de exceção A:

1. Os Jogadores informam ao Sistema que pretendem disputar uma partida, para a qual eles precisam de Decks de Jogo, que devem ser gerados a partir de Decks Base.
2. Um ou mais Jogadores adulteram o mecanismo de escolha aleatória de cartas utilizado pelo Sistema.
3. O Sistema constrói diferentes Decks Base a partir do Deck Universal e os atribui a cada um dos Jogadores, porém a aleatoriedade e a confidencialidade das cartas não pode ser assegurada.

Fluxo de exceção B:

1. Os Jogadores informam ao Sistema que pretendem disputar uma partida, para a qual eles precisam de Decks de Jogo, que devem ser gerados a partir de Decks Base.
2. O Sistema, a partir da escolha aleatória das cartas disponíveis no Deck Universal do jogo, constrói diferentes Decks Base e os atribui a cada um dos Jogadores.

3. Um ou mais Jogadores burlam o mecanismo de confidencialidade de cartas utilizado pelo Sistema, desvendando a composição de Decks Base que não deveria(m) conhecer.

Pós-condições: Existem diferentes Decks Base atribuídos a cada um dos Jogadores, construídos de maneira aleatória e cuja composição de cartas é desconhecida pelos seus adversários.

**Caso de uso UC02**

Nome: Construir Deck de Jogo.
Descrição: O Jogador constrói seu Deck de Jogo selecionando as cartas desejadas dentre as disponíveis no seu Deck Base, mantendo-as confidenciais para seus adversários.
Objetivo: Construir Deck de Jogo de composição confidencial para os Jogadores adversários.
Fase: Construção dos Decks.
Ator(es): Jogador.
Pré-condições: O Jogador possui um Deck Base.
Fluxo normal:

1. O Jogador escolhe um subconjunto de cartas de seu Deck Base para compor seu Deck de Jogo, de forma confidencial para seus adversários.

Fluxo de exceção:

1. O Jogador escolhe um subconjunto de cartas de seu Deck Base para compor seu Deck de Jogo.
2. Um ou mais Jogadores adversários burlam o mecanismo de confidencialidade de cartas utilizado pelo Sistema, obtendo conhecimento indevido sobre a composição de Deck de Jogo.

Pós-condições: O Jogador possui um Deck de Jogo, formado por cartas selecionadas do seu Deck Base e confidenciais para seus adversários.

**Caso de uso UC03**

Nome: Comprar carta.
Descrição: O Jogador, uma vez a cada turno seu, retira uma carta aleatória de seu Deck de Jogo e a coloca em sua mão, sem revelá-la aos Jogadores adversários.
Objetivo: Retirar uma carta do Deck de Jogo e colocá-la em sua mão, sem revelá-la aos Jogadores adversários.
Fase: Turnos da partida.
Ator(es): Jogador.
Pré-condições: O Jogador possui cartas no seu Deck de Jogo e está em um turno seu da partida, em que ainda não comprou nenhuma carta.
Fluxo normal:

1. O Jogador retira aleatoriamente uma carta de seu Deck de Jogo, sem revelá-la aos Jogadores adversários.
2. O Jogador coloca a carta em sua mão, também sem revelá-la aos Jogadores adversários.

Fluxo de exceção A:

1. O Jogador adultera o mecanismo de escolha aleatória de cartas do Sistema.
2. O Jogador retira uma carta de sua escolha de seu Deck de Jogo, sem revelá-la aos Jogadores adversários.
3. O Jogador coloca a carta em sua mão, também sem revelá-la aos Jogadores adversários.

Fluxo de exceção B:

1. Um ou mais Jogadores adversários adulteram o mecanismo de escolha aleatória de cartas do Sistema.
2. O Jogador retira uma carta de seu Deck de Jogo cuja escolha foi influenciada por um ou mais Jogadores adversários.
3. O Jogador coloca a carta em sua mão.

Fluxo de exceção C:

1. O Jogador retira aleatoriamente uma carta de seu Deck de Jogo, sem revelá-la aos Jogadores adversários.
2. O Jogador coloca a carta em sua mão, também sem revelá-la aos Jogadores adversários.

3. Um ou mais Jogadores adversários burlam o mecanismo de confidencialidade de cartas utilizado pelo Sistema, obtendo conhecimento indevido sobre a carta comprada.

Pós-condições: O Jogador possui uma carta a mais em sua mão, retirada aleatoriamente do seu Deck de Jogo e sem que seus adversários a conhecessem, e não pode mais comprar cartas no turno.

