A FAST MENTAL POKER PROTOCOL

TZER-JEN WEI AND LIH-CHUNG WANG

T.J. Wei and L.C. Wang at Department of Applied Mathematics, National Donghua University, Shoufeng, Hualien, Taiwan 97401

Abstract. We present a fast and secure mental poker protocol. It is twice as fast as similar protocols, namely Barnett-Smart's and Castellà-Roca's protocols. This protocol is provably secure under DDH assumption.

1. Introduction

1.1. **Mental Poker.** Mental poker is the study of protocols that allow players to play fair poker games over the net without a trusted third party. It can be considered as a kind of multiparty computation. In the study of mental poker, there are very few assumptions on the behavior of adversaries. Adversaries are typically allowed to have coalition of any size and can make active attacks.

The apparent application of mental poker is playing online poker game over the Internet. However, it is not easy to design a fast enough protocol to satisfy practical needs. Despite many protocols have been proposed ([2, 3, 7, 9, 8, 16, 15, 18, 20, 21, 23, 25, 27, 28, 29]), online poker rooms are still based on client-server architectures. Therefore, online players are assumed to trust the server. However, it is not uncommon for players to question the integrity of online games. These players might be right. In fall 2007, there is a major employee cheating scandal occurred at a famous online poker room, Absolute Poker. In 2008, similar scandal occurred at another famous online poker room, UltimateBet (see http://en.wikipedia.org/wiki/Online_poker for detail and news sources).

Therefore, an efficient decentralized poker protocol is desirable. We present a fast protocol in this paper.

1.2. **Previous Works.** The first mental poker protocol was proposed by Shamir et al in 1979 ([25]), which allows only two players to play. Unfortunately, it has a security flaw (see [24, 20]). The first secure mental poker protocol is proposed by Crépeau in 1987 ([15]). Since then, several other secure protocols have been proposed([2, 3, 7, 9, 8, 16, 15, 18, 20, 21, 23, 25, 27, 28, 29], see [7] for a survey).

Barnett-Smart's protocol is proposed in 2003 ([3]). It can be implemented by using either ElGamal or Paillier encryption scheme. However, Paillier encryption based version depends on Boneh-Franklin's protocol ([4, 5]), which is only secure under the assumption that adversaries are coalition of size at most $\frac{N-1}{2}$, where N is the number of players. In this paper, we consider active adversaries with coalition of size up to $N-1$. Therefore, in the rest of this paper, we consider only the ElGamal-based version.

Castellà-Roca's protocol is proposed in 2004 ([7]). It is similar to Barnett-Smart, but faster than Barnett-Smart in the shuffle.

Both Barnett-Smart's and Castellà-Roca's are secure and efficient.

1.3. **Our Result.** We present a fast and secure mental poker protocol. It shares the similar basic structure with Barnett-Smart's and Castellà-Roca's protocols, only the card encryption scheme is different. However, this difference is significant. Conseqently, the security proof of Barnett-Smart and Castellà-Roca does not work on our protocol.

In Barnett-Smart and Castellà-Roca, every player generates a private key at the beginning of the protocol. The private keys are used to turn the cards "face down". In each round of shuffling, each player generates a temporary secret to hide the permutation of the cards. To shuffle a deck of face down cards, only the temporary secrets are needed. To turn a card "face up", only the private key is needed. Since the shuffle only depends on the temporary secrets and the card encryption scheme only depends on the private key, we can prove the security of the shuffle, card dealing and opening separately. Then use composition theorem to show the security of whole protocol.

In our protocol, however, the same secret is used for both shuffle and card decryption.

Before further discussion, let us briefly describe the idea of the card encryption procedure in our protocol. Let G be a cyclic group and $g \in G$ a generator. Each card i is represented by an element $a_i \in G$. These a_i are chosen from G randomly via a multiparty protocol, so that a_i are indistinguishable from independent uniform random variable (under DDH assumption, which we discuss below). A face-up deck of M cards can be considered as the set $\{a_i\}_{i \le M}$. When a player, say Player j, wishes to shuffle the deck $\{a_i\}$, he privately chooses a new random secret x_j and then encrypts the deck as $\{a_i^{x_j}\}$ and the generator as g^{x_j}.

At some point of dealing a card, other players send an element $b \in G$ to Player j. Player j then send back $b^{x_j^{-1}}$ to other players. The owner of the card then use this information to decrypt the card. Obviously, if b can be freely chosen by other players, they can easily break Player j's shuffle. Therefore, there must have some restrictions on b in the card dealing protocol. So, there is no way to prove the security of the shuffle alone without investigating the card dealing protocol.

On the other hand, each card encryption requires only one exponentiation in our protocol. In Barnett-Smart and Castellà-Roca, each card encryption requires two exponentiations. Therefore, our protocol is roughly twice as faster. Detail comparison can be found in Section 4.

