Cheat-Proof Peer-to-Peer Trading Card Games

Daniel Pittman
Department of Computer Science
University of Denver
dpittman@cs.du.edu

Chris GauthierDickey
Department of Computer Science
University of Denver
chrisg@cs.du.edu

***Abstract*—In trading card games (TCGs), players create a deck of cards from a subset of all cards in the game to compete with other players. Each card in the deck has some feature or ability that may be used strategically to help a player defeat her opponent. Recently, trading card games have been moving from physical cards to digital, online versions. We propose a cheat-proof peer-to-peer protocol for implementing online trading card games. We break down actions common to all TCGs and explain how they can be executed between two players without the need for a third party referee (which usually requires an unbiased server). In each action, the player is either prevented from cheating or if they do cheat, the opponent will be able to prove they have done so. We conclude by showing how these methods are secure and how they may be intermixed for other styles of TCGs and other peer-to-peer games.**

I. Introduction

Trading card games (TCGs), also known as collectible card games, are a type of card game where players purchase, collect, or trade cards, allowing them to create a playing deck from their collection which can be used to compete with other players. Cards in the game have different features or abilities and may be used strategically and in conjunction with other cards in their deck. Recently, TCGs have started to move to the computer game realm and typically use client/server architectures primarily because a centralized location is needed for handling game transactions, but also because servers can act as a referee for game play, thereby preventing players from cheating. The question naturally arises: can TCGs be played without cheating in a purely peer-to-peer fashion? We present a protocol showing that this is indeed possible.

The motivation for using peer-to-peer (P2P) protocols instead of client/server protocols is that they can provide low-latency communication and scalable services, but they introduce problems with cheating. Further, P2P protocols have the problem that today, many peers cannot directly connect to each other due to network address translation (NAT). While outside the scope of the protocol, we discuss NAT issues in Section II. On the other hand, P2P protocols allow messages to be sent directly (or almost directly) between peers, reducing the latency of messages. This is important for TCGs with *instant*-type cards, which are cards that are played immediately in response to an action by the opponent, as it allows the game play to progress quickly. In addition, P2P protocols can scale to a large number of simultaneous players since a game company would no longer be required to host online game servers. Finally, on the rare occasions when users do not have full connectivity to the Internet, players would still be able to compete with a P2P TCG protocol (e.g., between mobile devices on an airplane).

Our protocol is concerned primarily with preventing or catching cheating within the game play of the TCG. We assume that a system already exists for securely purchasing digital cards from a vendor (see Section IV for details). For a more complete architecture for online TCGs, one would likely need various additional services, and some of these would necessarily be client/server, especially those involving financial transactions.

Modern trading card games consist of several styles of play, including sealed-deck tournament games, draft games, and constructed deck games. We decompose these types of games into the primary sets of actions, from randomly and fairly generating decks of cards, to ensuring that any card played came from a deck that was fixed prior to when play commenced. We then show how to securely perform these steps so that all styles of play can be supported.

One might assume that the problems in TCGs are exactly those in the game of *mental poker* [1], a fictional game where two people can play poker without seeing each other's cards (e.g., over a phone without a built-in video camera). However, TCGs are different in that the deck of cards is not shared between players, which nicely side-steps the impossibility results of mental poker. Further, TCGs typically have different types of rules of play and therefore require slightly different techniques to prevent cheating.

As with many card games, most TCGs do not have specific time limits for playing individual cards to resolve a turn, except those limits associated with social norms. This gives TCGs some advantage in preventing cheating since certain types of cheats no longer apply when time constraints are not tightly bound. For example, research in the past has looked at cheating in specific types of games such as role-playing, first-person shooters, and real-time strategy games [2]–[4]. However, this prior work does not address cheating *within* the actual game such as by not actually shuffling cards, or choosing your most desired card instead of the next one from the top of your deck. In particular, we see that many of the issues related to cheating in TCGs can be solved by securely and fairly generating random numbers. This is mainly because TCGs rely on random generation of decks and the fair shuffling of those decks for play. To the best of our knowledge, this is the first work to address cheating in peer-to-peer TCGs.

To appear in the 10th Annual ACM/IEEE Network and Systems Support for Games (NetGames 2011).