**Caso de uso UC04**

Nome: Revelar carta.
Descrição: O Jogador, uma vez a cada turno seu, retira uma carta de sua escolha de sua mão e a coloca em jogo, revelando-a aos Jogadores adversários.
Objetivo: Retirar uma carta de sua mão e colocá-la em jogo, revelando-a aos Jogadores adversários.
Fase: Turnos da partida.
Ator(es): Jogador.
Pré-condições: O Jogador possui cartas em sua mão e está em um turno seu da partida, em que ainda não revelou nenhuma carta.
Fluxo normal:

1. O Jogador retira uma carta de sua escolha de mão.
2. O Jogador coloca a carta em jogo, revelando-a aos Jogadores adversários.

Fluxo de exceção:

1. O Jogador adultera o mecanismo de controle das cartas do Sistema.
2. O Jogador escolhe uma carta que não estava em sua mão.
3. O Jogador coloca a carta em jogo, revelando-a aos Jogadores adversários.

Pós-condições: O Jogador possui uma carta escolhida da sua mão a mais em jogo, revelada aos seus adversários a conhecessem, e não pode mais revelar cartas no turno.

**Caso de uso UC05**

Nome: Atacar um adversário.
Descrição: O Jogador, uma vez a cada turno seu, escolhe uma de suas cartas em jogo e a utiliza para atacar um dos Jogadores adversários.
Objetivo: Atacar um Jogador adversário utilizando uma de suas cartas em jogo.
Fase: Turnos da partida.
Ator(es): Jogador.
Pré-condições: O Jogador possui cartas em jogo e está em um turno seu da partida, em que ainda não atacou.
Fluxo normal:

1. O Jogador escolhe uma de suas cartas em jogo.
2. O Jogador escolhe um dos Jogadores adversários para atacar, que deve arcar com o dano determinado pelos pontos de ataque da carta utilizada.

Pós-condições: O Jogador realizou um ataque a um Jogador adversário com uma carta escolhida de seu jogo, e não pode mais atacar no turno.

**Caso de uso UC06**

Nome: Defender um ataque.
Descrição: O Jogador, ao sofrer um ataque de um Jogador adversário, escolhe uma de suas cartas em jogo, se tiver, e a utiliza para absorver o ataque.
Objetivo: Defender um ataque de um Jogador adversário utilizando uma de suas cartas em jogo ou os próprios pontos de vida.
Fase: Turnos da partida.
Ator(es): Sistema, Jogador.
Pré-condições: O Jogador sofre um ataque de um Jogador adversário.
Fluxo normal:

1. O Sistema notifica o Jogador que ele está sofrendo um ataque por um Jogador adversário.
2. O Jogador obrigatoriamente escolhe uma de suas cartas em jogo para se defender, cujo valor de pontos de vida é maior que o valor dos pontos de ataque da carta atacante.

3. O Sistema decrementa os pontos de vida da carta defensora do valor dos pontos de ataque da carta atacante.

Fluxo alternativo A1:

1. O Sistema notifica o Jogador que ele está sofrendo um ataque por um Jogador adversário.
2. O Jogador obrigatoriamente escolhe uma de suas cartas em jogo para se defender, cujo valor de pontos de vida é menor ou igual ao valor dos pontos de ataque da carta atacante.
3. A Sistema zera os pontos de vida da carta defensora.
4. O Jogador retira a carta defensora de jogo e a coloca na pilha de descarte.
5. Se o valor de pontos de vida da carta defensora era menor que o valor dos pontos de ataque da carta atacante, o Sistema decrementa o dano excedente dos pontos de vida do Jogador.

Fluxo alternativo A2:

1. O Sistema notifica o Jogador que ele está sofrendo um ataque por um Jogador adversário.
2. O Jogador não possui cartas em jogo para se defender, então o Sistema decrementa seus pontos de vida do valor dos pontos de ataque da carta atacante.

Fluxo de exceção E1:

1. O Sistema notifica o Jogador que ele está sofrendo um ataque por um Jogador adversário.
2. O Jogador obrigatoriamente escolhe uma de suas cartas em jogo para se defender, cujo valor de pontos de vida é menor que o valor dos pontos de ataque da carta atacante.
3. A Sistema zera os pontos de vida da carta defensora.
4. O Jogador retira a carta defensora de jogo e a coloca na pilha de descarte.
5. O Sistema decrementa os pontos de vida do Jogador do valor excedente do ataque não absorvido pela carta defensora, resultando em um valor de pontos de vida menor ou igual a zero.
6. O Sistema remove o Jogador derrotado da partida.