The security of our protocol depends on an intractability assumption, namely, Decisional Diffie-Hellman (DDH) assumption. This assumption is widely used in cryptography. There are many cryptographic primitives based on DDH assumption. For example, ElGamal encryption scheme ([17]), Diffie-Hellman key exchange, Cramer-Shoup cryptosystem ([13]). The security of Barnett-Smart and Castellà-Roca is also depends on DDH assumption.

Let Γ be a family of cyclic groups. DDH assumption (for Γ) states that, for any generator $g \in G \in \Gamma$, the following two distributions

- $\left(g, g^a, g^b, g^{ab}\right)$, where a, b are independent uniformly random;
- $\left(g, g^a, g^b, g^c\right)$, where a, b, c are independent uniformly random;

are indistinguishable.

DDH assumption is believed to be true for some families of groups. The typical example is the group of quadratic residues modulo a safe prime (i.e. prime of the form $2p+1$ where p is a prime). It is also believed to hold on a prime-order elliptic curve E over the field $GF(p)$, where p is prime and E has large embedding degree. More detail can refer to [6].

DDH assumption implies that the following two distributions

- $(a_0, a_1, a_2, \dots, a_M, a_0^x, a_1^x, \dots, a_M^x)$, where a_i, x are uniformly random;
- $(a_0, a_1, a_2, \dots, a_M, b_0, b_1, \dots, b_M)$, where a_i, b_i are uniformly random;

are indistinguishable (see [1]).

In other words, DDH assumption implies that the "shuffled deck", $\{a_i^x\}$ is indistinguishable from random variables $\{b_i\}$. This evidence strongly suggests the security of our protocol. However, as we discuss above, this result alone is not enough to prove the security of whole protocol. The proof is given in Section 3.

2. Protocol Description

2.1. **Overview.** The basic structure and usage of our protocol is same as those of Barnett-Smart and Castellà-Roca. Detail considerations and theoretical description can be found in [3].

The poker protocol can be divided into four parts: Deck Preparation (Protocol 1), Shuffle (Protocol 3), Card Drawing (Protocol 6) and Card Opening (Protocol 7).

To play a card game, players first use Deck Preparation to prepare a deck of cards. Players only need to prepare the deck once. After a deck being prepared, players can shuffle the deck or draw cards from the deck many times.

Players use Shuffle to shuffle the deck. When dealing cards, players can draw cards from the shuffled deck using Card Drawing. By using Card Opening, a player can show his hole cards to other players. Dealing a community card can be simulated by Card Drawing and Card Opening.

2.2. **Deck Preparation.** Let us fix a family of cyclic groups Γ that satisfies DDH assumption. We assume that there is a way to efficiently generate a group $G \in \Gamma$ for arbitrary large order and the group operation of G can be computed efficiently. For example, DDH assumption is generally believed to be true for the group of quadratic residues modulo a safe primes (a prime of the form $2p+1$ where p is a prime). For more detail consideration on the efficiency of G and Γ, please refer 4.1 of [14].

Let us fix a large prime n and a group $G \in \Gamma$ of order n. Consider there are N players playing with a deck of M cards. We name the cards in the deck as Card 1, Card 2, ..., Card M.

Protocol 1. *Deck Preparation*

(1) *Players generate distinct generators $a_i \in G$ for every $0 \le i \le M$ via some multiparty protocol, so that a_i are indistinguishable from independent uniform random variables (from the view of any proper subset of players).*
(2) *$\langle a_i \rangle_{0 \le i \le M} = a_0, a_1, a_2, \dots, a_M$ is the prepared deck of M cards.*

$\langle a_i \rangle_{0 \le i \le M}$ can be considered as the "face up" representation of the deck. a_0 is used as a "base" and for every $i \ge 1$, Card i is represented by a_i.

At step 1, players can choose any suitable protocol to generate a_i. For example, the following protocol is secure under DDH assumption.

Protocol 2. *Generate a random element*

(1) *For $j = 1$ to N, Player j does the following:*

   (a) *randomly choose generators $g_j, h_j \in G$ and randomly choose $0 < x_j < n$.*
   (b) *broadcast $g_j, g_j^{x_j}, h_j$.*
2) *For $j = 1$ to N, Player j does the following:*
   (a) *broadcast $h_j^{x_j}$.*
   (b) *use an auxiliary input zero-knowledge argument (see* [11], *for example) to convince other players that $\log_{g_j} g_j^{x_j} = \log_{h_j} h_j^{x_j}$.*
3) *The result element $h = \prod h_j^{x_j}$ is indistinguishable from a uniform random variable.*

The result h is indistinguishable from an independent uniform random variable if at least one player is honest.