II. Background

Most modern trading card games have their roots in Magic: The Gathering™, which was released in 1993. The game consists of a complete library of cards where players build their own collection by purchasing packs of cards. Each individual card has some level of rarity such that the more rare a card is, the fewer copies are produced and randomly distributed in card packs. Each card pack has some guarantee of containing a set ratio of very rare, rare, and common cards. For example, a card pack may be guaranteed to contain at least one very rare card, three rare cards, and the rest common cards. As one may guess, the rarest cards tend to be the most valuable and are often the most *powerful* in terms of gameplay, thereby creating an economy around collecting the rarest cards. Other well-known examples of trading card games include Pokémon™, the World of Warcraft Trading Card Game™, and the Yu-gi-oh! Trading Card Game™.

The idea of playing cards in a distributed fashion without cheating, termed *mental poker*, was first discussed by Shamir et al. [1]. The authors show that its impossible to deal from a *shared deck* of cards without one player knowing what card another player received. They then described a protocol that relies on commutative encryption to solve this problem. Note that their impossibility result still holds: if one could break the encryption in a reasonable amount of time, dealing from a shared deck of cards without revealing who received which card to the other player is still impossible. The primary difference between this problem and our problem is that in our case, the deck of cards between players *are not shared*. Without a shared deck, our solution can avoid using commutable encryption. As with their solution, we rely on the encryption not being easily breakable, where *easy* means within the span of time it takes to complete a game.

The protocol in this paper relies on being able to generate a random number between two or more players fairly. In this case, we define fairly as either player not being able to influence the outcome of the generated random number and this is solvable by the well-known coin flipping by telephone protocol [5]. With this method, Alice picks a random number r_A, cryptographically hashes it and sends the hash, $H(r_A)$ to Bob. Bob picks a random number, r_B, signs it and sends it to Alice. Alice then XORs r_A and r_B to determine the final value. Alice can then reveal the result later by giving Bob her private number allowing Bob to compute the same value. Note that since Alice has no idea what number Bob will pick, she cannot influence the final random value. Furthermore, Bob cannot influence the final value since he has no idea what initial value Alice chose. Expanding this to n players requires that each player generates a private number and 1 public number to share with the other $n-1$ players. Each player then XORs their private number with the $n-1$ other public numbers.

Cheating in games can be categorized by the *level* at which it occurs: the game, application, protocol or network [4]. Much of the research in cheat prevention in games has looked at preventing protocol or application level cheats, i.e., those cheats which occur by modifying network protocol behavior or altering the application in order to gain an unfair advantage [2]–[4], [6], [7]. In this paper, we focus specifically on avoiding or detecting *game level* cheats, which are those cheats that occur by breaking the rules of the game. We detail this further in Section IV.

As a peer-to-peer protocol, users invariably have to deal with the problem of network address translation (NAT). Using consistent endpoint translation in the NAT and *TCP hole-punching*, 80-90% of the peers can successfully connect to each other [8]. However, this does not account for the case where two users are behind the same NAT trying to connect to other users. In this case, a 3rd party would need to be used for relaying port numbers prior to TCP hole-punching. However, this is outside the scope of our protocol.

III. Play Styles

In modern trading card games there are multiple styles of play. For each style, there are different steps and techniques associated with them. When referring to the different gameplay steps, play styles, and card selection techniques, it is important to be clear on the terminology involved.

A. Definitions

1) **Universe deck** (D_u) - The set of all cards that exist in the game. This set is defined by the manufacturer of the trading card game.
2) **Base deck** (D_b) - The set of all cards a player owns and is therefore authorized to use during a gameplay session. Note that, $D_b \subseteq D_u$, since any card outside of the *universe deck* cannot exist. Each player has their own *base deck*, determined by their purchases or trades.
3) **Play deck** (D_p) - The set of cards from their *base deck* a player has selected to use during this gameplay session. The *play deck* must be a subset of the *base deck*, thus $D_p \subseteq D_b \subseteq D_u$. The *play deck* is typically constructed ahead of time based on the synergies of using particular cards together.