Fluxo de exceção E2:

1. O Sistema notifica o Jogador que ele está sofrendo um ataque por um Jogador adversário.

2. O Jogador não possui cartas em jogo para se defender, então o Sistema decrementa seus pontos de vida do valor dos pontos de ataque da carta atacante, resultando em um valor de pontos de vida menor ou igual a zero.
3. O Sistema remove o Jogador derrotado da partida.

Pós-condições: O Jogador absorveu um ataque de um Jogador adversário com os pontos de vida de uma carta escolhida de seu jogo, com seus próprios pontos de vida ou com uma combinação dos dois.

**Caso de uso UC07**

Nome: Encerrar turno.
Descrição: O turno do Jogador é encerrado voluntariamente ou pelo Sistema.
Objetivo: Encerrar o turno do Jogador.
Fase: Turnos da partida.
Ator(es): Sistema, Jogador.
Pré-condições: O Jogador está em um turno seu da partida.
Fluxo normal:

1. O Sistema encerra automaticamente o turno corrente do Jogador após ele ter realizado todas as ações permitidas.

Fluxo alternativo:

1. O Jogador encerra voluntariamente seu turno corrente no momento desejado.

Pós-condições: O Jogador encerrou o seu turno.

# 6 AMBIENTE DE DESENVOLVIMENTO

A prova de conceito para o protocolo SecureTCG será desenvolvida em linguagem orientada a objeto Java em ambiente de desenvolvimento NetBeans. A principal razão para a escolha da linguagem Java foram a alta portabilidade oferecida por essa linguagem (para migração da aplicação para ambiente Mobile, por exemplo) e a boa oferta de bibliotecas de criptografia.

O framework de interface a ser utilizado para a prototipação das telas será o Swing, dado que possui rápida curva de aprendizado e extensa documentação com exemplos.

Optou-se nesta prova de conceito por realizar a comunicação entre os dispositivos (PCs) através de conexão Wi-Fi, dado a sua maior estabilidade em relação à tecnologia Bluetooth, o que torna o ambiente mais robusto para testes.

No que diz respeito à escolha do algoritmo de cálculo da função hash, sendo este amplamente utilizado para a implementação do módulo criptográfico do protocolo, foi selecionado a rotina Blake2 devido a sua performance ser semelhante ao MD5 e sua estrutura de segurança ser robusta, atendendo bem, dessa forma, a todos os requisitos definidos.

Para este projeto, o repositório remoto de código escolhido para gerenciar as entregas de novas implementações pelos membros do grupo foi o BitBucket, utilizando a tecnologia git para a criação e manutenção de novos branches e atualizações de versões de código.


Figura 19 – Tecnologias utilizadas no desenvolvimento da Prova de Conceito

# 7 ARQUITETURA DO SOFTWARE

A arquitetura do sistema seguirá os princípios SOLID conforme as boas práticas da estrutura de um programa escrito em linguagem de paradigma de orientação a objetos. A solução possuirá um módulo SecurityTCGCore que será implementado pelos demais componentes da aplicação e será responsável pela gestão do protocolo de segurança. Também serão definidas como componentes do sistema as camadas relativas à interface visual, ao acesso a dados, à comunicação entre jogadores e à modelagem de entidades do jogo. A arquitetura de uma possível aplicação Android poderá seguir a mesma organização de projeto descrita levando em conta algumas particularidades do sistema operacional embarcado.