2.3. **Shuffle.** Let $\langle a_i \rangle_{0\le i\le M}$ be a prepared deck of cards. To shuffle the deck, a player first encrypts the deck as $\langle a_i^x \rangle_{0\le i\le M}$ with a secret x. The encrypted deck $\langle a_i^x \rangle_{0\le i\le M}$ can be considered as a "face down" representation of the deck. Then the player can mix the cards up, so that the shuffled deck becomes

$$a_0^x, a_{\pi(1)}^x, a_{\pi(2)}^x, \ldots, a_{\pi(M)}^x,$$

where π is a permutation. Conversely, given a properly shuffled deck

$$\langle b_i \rangle_{0\le i\le M} = b_0, b_1, b_2, \ldots, b_M,$$

we can recover $x = \log_{a_0} b_0$ and π by comparing a_i^x and b_i (with unbounded computation power). That is, there is a unique face up deck corresponding to a properly shuffled deck. The player can use Protocol 4, which is a zero-knowledge proof, to convince other players that the result of his shuffle is proper.

Following is the detail description of the Shuffle protocol.

Protocol 3. *Shuffle*

1) *Let $B_0 = \langle b_{0,i} \rangle$, where $b_{0,i} = a_i$.*
2) *For $j = 1$ to N, Player j does the following:*
   (a) *randomly choose a secret integer $0 < x_j < n$;*
   (b) *randomly choose a permutation π_j of $(0, 1, 2, \ldots, M)$, such that $\pi_j(0) = 0$;*
   (c) *compute $B_j = \langle b_{j,i} \rangle$, where $b_{j,i} = \left(b_{j-1,\pi(i)}\right)^{x_j}$;*
   (d) *broadcast B_j to other players;*
   (e) *execute Protocol 4 with other players to prove his shuffle.*
3) *$B = B_N = \langle b_{N,i} \rangle_{1\le i\le M}$ is the shuffled deck.*

Player j use the following protocol to prove his shuffle to other players at step 2(e) of Protocol 3.

Protocol 4. *Shuffle Verification*

1) *Player j randomly chooses integers $0 < y_1, y_2, \ldots, y_K < n$.*
2) *Player j randomly chooses permutations $\pi'_1, \pi'_2, \ldots, \pi'_K$ of $(0, 1, 2, 3, \ldots, M)$.*
3) *Player j computes $C_k = \langle c_{k,i} \rangle_{0\le i\le M}$, where $c_{k,i} = b_{j,\pi'_k(i)}^{y_k}$ for $k = 1, 2, \ldots, K$.*
4) *For each $k = 1, 2, \ldots, K$*
   (a) *Player j broadcasts C_k to other players.*
   (b) *Other players cooperatively generate a random bit e_k via the multiparty protocol (see below)*
   (c) *Send e_k to Player j.*

(d) *If* $e_k = 0$, *Player* j *broadcasts* y_k, π'_k *and every player compute* $d_{k,i} = \left(b_{j,\pi'_k(i)}\right)^{y_k}$ *for every* i.
(e) *If* $e_k = 1$, *Player* j *broadcasts* $x_k y_k$, $\pi'_k \pi_j$ *and every player compute* $d_{k,i} = \left(b_{j-1,\pi'_k \pi_j(i)}\right)^{x_k y_k}$ *for every* i.
(f) *If* $d_{k,i} \neq c_{k,i}$ *for any* i, *then Player* j *does not pass the verification.*

(5) *Player* j *passes the shuffle verification.*

Let us discuss the multiparty protocol which is used to generates e_k at step 4(b). Players can, for example, use the following protocol:

Protocol 5. *Generate a random bit*

(1) *Every Player* j *commits a random bit* b_j.
(2) *Every Player* j *reveals the random bit* b_j
(3) *The output is* $b = \sum b_j \mod 2$.

So if at least one player is honest, then $\Pr(A(X) = b) < \frac{1}{2} + \epsilon$ for any efficient probabilistic algorithm A and any information X that one can get before the execution of Protocol 5. Therefore, if Player j does not shuffle properly, then the probability of players accepting Player j's shuffle is at most $2^{-K} + \epsilon$.

2.4. **Card Drawing and Opening.** Fix an arbitrary efficient auxiliary input zero-knowledge argument of equality of discrete logarithms (see [12, 3] for example). Player j_0 can use the following protocol to draw a card from the shuffled deck B.

Protocol 6. *Card Drawing*

(1) *Player* j_0 *picks a* $c_0 \in B$.
(2) *For* $j = 1$ *to* N, *one, Player* j *does the followings:*
(a) *if* $j \neq j_0$, *then compute* $c_j = c_{j-1}^{x_j^{-1}}$;
(b) *if* $j = j_0$, *then compute* $c_j = c_{j-1}$;
(c) *broadcast* c_j;
(d) *if* $j \neq j_0$, *use the zero-knowledge argument to convince other players that* $\log_{c_j} c_{j-1} = \log_{b_{j-1,0}} b_{j,0}$.