B. Styles of Play

In modern trading card games, play styles can be divided into the following common forms:

1) **Sealed deck**, where each player begins with a fresh random set of cards. The random set of cards becomes the player's *base deck* for the duration of the session.
2) **Draft deck**, where each player *drafts*, or picks, cards from a random set of cards. The drafted cards become the player's *base deck* for the duration of the session.
3) **Constructed deck**, where each player has already purchased, collected, or traded cards to create their *base deck* from which a subset of cards are chosen to construct a *play deck*.

Although the gameplay styles of these three forms are unique, the individual steps that compose these styles have significant overlap. By decomposing these styles into discrete securable steps, we can reduce the problem space, allowing us

to present a common solution for each kind of problem faced by these different play styles. Note that from these common steps, new gameplay styles can be crafted.

C. Sealed Deck Play

In a sealed deck game, each player is given an unopened deck of cards which is used to strategically construct a play deck prior to the actual match. This set of cards represents the player's *base deck*. Sealed deck games come from tournaments, where the idea is that if the decks are chosen randomly (consisting of some distribution of common, rare and very rare cards), then the matches are more representative of the skills of the players. Beyond this initial step in creating the play deck, sealed deck games are similar to constructed play styles. In the online equivalent of this type of game, a randomly generated deck of k cards (from the entire universe of cards) must be chosen fairly for each player. The securable steps needed to play this game can be described as:

1) Randomly generate a set of cards from the *universe deck* to represent each player's *base deck*. Typically a server would perform the random deck generation, but in a peer-to-peer system, the protocol must handle this step.
2) Have each player select a *play deck* from their *base deck* in a verifiable manner.
3) Draw a card at random from the *play deck*. This simulates the step of shuffling cards.
4) Verify when a card is played that it came from the set of that players' *play deck*.

D. Draft Deck Play

In draft deck play, each player participates in N *draft steps*. In each draft step, a player starts with *draft deck* consisting of a small number of random cards from the *universe deck*. The player then selects one card from this deck and then passes the deck to the next player. This "select and pass" step repeats until all cards are selected, at which point the next draft step starts except with the order of passing reversed. After all draft steps have completed, each player uses their drafted cards as their *base deck* and then constructs a *play deck* from it. The securable steps needed to play this style of game are:

1) Randomly generate N sets of cards from the *universe deck* to represent each player's starting *draft deck* each draft round. Note that this problem can be reduced to holding N rounds of the random base deck generation step used in the sealed deck play style.
2) Pass a player's *draft deck* to another player in a verifiable manner. This verification is similar to the verification of a card during gameplay. The main difference is that all cards are verified at once instead of one at a time as they are played.
3) Have each player select a *play deck* from their *base deck* in a verifiable manner.
4) Draw a card at random from the *play deck*. This simulates the step of shuffling cards.
5) Verify when a card is played that it came from the set of that players' *play deck*.

E. Constructed Deck Play

Constructed play is a game where each player creates a *play deck* by strategically choosing a subset of cards from their *base deck* and then plays according to the card game rules. Their *base deck* consists of all cards which they have purchased or traded with other players and verification that a player owns a particular card can be achieved by verifying the signature from a purchasing authority. Constructed games represent those games where players or friends compete with each other. The securable steps needed to play this game can be described as:

1) Have each player select a *play deck* from their *base deck* in a verifiable manner. Selecting a *play deck* from a player's collection is no different than the problem of selecting a *play deck* from a randomly generated deck, so this problem can be solved similarly.
2) Draw a card at random from the *play deck*. This simulates the step of shuffling cards.
3) When a card is played, verify that it came from the set of that players' *play deck*.

F. Securable Steps

Now that we have enumerated the popular play styles and their individual steps, we can create a master list of securable play steps for online trading card games:

- Randomly generate a set of cards from the *universe deck* to represent each player's *base deck* (or *draft deck*).
- Pass a player's *base deck* (or *draft deck*) to another player in a verifiable manner.
- Have each player select a *play deck* from their *base deck* in a verifiable manner.
- Draw a card at random from the *play deck*.
- Verify when a card is played that it came from the set of that player's *play deck*.