## 7.1 Diagrama de componentes

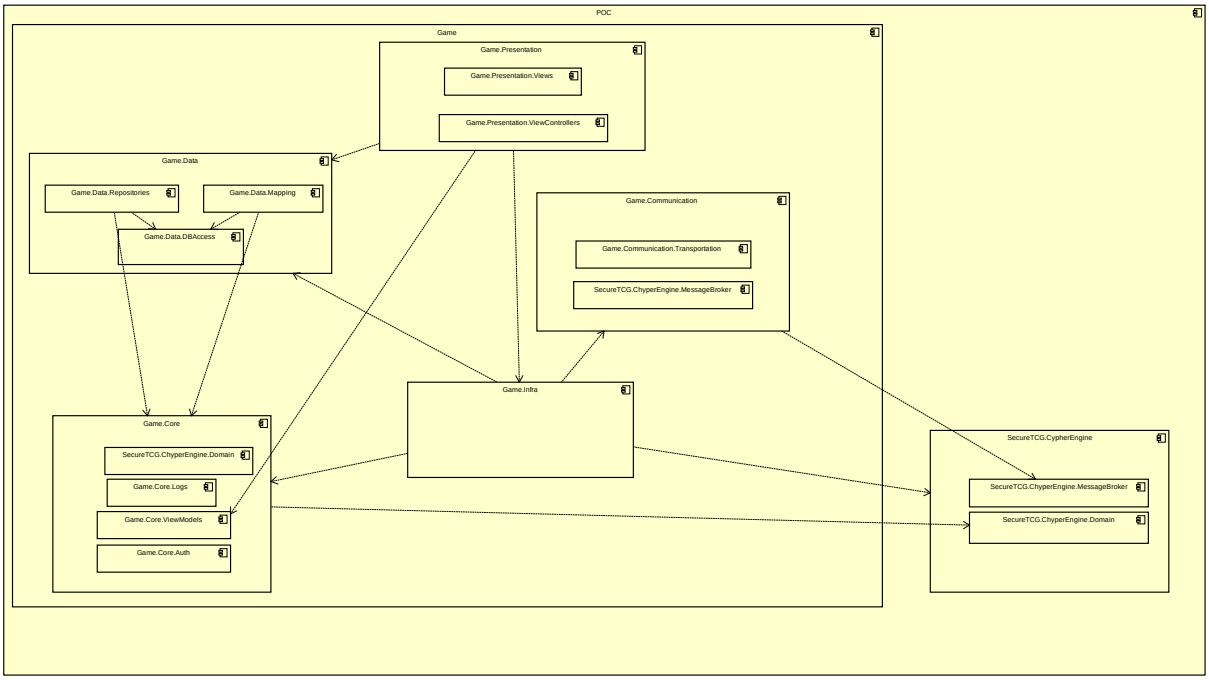


Figura 20 – Diagrama de componentes

Como o projeto visa criar um mecanismo de fácil acoplagem com sistema já existentes, o diagrama explicita a parte do jogo (pacote “Game”) e a parte de garantia de honestidade no jogo (pacote “SecureTCG.ChyperEngine”). Na primeira, é implementado um jogo simples para exemplificar o protocolo, porém poderia ser

qualquer jogo de cartas colecionáveis, podendo até mesmo representar jogos comerciais modernos que se utilizariam da arquitetura mostrada no documento atual. Esse programa referencia o pacote “SecureTCG.ChyperEngine” para implementar todos os protocolos de segurança definidos em SecureTCG. Assim, o sistema fica mais modular e de possível implantação por terceiros em sistemas já em produção.

Alguns jogos do mercado foram estudados para se chegar no diagrama apresentado para o pacote “Game”. Nele pode ser identificado um modelo que se enquadra no tradicional padrão de arquitetura Model-View-Controller (MVC). Podemos identificar as Views, que são as telas especificadas e os controladores das telas no pacote “Game.Presentation”.

Para separar mais ainda o acesso aos dados foi inserida uma camada de acesso aos dados “Game.Data”. Nela estão todos os repositórios “Game.Data.Repositories” juntamente dos mapeadores “Game.Data.Mapping” que irão, respectivamente, acessar a camada de dados para inserir, atualizar, buscar e remover e mapear os modelos definidos nas tabelas, e suas colunas, do banco de dados que é acessado pelo “Game.Data.DBAcess” (Obs.: Gamedata.dbAccess não feito, para o estudo fizemos acesso a tabelas pré-definidas, o método utilizado para alternar entre tabelas e banco de dados será demonstrado a baixo).

Toda a definição dos dados está no pacote “Game.Core”, nele são definidas as entidades relacionadas ao jogo de cartas colecionáveis, como por exemplo: Cartas, jogadores, magias, etc.

O pacote “Game.Communication” trata o estabelecimento de comunicação, e as trocas de mensagens entre as partes. Nas partes constituintes deste pacote ocorrerão diversas trocas referentes ao protocolo sendo implementado.