(3) *Player* j_0 *computes* $c = c_N^{x_{j_0}^{-1}}$ *and finds the* i *for which* $a_i = c$.
(4) *Card* i *is the card Player* j_0 *drew.*

After Player j_0 has drew a Card i, he can reveal the card to other players by the following protocol.

Protocol 7. *Card Opening*

Player j_0 *claims that he has Card* i *and use the zero-knowledge argument to show that* $\log_{a_i} c_N = \log_{b_{j_0-1,0}} b_{j_0}$.

3. Security Analysis

3.1. **Overview.** We shall compare our protocol to an *ideal card game*, or *ideal game* in short. In an *ideal game*, the shuffle is done by a trusted third party and no player can track the shuffle. No player can mark, steal, duplicate, or forge cards. No player can peek any face down card other than his own cards. However, players can communicate to each other via reliable and safe private channels and open channel. Moreover, every player is allowed to *surrender by declaring himself as* *cheater* any time in the game and loses the game immediatly. This is because that the adversary is allowed to cheat and being caught on purpose.

Players, including malicious players, are modeled as efficient auxiliary input Turing machines. The goal of a mental poker protocol is to allow players to play a card game over the network resembling an ideal game. We use the terminology "hand history" (or "game history") to denote the transcript of the idea game that the mental game tries to mimic. The hand history can be considered as the "pure card game" part of the transcript of a mental game. If a player surrenders (by declaring himself as cheater), then it should be recoreded as "cheating" in hand history.

To prove the security of our protocol, we show that the hand histories of mental games are indistiguishable from those of idea games. We have the following definition.

Definition 1. Let Z be a subset of players. Assume other players are honest. A mental poker protocol is said to be *secure against* Z if for any polynomial time strategy S of the mental game for Z, we can efficiently derive an expected polynomial time strategy S' of the ideal game for Z, so that the hand histories are indistinguishable.

A mental poker protcol is *secure* if it is secure against any proper subset of players.

Note that S' is expected polynomial time, not a polynomial time strategy like S. This is a situation not unlike that of the definition of zero-knowledge. Since the definition of zero-knowledge is generally accepted, this may not be a big issue. However, one may still wish to have the same notion of efficiency for both S and S'. It is possible to allow both S and S' being in a wider class of efficient machine (see [19, 22]).

We prove in Theorem 2 that our protocol is secure. Therefore, no player can increase his chance of winning a poker (or bridge, blackjack) game in our protocol by cheating. No player can lose a mental game more than he can in an ideal game. The information that a player learns in a mental game does not help his future games more than what he can learn in an ideal game.

Since players can private communication channels, there is no way to prevent coalitions entirely. What a mental poker protocol can do is to "minimize the effect of coalitions", as stated in Crépeau's requirements ([16]). That is, having coalitions should get no more advantage in a mental game than in an ideal game.

To simplify the proof of Theorem 2, we consider the worst scenario for the honest player. We can assume that there are 3 players in the card game and Player 2 is the only honest player, Bob. Player 1 and Player 3 are both played by the adversary, Alice. It is easy to check that it is enough to prove the security for this setting.

In order to simplify our proof, we introduce a sequence of games in Section 3.2. This a common technique to present an otherwise complicate security proof (see [26] for more information). Fix an event A of the card game. We define a series of games played by Alice and Bob, where the first game, Game 0, is the mental game and last game, Game 8, is the ideal game. For every k, Alice wins Game k if A occurs and she is not caught on cheating.

Fix a strategy S_0 of Game 0 for Alice. We show in Theorem 2 that there is a correspondence strategy S_k for Game k, such that Alice does no worse in Game k then in Game $k-1$.

3.2. **Games.** Let us fix a card game of 3 players. Since Bob is honest, he has a card game strategy that depends only on hand history and other information he supposed to know in the ideal game, like his hole cards. He use the same card game strategy to play all following games.

3.2.1. *Game 0.* Alice and Bob play the card game using our mental poker protocol. Bob plays the card game as Player 2. Player 1 and Player 3 are played by Alice. Bob follows the protocol properly but Alice may cheat. Let A_0 be the event that Alice is caught on cheating. When A_0 occurs, Game 0 is terminated.

3.2.2. *Game 1.* Game 1 is similar to Game 0, but after step 2(e) of Shuffle (Protocol 3), Bob attempts to extract x'_1 and x'_3 from Alice, if she passes Shuffle Verification (Protocol 4). where $x'_j = \log_{b_{j-1,0}} b_{j,0}$.