With this common securable framework, game designers could mix and match game steps to create completely new styles of play, allowing flexibility in the game style. We acknowledge that while this list may not be completely comprehensive for creating all possible game styles, it may be possible to add new securable steps using similar ideas to those presented here.

IV. Protocols

For each of the play steps which must be secured, we have developed an appropriate method to ensure that a player cannot cheat in that step. Composing a game from these steps leads to a particular play style. We begin by describing our assumptions, notation, the list of threats we are attempting to prevent, and then detail the protocols individually.

A. Preliminaries

In order for our protocols to work successfully, we make a few important but reasonable assumptions. First, each player has a unique identifier. Second, the size of the *universe deck* in the TCG is n. Without a loss of generality, we assign the numbers $1...n$ as unique identifiers for each card. Third, we

assume that since duplicates of each card in the set of cards can exist in a player's deck of cards, each valid card has a second number from $1...m$. When a player purchases a card from a company, this card first has its unique identifier, and second has this monotonically increasing sequence number $(1...m)$ depending on how many of that card the player owns. Cards are then signed by the company with both numbers and with the player's unique identifier to prove that they were purchased legally.

1) Notation: We use the letter U to indicate the entire universe of cards in the trading card game, where $|U|$ is the number of cards in the set. $H(i)$ is the cryptographically secure hash of i while $E_A(i)$ is i encrypted by A (usually Alice in this case). A digital signature of i is noted as $S_A(i)$. Recall that $S_A(i) = E_A(H(i))$. As such, $S_A(i)$ does not reveal any information about i but can be held as proof that i was the value when i is either revealed (i.e., if using a public-key cryptography system) or if the key K_A was revealed with i since $K_A(S_A(i)) = H(i)$ and the cryptographic hash functions are known to all parties.

$S_c(cuid, sn, pid)$ indicates a card which is digitally signed by the company (S_c), with its card unique identifier ($cuid$), its purchased sequence number (sn) and the player's unique identifier (pid). sn is not a unique number, but the signed tuple $S_c(cuid, sn, pid)$ is a unique tuple.

B. Threat Model

For our threat model, we assume a typical computationally-bounded adversary, capable of injecting packets and passive listening. We assume standard cryptography will prevent the attacker from decrypting packets in a reasonable amount of time (i.e., before the end of the game). Peer-to-peer TCGs are susceptible to the following types of threats:

1) **Unfairness in Card Selection** - We must be sure that while cards are being generated for a player that the player cannot unfairly influence the outcome of that operation. For example, during the generation of a random deck for a sealed deck game, we must prevent a player from influencing the set of cards generated for the random deck.
2) **Discovery of Private Information** - We must ensure that an opponent cannot garner private data concerning which cards another player has during the information exchanges needed in the setup and running of a game-play session.
3) **Changing Cards at Playtime** - As with a real TCG, players cannot be allowed to add or remove cards from their play deck *during* game play. Thus, any played card must be able to be verified during game play that it was actually a card from the player's play deck.
4) **Collusion** - The mechanism developed for generating and verifying cards must be resistant to collusion. Group operations (such as generating a random *base deck*) must be performed in such a way as to mitigate the case where some, but not all, of the players in the game are attempting to influence the outcome.
5) **Replay** - We must be able to prevent an adversary from replaying moves they eavesdropped on so that they cannot fool another player into thinking that the replayed packet is a cheating attempt by the originator of the move.

To verify if specific game rules (such as how cards behave) are broken or not, each player must keep a log of the game. Since each card is identified and digital signatures are used with moves, one can prove if a player cheats by their signed invalid actions during gameplay. However, for a log system to work, an additional sequence number is required for each move in a match so that a total ordering on the TCG moves can be identified.

C. Securely Generating a Base Deck

We begin by describing how Alice and Bob fairly generate a random *base deck* from a *universe deck* in a purely distribution fashion. This deck forms the basis for providing a sealed deck for the sealed deck game style and the base decks for the draft play styles.