Para desacoplar todos os módulos e possibilitar mudanças nas implementações do sistema, foi inserido o módulo “Game.Infra” nele é implementado o injetor de dependências que o sistema utiliza. Esse conceito faz parte do Padrão de arquitetura conhecido como Injeção de Dependências. Nesse padrão, os clientes (classes) delegam a um terceiro (o injetor) a responsabilidade de prover suas dependências. Assim, é possível ter diversas implementações das interfaces e dependendo da situação o injetor irá devolver a mais apropriada. No exemplo descrito, onde o acesso ao banco de dados não foi desenvolvido, só foi necessário dizer ao injetor que em qualquer momento que for necessária uma implementação de um objeto que acesse

o banco de dados, este deverá retornar uma implementação em memória do mesmo. Isso possibilita um acoplamento fraco entre pacotes.

O pacote “SecureTCG.CypherEngine” possui classes abstratas referentes a comunicações entre as partes e possui classes abstratas referentes às entidades relativas ao jogo. Assim, o programa precisará estender as classes relativas pelas definidas neste pacote, adicionando propriedades e métodos específicos do protocolo ao programa.

## 7.2 Diagrama de Pacotes

Para melhor visualização colocamos apenas os diagramas referentes ao domínio e a camada de comunicação. Os demais podem ser encontrados no Apêndice A.

**Game.Core.Domain**

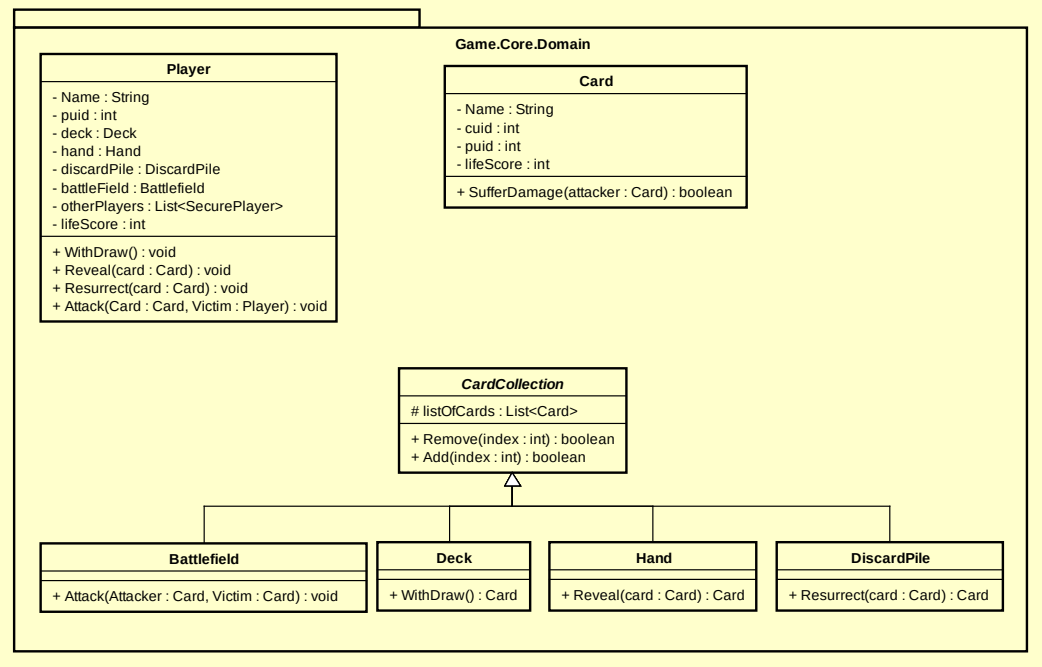


Figura 21 – Diagrama de pacotes do Game.Core.Domain

Neste pacote estão as principais classes do sistema. Nota-se que a Classe Player é que possui todas as informações do estado do jogo. Nela estão informações do próprio jogador, como cartas na mão, cartas no deck, Cartas no cemitério e informações dos

demais jogadores, pois existe a lista otherPlayers que representa todas as informações que esse jogador sabe dos demais até o momento. Além destas, informações comuns a todos no estão no Battlefield, Objeto que possui todas cartas em jogo.