Suppose Alice passes the Shuffle Verification. Let $\langle e_k \rangle_{1 \leq k \leq K}$ be the bits generated at step 4(b) of Shuffle Verification. Bob uses Alice as a blackbox to extract x'_j:

Protocol 8. *Extract x_j*

For each $k = 1, 2, \ldots, K$,

(1) *Rewinds Alice back to step 4(b) for k of Shuffle Verification.*
(2) *Run step 4(b)-4(f) of Shuffle Verification and let e'_k be the random bit generated at step 4(b).*
(3) *If Alice does not passes the verification at step 4(f) of Shuffle Verification, got to step 1.*

If some e'_k is different from the e_k, then Bob knows both y_k and $x_j y_k$ and he can easily calculate $x'_j = x_j y_k / y_k$. Note that if Bob can extract x'_j, then B_j is properly shuffled by 4(d)(e) of Protocol 4.

Let A_1 be the event that Alice passes Shuffle Verification but Bob can not extract both x'_1, x'_3. Game 1 is terminated when A_1 occurs.

3.2.3. *Game 2.* Similar to Game 1, but Bob uses the knowledge of $x_1 = x'_1$ and $x_3 = x'_3$ to detect cheating. That is, in addition to the zero-knowledge argument at step 2(d) in Card Drawing (Protocol 6) and Card Opening (Protocol 7), Bob also checks whether $\log_{c_j} c_{j-1} = x_j$ directly for $j = 1, 3$ if $j \neq j_0$.

Let A_2 be the event that Alice is caught on cheating by the additional cheating detection. Game 2 is terminated when A_2 occurs.

3.2.4. *Game 3.* Same as Game 2, except that Bob uses a different way to decrypt cards at step 2(a) and 3 of Card Drawing (Protocol 6). Suppose $c_0 = b_{N,\pi_3\pi_2\pi_1(i)}$. Bob first use the knowledge of x_1, x_3 to recover π_1, π_3 efficiently.

If $j_0 \neq 2$, instead of computing $c_2 = c_1^{x_2^{-1}}$, Bob compute c_2 as

$$\begin{cases} a_i^{x_1 x_3} & \text{if } j_0 = 1 \\ a_i^{x_3} & \text{if } j_0 = 3 \end{cases}$$

at step 2(a) of Protocol 6.

If $j_0 = 2$, instead of computing $c = c_3^{x_2^{-1}}$, Bob compute $c = a_i$ at step 3 of Protocol 6.

Note that the value of c_2 and c remain the same, Bob merely uses a different way to compute them.

3.2.5. *Game 4.* Same as Game 3, except that

(1) At step 2(d) of Card Drawing (Protocol 6), Bob does not execute the zero-knowledge argument to prove $\log_{c_2} c_1 = \log_{b_{1,0}} b_{2,0}$. Instead, Bob runs the simulator for the zero-knowledge argument and generates a transcript that is indistinguishable to the real transcript.
(2) Bob does not use Shuffle Verification(Protocol 4) to prove his shuffle at 2(e) of Shuffle (Protocol 3). Instead, Bob uses the following simulator to generate a transcript:

Protocol 9. *Simulator for Shuffle Verification*

For each $k = 1, 2, \ldots, K$

(a) *Choose a random bit e', a random $0 < y < n$ and a random permutation π'*
(b) *If $e' = 0$, compute $\langle c_i \rangle_{0 \le i \le M}$, where $c_i = b^y_{j,\pi'(i)}$.*
(c) *If $e' = 1$, compute $\langle c_i \rangle_{0 \le i \le M}$, where $c_i = b^y_{j-1,\pi'(i)}$.*
(d) *Use Alice as a blackbox and run step 4(b) of Shuffle Verification to generate a bit e by treating $\langle c_i \rangle_{0 \le i \le M}$ as C_k.*
    (i) *If failed to generate e, generate a random bit f.*
    (ii) *Otherwise, let $f = e$.*
(e) *If $f \neq e'$, go to step (a).*
(f) *Write $\langle c_i \rangle_{0 \le i \le M}$, the transcript generated in step 2(d), e, y, π' into the transcript.*

3.2.6. *Game 5.* Same as Game 4 except that Bob uses a different way to generate $B_2 = \langle b_{2,i} \rangle_{0 \le i \le M}$ in Shuffle (Step 2(a)-(c) of Protocol 3).

Bob does not do Step 2(a)-(c) of Protocol 3. Instead, recall that $\langle a_i \rangle_{0 \le i \le M}$ is the face up deck generated in Deck Preparation (Protocol 1). Bob generates a random x and let $f_i = a_i^x$. He uses the knowledge of x_1 to recover π_1 and computes $b_{2,i} = f_{\pi_2 \circ \pi_1(i)}$, where π_2 is a random permutation that $\pi_2(0) = 0$.