1) Alice randomly generates a private number i_A and a public number j_A.
2) Alice signs her private number and only sends the signature $S_A(i_A, \text{nonce})$ to Bob. Recall from our notation that this is simply Alice's digital signature of the tuple (i_A, nonce) and does not include the actual values.
3) Bob randomly generates a private number i_B and a public number j_B.
4) Bob signs his private number and gives $S_B(i_B, \text{nonce})$ to Alice.
5) Alice and Bob exchange the tuples $(j_A, S_A(j_A))$ and $(j_B, S_B(j_B))$. In other words, they exchange their public numbers and sign those numbers (so that they cannot later argue that they gave *different* public numbers).
6) Alice XORs j_B with i_A to create a new random number, k_A, while Bob XORs j_A with i_B to create k_B.
7) $k_A \mod |U|$ is the unique identifier of Alice's card from U, while $k_B \mod |U|$ is Bob's first card from U.

At any step, either of the players may refuse to continue. For example, after Alice gives Bob her $S_A(i_A)$ for a particular card, she may wait for Bob's $S_B(j_B)$ in step 5, but not give her $S_A(j_A)$ to Bob. If so, Bob can simply refuse to continue playing as nothing (but time) has been lost. As with the coin-flipping protocol, Alice cannot choose her i_A in such a way so that the resulting k_A is a card that she wants because she does not know j_B before she has encrypted and sent i_A to Bob. The same holds for Bob's choice of j_B—he cannot influence k_A so that Alice gets a poor card from the deck because he has no idea what i_A is. Thus, Alice and Bob can fairly and randomly choose a card from U to be part of their tournament deck.

The above sequence of steps can be repeated k times so that each player has an *base deck* of k cards. However, the players can speed up the processes by generating a sequence of private

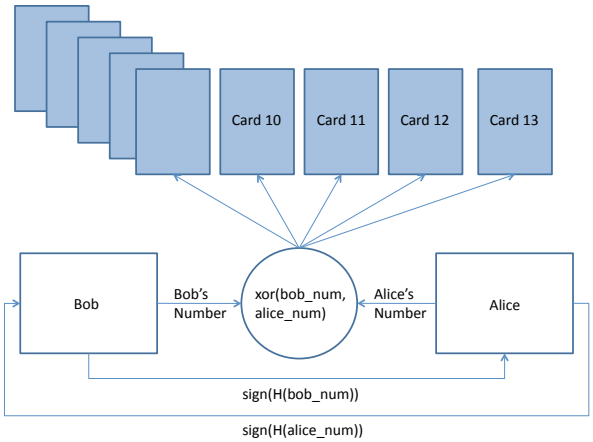

Fig. 1. *Random Card Selection*: This diagram shows how Bob and Alice can participate in random number selection in a verifiable manner while not revealing information about their random number to each other

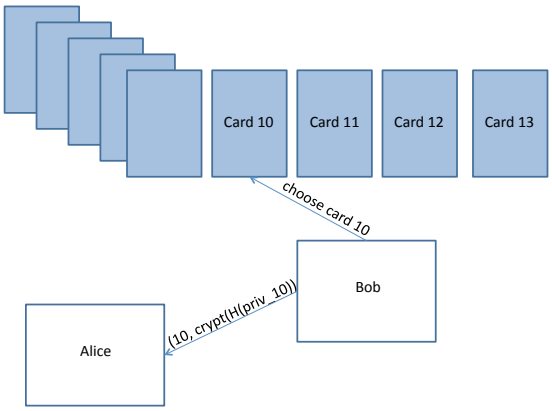

Fig. 2. *Play Deck Selection*: This diagram shows how Allice can be informed what cards Bob is selecting for his deck ahead of time without revealing the value of the card

and public numbers. In the first step, Alice generates k private numbers $i_{1A}...i_{kA}$ and public numbers $j_{1A}...j_{kA}$. Bob does the same thing for private and public numbers. In the second and fourth steps, each private number is signed individually (instead of encrypting the entire sequence) since the values and nonces are revealed as the cards are played to show that they indeed came from the *base deck* of k cards.

At this point, Alice and Bob have *base decks* consisting of k cards. Figure 1 diagrams the flow of steps for random card selection.

D. Play Deck Creation

Once a *base deck* of k cards has been selected, Alice and Bob must typically choose a subset of s cards from the *base deck* to form their *play deck*. Note that Alice and Bob choose cards for their *play deck* strategically as certain cards may work better with other cards. Further, the *play deck* does not have a specific size (i.e., Alice and Bob need not have exactly the same sized *play deck*) since for strategic reasons they may choose to construct a larger (for more variety) or smaller (for a higher probability of certain cards) deck.