**Game.Communication**

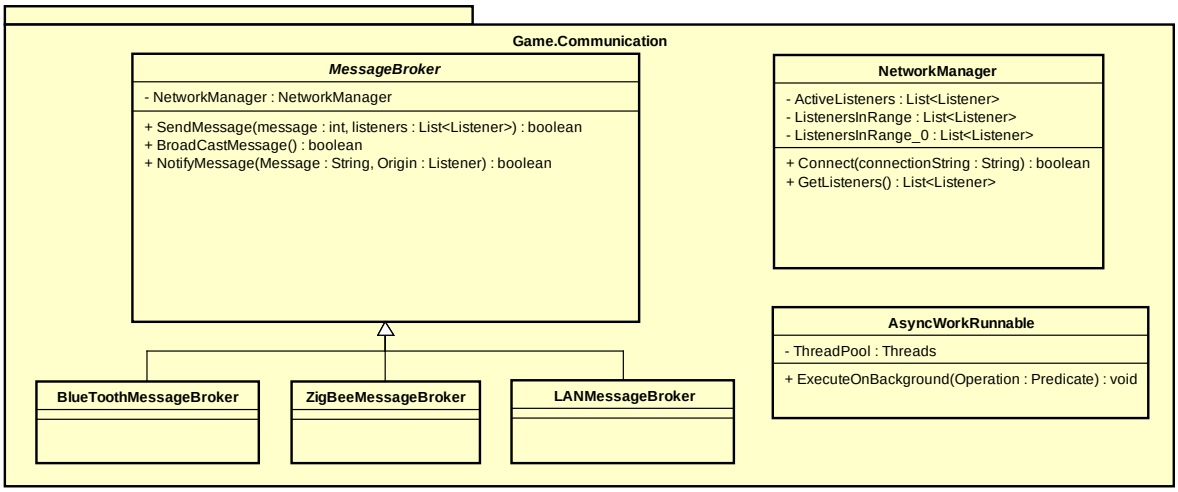


Figura 22 – Diagrama de pacotes do Game.Communication

Neste pacote se encontram as classes que administram a conexão e realizam a troca de mensagens entre os usuários. Foi desenvolvido ao longo do projeto o módulo de comunicação via BlueTooth e LAN, porém, para demonstração e facilidade de depuração seguimos com a implementação em LAN. Foi incluída ainda uma classe para controlar o paralelismo do sistema, assim, o sistema se tornou mais responsivo, por não esperar a resposta dos demais usuários para liberar o processo. Vale ressaltar que o estabelecimento da comunicação se dá na etapa inicial do jogo e deste ponto em diante qualquer mensagem que tiver de ser mandada pode ser corretamente encaminhada para o destinatário ou para todos os Listeners registrados (multicast).

## SecureTCG.CypherEngine

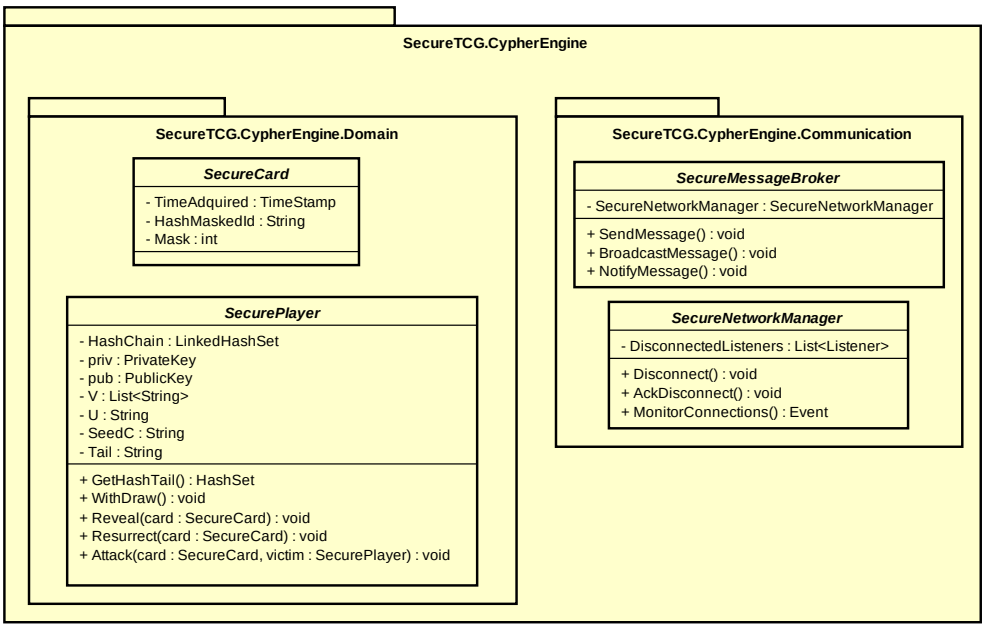


Figura 23 – Diagrama de pacotes do SecureTCG.CypherEngine

Esse pacote contém a implementação necessária para transformar o jogo representado pelos pacotes anteriores em um jogo seguro contra trapaças. Podemos notar que foram inseridas informações às classes das cartas e do jogador. Essas informações foram detalhadas no estudo do algoritmo selecionado, e são essenciais para os métodos seguros. Os métodos WithDraw, Reveal, Resurrect e Attack são os métodos que possibilitam a dinâmica do jogo. Mas as implementações destes também incluem trocas de informações criptográficas definidas no protocolo SecureTCG.