3.2.7. *Game 6.* Same as Game 5, except that Bob generates uniformly random f_i and does not generate x.

3.2.8. *Game 7.* Same as Game 6, except that Bob uses a different way to encrypt cards. Instead of computing $b_{2,i} = f_{\pi_2(i)}$, Bob computes $b_{2,i} = f_i$. Bob still generates π_2 privately, which is used for card drawing (see the description of Game 3).

Put all modification together, Alice and Bob play Game 7 as following:

Deck Preparation: Protocol 1.

Shuffle:

Protocol 10. *Game 7 Shuffle*

(1) *Let $B_0 = \langle b_{0,i} \rangle_{0 \le i \le M}$, where $b_{0,i} = a_i$.*
(2) *Run step 2 of Shuffle (Protocol 3) to generate B_1.*
(3) *Rewind the game to extract x_1 from Alice (Protocol 8).*
(4) *Generate a random $B_2 = \langle b_{2,i} \rangle_{0 \le i \le M}$.*
(5) *Simulate the Shuffle Verification and generate an indistinguishable transcript (Protocol 9).*
(6) *Run step 2 of Shuffle (Protocol 3) to generate B_3 from B_2.*
(7) *Rewind the game to extract x_3 from Alic (Protocol 8).*

(8) *Let $B = \langle b_{N,i} \rangle_{0 \le i \le M}$ be the shuffled deck.*

Moreover, Bob also privately generates a permutation π_2 such that $\pi_2(0) = 0$.

After the shuffle, Bob can recover π_1, π_3 by x_1 and x_3. Let $\pi = \pi_3\pi_2\pi_1$. When Alice and Bob run steps from the original protocols, Alice runs the steps the same way as she would in Game 0.

Card Drawing:

Protocol 11. *When Player j_0 draws a face down card $c_0 = b_{i'}$ from a shuffled deck B.*

(1) *Bob computes $i = \pi^{-1}(i')$*
(2) *If $j_0 \neq 2$ (when Alice draws the card):*
  (a) *Run step 2(a)-(d) of Card Drawing (Protocol 6) to generate c_1.*
  (b) *Bob broadcasts*

$$c_2 = \begin{cases} a_i^{x_1 x_3} & \text{if } j_0 = 1 \\ a_i^{x_3} & \text{if } j_0 = 3 \end{cases}$$

  and uses the simulator to generate a fake transcript of zero-knowledge argument.
  (c) *Run step 2(a)-(d) of Card Drawing (Protocol 6) to generate c_3.*
  (d) *Alice can run step 3, 4 of Card Drawing (Protocol 6) to find out i.*
(3) *If $j_0 = 2$ (when Bob draws the card):*
  (a) *Run step 2 of Card Drawing (Protocol 6).*
  (b) *Bob knows that the card he drew is Card i.*

Card Opening:

Alice uses Card Opening (Protocol 7). Bob opens the card by showing i and then use the simulator to generate a fake transcript of zero-knowledge argument.

3.2.9. *Game 8.* Alice and Bob play the card game using the following protocol.

Shuffle: Bob randomly choose a π.

Drawing: Player j_0 picks a number $i_0 \le M$. Bob sends $\pi^{-1}(i_0)$ to Player j_0.

Opening: When a player wish to open a face down card i_0, Bob announces $\pi^{-1}(i_0)$.

This is the idea game where Bob acts as a trusted party. Bob uses the same card game strategy of Game 0 to play Game 8. Alice uses the partial information of π that Bob sent her and the real hand history of Game 8 to simulate a correspondent Game 7. Then she copies her next move in the simulation to play Game 8. If $A_2 \cup A_1 \cup A_0$ occurs in the simulation, Alice surrenders.

3.3. Security Proof.

Theorem 2. *Assume K is bounded by a polynomial of n and 2^{-K} is negligible, where $K = K(n)$ is the parameter in Shuffle Verification (Protocol 4). Assume the running time T of the mental game is bounded by a polynomial of n and all players are modeled as auxiliary input polynomial time Turing machine. If there is at least one honest player, then the mental game is secure.*

Proof. As discussed in Section 3.1, we fix a card game of 3 players. Alice plays as Player 1 and Player 3. Bob plays as Player 2 honestly. Both Alice and Bob are modeled as auxiliary input polynomial time Turing machines.

Fix an arbitrary polynomial time machine T. Let P_k be the probability that the output of $T(X)$ is 1, where X is a random hand history of Game k. We shall show that $|P_k - P_{k+1}| < \epsilon$ for $k = 0, \ldots, 7$, where ϵ is a negligible function.

($|P_0 - P_1| < \epsilon$)

Game 1 and Game 0 are otherwise the same except A_1 occurs. Thus, $|P_0 - P_1| \leq \Pr(A_1)$.