The primary rule for creating the *play deck* is that it must be done entirely before the game begins. One cannot add cards to the *play deck* from the *base deck during* gameplay. Thus, the following steps must occur to fairly choose the *play deck*:

- Alice chooses a card for her *play deck*. Recall that Bob sent her k public numbers for each of the k cards in her *base deck*. Alice sends $(p, S_A(p))$ where p corresponds to the order of the $1...k$ values Bob sent her. For example, if the card she chose for her *play deck* was selected by XORing her 5th private number with his 5th public number, she sends $(5, S_A(5))$ to Bob. Alice repeats this for every card she adds to her *play deck* from her *base deck*. This prevents Bob from knowing Alice's play deck, though he knows her base deck.
- Bob chooses a card for his *play deck*, sending her $(p, S_B(p))$ for his chosen card, where p is the order of the public values correponding to the card he chose. Bob repeats this step for every card he adds to his *play deck* from his *base deck*.

Choosing the *play deck* must occur before gameplay begins and both Alice and Bob may create their decks simultaneously, though order does not matter in this case. When Alice or Bob play a card from their *play deck*, they reveal the associated private number and the order value (which they sent to represent each card). For example, when Alice plays the card that was chosen by Bob's 5th public number, Alice sends the tuple $(5, i_5, S_A(5, i_5))$ to Bob. As Bob knows his 5th public value and was previously sent the hash of Alice's 5th private value, he can calculate the hash of i_5 to see if it matches the previously sent hash. Further, he can XOR i_5 with his 5th public number to determine the $cuid$ of the card and verify that the correct card was played.

A diagram describing the process of selecting a card from the *base deck* is shown in Figure 2.

E. Drawing a Card from the Play Deck

Once a *play deck* has been created, we need to ensure that when a player chooses a card randomly from their deck during gameplay that the card they chose is truly random. In a physical game, the decks are shuffled and opponents may even *cut* the cards to introduce additional randomness. Players in fact typically want their cards shuffled so that they get an even distribution of various types of resource cards while playing as they cannot predict how the game will unfold. However, since the players cannot observe each other during play, we have to ensure that we get the equivalent of a deck shuffle without cheating. The protocol for this scenario is described below:

1) Alice's *play deck*, consisting of p cards are shuffled. Recall that she has already told Bob which cards are in her *play deck* by referring to them by their pth order value.

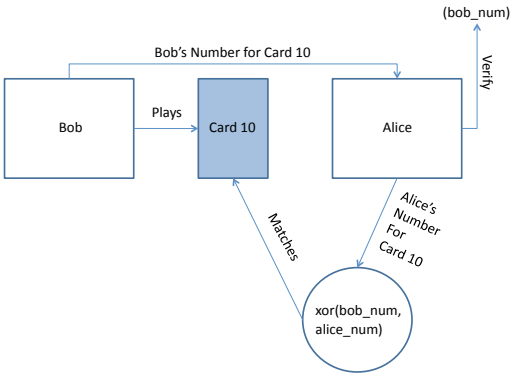

Fig. 3. *Card Verification*: This diagram shows how Allice can verify a card played by Bob during gameplay, allowing for real-time verification of correctness in gameplay

2) For each card in her deck, Alice sends $S_A(p, \text{nonce})$ to Bob.
3) Bob further shuffles the deck to ensure that Alice shuffled it properly. When Alice needs a card, she simply asks Bob for the next one, which he sends.

Bob repeats the procedure for his *play deck* so that Alice can ensure his cards are shuffled. When either player plays a card, they can reveal the values (p, nonce) so that the other player can verify that the card is not still in his or her deck from which cards are being dealt.

F. Playtime Verification of Cards

Playtime verification of cards is a two step process. First, the card has to be determined to be legitimately selected from the *base deck*. Second, the card has to be verified as a card that exists in the *play deck*.