Além de prover classes abstratas relacionadas ao domínio do sistema (ao jogo especificamente), esse pacote prove abstrações para a comunicação adicionando suporte a saída de jogador no meio do jogo, avisando os demais jogadores e decidindo o que fazer, esperar o jogador voltar ou desconsiderar o turno dele.

# 8 DEFINIÇÃO DA ACEITAÇÃO

Baseado nos requisitos levantados, a seguir estão os critérios para aceitação da Prova de Conceito do protocolo SecureTCG:

- Desenvolvimento de aplicação desktop de jogo TCG com suporte a múltiplos jogadores, que trate dos processos de estabelecimento de comunicação entre jogadores interessados em disputar uma partida e apresente boa performance e usabilidade;
- Implementação das regras e da dinâmica comumente encontradas nos TCGs comerciais definidas para o jogo TCG exemplo, que permitam a disputa de partidas completas e o surgimento de cenários suscetíveis a trapaças;
- Implementação dos mecanismos protocolo SecureTCG na aplicação, de forma a garantir a prevenção ou detecção das trapaças estipuladas na inicialização e no decorrer de uma partida, de maneira puramente P2P;
- Implementação de ferramentas de execução deliberada de trapaças na inicialização e no transcorrer de uma partida, que permitam exercitar a implementação dos mecanismos de segurança propostos pelo protocolo;
- Criação de mecanismo de gravação e exibição de log, rodando em todas as etapas do jogo, cobrindo principalmente a troca de mensagens do protocolo, de forma a evidenciar a sua atuação na garantia de prevenção e detecção de trapaças em uma partida;
- Elaboração de interface gráfica amigável e intuitiva na solução final, que permita fluidez nas ações do usuário, inclusive no jogo;
- Obtenção de uma solução final que permita, através da operação conjunta das implementações acima citadas, validar a viabilidade de utilização do protocolo SecureTCG na obtenção de segurança contra trapaças em partidas TCG online sobre arquitetura P2P, atingindo a meta principal da Prova de Conceito.

# 9 CONCLUSÃO E TRABALHOS FUTUROS

O projeto de prova de conceito do protocolo SecureTCG foi concluído de forma a evidenciar o funcionamento esperado fornecendo as garantias de segurança adequadas a um Trading Card Game. Para isso, foi desenvolvido uma aplicação de jogo de cartas com regras simples de ataque e defesa com interface amigável e log de monitoramento dos processos criptográficos em tempo real.

A implementação do projeto apresentou desafios no que diz respeito à necessidade de refatorar certas partes do projeto do SecureTCG original de forma a garantir uma maior modularidade do sistema e do projeto do protocolo em si. Além disso, o desenvolvimento das rotinas base do gestor de mensageria criptográfica e de entidade que utilizam bibliotecas gráficas representaram desafios importantes no decorrer do projeto.

Os membros do grupo, no decorrer do projeto, evoluíram tanto tecnicamente através da implementação de uma aplicação utilizando o paradigma de orientação a objetos quanto no que diz respeito a conceitos de segurança e criptografia envolvidos na solução proposta.

Os trabalhos futuros propostos pelo grupo se dividem em dois grupos de atividades: os relacionados a melhorias do protocolo SecureTCG descrito e os relativos a implementações de provas de conceito do protocolo de segurança em ambientes que não foram explorados neste trabalho.

No que diz respeito a melhorias do protocolo SecureTCG, podemos destacar que no formato atual da solução o servidor do jogo participa tanto no processo de aquisição de cartas pelo jogador como nas trocas de cartas entre os jogadores. A possibilidade de os jogadores trocarem cartas sem a intervenção de um servidor central é uma questão que não foi explorada na versão atual do protocolo. Este tópico apresenta alguns desafios semelhantes aos encontrados no estudo de dinheiro eletrônico. Por exemplo, usar um cartão que foi trocado com outro jogador múltiplas vezes é semelhante a gastar a mesma moeda digital mais de uma vez. Dessa forma, avanços no protocolo SecureTCG de forma a incorporar tais problemáticas à narrativa do jogo é um item a ser potencialmente explorado no futuro.