Recall that A_1 is the event that Alice passes the Shuffle Verification but Bob can not extract both x'_1, x'_3.

Let G_t be the event that a Shuffle Verification starts at time t (in Game 0) and Alice, as a prover, passes the Shuffle Verification.

Also let E_t be the event that G_t occurs and Bob can not extract x'_1 or x'_3 for the Shuffle Verification starts at time t.

Since the mental game has a polynomial time bound, we only need to show that $\sup_t \Pr(E_t)$ is negligible. The following lemma implies that $\sup_t \Pr(E_t)$ is negligible

Lemma. $\sup_t \Pr(E_t) < 2^{-K} + \epsilon$ for all t, for some negligible ϵ.

Proof. Fix an arbitrary m and a Shuffle Verification starts at time t , in which Player j (who is played by Alice) is a prover. Suppose Alice just broadcasts the C_k at step 4(a) of Shuffle Verification. Let p be the probability that $e_k = 0$ and Player j passes step 4(f) of Shuffle Verification. Let q be the probability that $e_k = 0$ and Alice passes step 4(f) of Shuffle Verification. So, the probability that Alices passes the shuffle verification and $e'_k = e_k$ is $\frac{p^2+q^2}{p+q}$ (e'_k is defined in of Protocol 8). Since $0 \leq p, q < \frac{1}{2}+\epsilon$, $\frac{p^2+q^2}{p+q} < \frac{1}{2}+\epsilon$ (see Protocol 5). E_t occurs iff Player j passses 4(f) for all K rounds and $e'_k = e_k$ for all $1 \leq k \leq K$. Thus, $\Pr(E_t) < \left(\frac{1}{2}+\epsilon\right)^K < 2^{-K} + \epsilon'$ for some negligible ϵ'. □

($|P_2 - P_1| = \epsilon$)

Recall that A_2 is the event that Alice fools the verifier in some of the zero-knowledge arguments. Game 2 and Game 1 are the same except A_2 occurs, so $|P_2 - P_1| \leq \Pr(A_2)$. By the soudness of the zero-knowledge argument, $\Pr(A_2)$ is negligible.

($P_2 = P_3$)

Bob uses a different way to decrypt cards that does not affect the result. Therefore, the transcripts of Game 2 and Game 3 are the same.

($|P_3 - P_4| = \epsilon$)

Observe that Protocol 9 is a simulator for Shuffle Verification (Protocol 4) . This is because the probability distributions of $\langle c_i \rangle_{0 \leq i \leq M}$ and e in simulation are identical to the genuine ones and independent to e'. Thus, the probability of $f = e'$ at step (e) is $\frac{1}{2}$, independent to $\langle c_i \rangle_{0 \leq i \leq M}$, e, and f. Thus, the simulated transcript has identical distribution as the genuine one.

So, the only difference between Game 3 and Game 4 is that zero-knowledge arguments in Game 3 are replaced by simulations in Game 4. Since simulated trascripts are indistinguishable from the genuine transcripts, $|P_3 - P_4|$ is negligible.

($P_4 = P_5$)

Bob uses a different way to generate B_2 that does not affect the result. Therefore, $P_4 = P_5$.

($|P_5 - P_6| = \epsilon$)

DDH assumption implies that the distribution of $\langle f_i \rangle_{0 \leq i \leq M}$ in Game 5 and Game 6 are indistinguishable. Since the game is played efficiently, $|P_5 - P_6| = \epsilon$.

($P_6 = P_7$)

Since $(f_i)_{i \leq M}$ is random, this is only a conceptional change to emphasize that π is information theoretically secure. Clearly, $P_6 = P_7$.

($|P_7 - P_8| < \epsilon$)

The first difference between Game 7 and Game 8 is that when $A_0 \cup A_1 \cup A_2$ occurs in Game 7, Alice surrenders Game 8. The probability that $A_1 \cup A_2$ occurs are negligible. Both A_0 and surrendering are the same as "being caught on cheating" in hand history.

The second difference between Game 7 and Game 8 is that $\langle a_i \rangle_{i \leq M}$ in Game 7 is generated by Deck Preparation (Protocol 1) and may not be genuine random. Since, $\langle a_i \rangle_{i \leq M}$ is indistinguishable to genuine random distribution from Alice's point of view, this difference is negligible. □

Sometimes, we may wish to study the utility function of cheaters.