Base deck verification depends on how the cards were generated, either randomly or user-selected. If the card comes from a user-selected *base deck*, the verification step is simply verifying the purchasing authority's signature on the card to ensure that the player has legitimately purchased that card. In the randomly generated *base deck*, since both Alice and Bob know the set U, and hence all the cards have unique identifiers, it suffices for a player to reveal the kth private value with its associated nonce to the other player which the other player can then XOR with the kth public value and match the unique identifier with the card just played. For example, Alice plays a card that was generated from the 5th public number, $(j_{5B}, S_B(j_{5B}))$, Bob gave to her. When she plays the card, she also sends $(i_5, \text{nonce}_5), S_A(i_5, \text{nonce}_5)$ to Bob. Using i_5 and nonce_5, Bob can check to see if this was the same value that Alice gave to him previously. If not, he knows she is cheating and has proof of it since he already has her hash of $H_A(i_5, \text{nonce}_5)$ that she sent in step 2.

A diagram describing the process of verifing that a card came from a player's *base deck* is shown in Figure 3.

Play deck verification occurs when Alice needs to play a card from her *play deck*. Alice sends the tuple $(p, i_p, S_A(p, i_p))$ where p indicates the order value of the public and private numbers for generating the card. Since she has already told Bob what order values she was using previously, he can easily verify if she is lying or not about its real value.

G. Passing a Base Deck to Another Player

When a *base deck* (or *draft deck*) is passed amongst players, it is important to make sure that the deck of cards being passed is not changed. Assuming the decks being passed were generated using the algorithms described above, verification occurs when the private values used to choose the base deck are revealed. Note that this occurs only when using the Draft Deck or Sealed Deck play styles. For example, Alice who generated a random base deck would reveal all the k private numbers $i_{1A} ... i_{kA}$.

With multiple players, revealing the private numbers for the randomly generated base decks does not leak information as all players must know all finalized base decks before play begins. Furthermore, a player cannot insert a new card for this exact same reason.

V. Conclusions and Future Work

In this paper we have demonstrated how using a set of common peer-to-peer protocols, one can develop multiple TCGs using different play styles while ensuring cheat-proof play. This common framework for peer-to-peer based card games enables the community to develop new styles of play without having to resolve the common problems facing games of this genre.

We believe there is a connection between generating sealed decks for online TCGs and automatically generating the monsters, treasures and maps for a game level. One could apply these ideas for fairly generating game content between players, especially if those levels are of a competitive nature.

For future work, we plan on evaluating the performance of these protocols in depth and developing other aspects of a P2P TCG architecture. We further plan on developing a common library for TCGs that will implement the protocols developed in this paper allowing game developers to focus on game design without having to worry about cheating.

References

[1] A. Shamir, R. L. Rivest, and L. M. Adleman, "Mental Poker," *The Mathematical Gardner*, pp. 37–43, 1981.
[2] N. E. Baughman, M. Liberatore, and B. N. Levine, "Cheat-proof playout for centralized and peer-to-peer gaming," *IEEE/ACM Trans. Netw.*, vol. 15, pp. 1–13, February 2007.
[3] C. Chambers, W.-C. Feng, W.-C. Feng, and D. Saha, "Mitigating information exposure to cheaters in real-time strategy games," in *Proceedings of ACM NOSSDAV*, 2005, pp. 7–12.
[4] C. GauthierDickey, D. Zappala, V. Lo, and J. Marr, "Low latency and cheat-proof event ordering for peer-to-peer games," in *Proceedings of ACM NOSSDAV*, June 2004.
[5] M. Blum, "Coin flipping by telephone a protocol for solving impossible problems," *SIGACT News*, vol. 15, no. 1, pp. 23–27, January 1983.
[6] E. Cronin, B. Filstrup, and S. Jamin, "Cheat-proofing dead reckoned multiplayer games," in *International Conference on Application and Development of Computer Games*, January 2003.
[7] C. Mönch, G. Grimen, and R. Midstraum, "Protecting online games against cheating," in *Proceedings of ACM NetGames*, Oct. 2006.
[8] F. Bryan, P. Srisuresh, and D. Kagel, "Peer-to-peer communication across network address translators," in *Proceedings of the USENIX Annual Technical Conference*, 2005.