Além disso, apesar do protocolo proposto englobar a maioria das operações básicos de um Trading Card Game, algumas operações particulares presentes em alguns

jogos como por exemplo uma carta presente em *Hand* ou em *Used* ser novamente inserida no Deck de Jogo não estão presentes no escopo da versão atual do protocolo. Cabe ressaltar também que operações específicas do jogo de prova de conceito como a ação de atacar um outro jogador não fazem parte dos requisitos de segurança do protocolo, de forma que a garantia de segurança destas também poderia constituir material para trabalhos futuros.

Já no que diz respeito a melhorias futuras relativas à implementação do jogo de prova de conceito, o desenvolvimento de aplicações focadas em dispositivos móveis também apresenta potencial para trabalhos futuros.

# 10 REFERÊNCIAS

BETHEA, D.; COCHRAN, R.; REITER, M. Server-side verification of client behavior in online games. ACM Transactions on Information and System Security (TISSEC) Journal, New York, NY, USA, v. 14, n. 4, a. 32, 2011.

BEVILACQUA, F. Building a Peer-to-Peer Multiplayer Networked Game. Tuts+ Free Game Development Tutorials, 2013. Disponível em: <http://gamedevelopment.tutsplus.com/tutorials/building-a-peer-to-peer-multiplayer-networked-game--gamedev-10074> Acesso em: 06 nov., 2015.

DIGITAL CARD GAMES REPORT. New York, NY, USA: SuperData, 2013. Disponível em: <https://www.superdataresearch.com/market-data/digital-card-games/> Acesso em: 26 jun., 2015.

ESSENTIAL FACTS ABOUT THE COMPUTER AND VIDEO GAME INDUSTRY. Washington, DC, USA: Entertainment Software Association, 2014. Disponível em: <http://www.theesa.com/wp-content/uploads/2014/10/ESA_EF_2014.pdf> Acesso em: 26 jun., 2015.

FIRST MODERN TRADING CARD GAME. London, UK: Guinness World Records, 1993. Disponível em: <http://www.guinnessworldrecords.com/world-records/first-modern-trading-card-game/> Acesso em: 30 out., 2015.

GAUDIOSI, J. Mobile game revenues set to overtake console games in 2015. Fortune - Fortune 500 Daily & Breaking Business News. Jan. 15, 2015. Disponível em: <http://fortune.com/2015/01/15/mobile-console-game-revenues-2015/> Acesso em: 26 jun., 2015.

LEAL, R. et al. SecureTCG: A lightweight cheating-detection protocol for P2P multiplayer online trading card games. São Paulo, Universidade de São Paulo, 2013. 29 p.

NEUMANN, C. et al. Challenges in Peer-to-Peer Gaming. ACM SIGCOMM Computer Communication Review Newsletter, New York, NY, USA, v. 37, n. 1, p. 79-82, 2007.

PITTMAN, D.; GAUTHIERDICKEY, C. Match+guardian: a secure peer-to-peer trading card game protocol. In MULTIMEDIA Systems, Springer Berlin Heidelberg, 2013. v. 19, n. 3, p. 303-314.

ROCA, J. Contributions to Mental Poker. 2005. Tese (PhD) – Universitat Autònoma de Barcelona, Barcelona, 2005.

ROTHAERMEL, F.; KOTHAAND, S.; MOXON, D. Wizards of the Coast. Seattle, WA, USA: University of Washington Business School, 1998. 12 p. Case para discussão em aula. Disponível em: <http://faculty.bschool.washington.edu/skotha/website/cases%20pdf/Wizards%20of%20the%20coast%201.4.pdf> Acesso em: 30 out., 2015.

SIMPLICIO, M. Segurança em Redes de Computadores - Funções de Hash, Códigos de Autenticação e Números Aleatórios. São Paulo: Epusp, 2015. p. 11. Slides utilizados para aulas da disciplina PCS2582 – Segurança da Informação.

WILLIAMS, J. Consumption and Authenticity in the Collectible Strategy Games Subculture. In: WILLIAMS, J.; HENDRICKS, S.; WINKLER, W. Gaming as Culture: Essays on Reality, Identity and Experience in Fantasy Games. Jefferson, NC, USA: McFarland & Company, 2006. p. 77-99. Disponível em: <http://www3.ntu.edu.sg/home/patrick.williams/PDFs/Williams%20-%20CSGs.pdf> Acesso em: 30 out., 2015.

# APÊNDICE A – DIAGRAMAS DE PACOTES