Corollary 3. *Let X be a bounded random variable that can be computed efficiently from the hand history. Assume $E[X|A_0] = 0$ and $X \geq 0$. Then we have $E_0[X] \leq E_8[X] + \epsilon$, where $E_k[X]$ is the expectation of X for Game k.*

Proof. Let m be an arbitrary integer and A be the event that $\frac{i}{n^m} < X \leq \frac{i+1}{n^m}$, where i is considered as an auxiliary input. By Theorem 2, $P_0 < P_8 + \epsilon$. Since we may assume

$$i = \arg\max_j \left| \Pr\left(\frac{j}{n^m} < X \leq \frac{j+1}{n^m}\right) - \Pr\left(\frac{j}{n^m} < X \leq \frac{j+1}{n^m}\right) \right|,$$

we have

$$E_0[X] < E_8[X] + n^m \varepsilon + n^{-m} < E_8[X] + \frac{1}{2} n^{-m}$$

essentially. Therefore, $E_0[X] \leq E_8[X] + \epsilon$. □

4. Efficiency Analysis

4.1. **Computational cost.** In this section, we compare the computational cost (time) of our protocol to similar protocols, namely, Castellà-Roca ([7]), and Barnett-Smart ([3]).

All these protocols are discrete logarithm based. The most time consuming operations in these protocols are exponentiation and zero-knowledge argument of equality of discrete logarithms. In order to compare with the result of [7], the computational cost of multiplication is also considered. The computational cost of other operations are assumed to be much cheaper and can be ignored. Denote by z, e, m the computational cost of a zero-knowledge proof, an exponentiation, a multiplication respectively.

Assume the game played by N players with a deck of M cards. The cost of the Shuffle is compared in Section 4.2, and the cost of Card Opening and Drawing is compared in Section 4.2.

To give some ideas of empirical execution time and how practical these protocols might be, we make some estimations of execution time in 4.4.

4.2. **Shuffle.** Shuffle is usually the most time consuming part of a mental poker protocol.

Recall the security parameter K in Shuffle Verification (Protocol 4). We have the following table (the calculation of the computational cost of Castellà-Roca and Barnett-Smart can be found in [7]) .

Table 1. Computational cost for Shuffle

	Total cost	Cost for each player
Protocol 3	$\left(1+\frac{1}{M}\right)(KN+1)\,MNe+\frac{1}{2}KNm$	$\left(1+\frac{1}{M}\right)(KN+1)\,Me+\frac{1}{2}Km$
Castellà-Roca	$2\,(KN+1)\,MNe+\frac{1}{2}KMNm$	$2\,(KN+1)\,Me+\frac{1}{2}MKm$
Barnett-Smart	$2\,(KN+1)\,MN(e+m)+Mm$	$2\,(KN+1)\,Me+2\left(N+\frac{2N+1}{2KN}\right)MKm$

Our shuffle is roughly twice as fast as others. If the computational cost m of multiplication is ignored, then Castellà-Roca and Barnett-Smart have the same cost.

4.3. **Card Opening and Drawing.** Card Opening and Drawing are much cheaper compare to Shuffle. The following table compares the computational cost of Card Opening and Card Drawing.

Table 2. Total computational cost for drawing and opening

	Card Opening	Card Drawing
Ours	z	$(N-1)\,z+Ne$
Castellà-Roca	$z+(N-1)\,e$	$(N-1)\,z+\left(N+\frac{M}{2}\right)e$
Barnett-Smart	$z+N\,(N-1)\,m$	$(N-1)\,z+Ne+Nm$

Our protocol is faster than the rest, but only slightly. If the computation cost m of multiplication is ignored, then the computational cost of ours and Barnett-Smart are the same.

4.4. **Execution time.** To give some sense of empirical execution time, let us assume $M=52$ and $N=9$, which is typical for a full table poker game.

On an AMD X2 3800+ 2Ghz, which is fairly mediocrity in today's PC hardware standard, e and m are about 4.4×10^{-4} and 1.3×10^{-6} seconds for 512 bits integers (when using both cores). We have the following estimation.

Table 3. Computational cost (seconds) for each player (512 bits)

	$K=10$	$K=20$	$K=100$
Protocol 3	2.12	4.22	21.01
Castellà-Roca	4.16	8.28	41.23
Barnett-Smart	4.18	8.31	41.35

On the same machine, e and m is about 3×10^{-3} and 3×10^{-6} seconds for 1024 bits integers. We have the following estimation.

Table 4. Computational cost (seconds) for each player (1024 bits)

	$K = 10$	$K = 20$	$K = 100$
Protocol 3	14.47	28.78	143.26
Castellà-Roca	28.39	56.47	281.12
Barnett-Smart	28.42	56.53	281.39

The difference between Castellà-Roca and Barnett-Smart are less than 1% and ours is roughly twice as fast.

Considering it is reasonable to expect a human player taking 10 to 15 seconds to shuffle and cut a deck physically, these protocols seems to be nearly practical when using 512 bits primes and lower security parameter K. Since our protocol is the fastest, it is more close to be practical than others.

When using 1024 bits primes and $K = 100$, all protocols are too slow.

To estimate the execution time of Card opening and Card Drawing, assume using Chaum-Pedersen's protocol (see [10]) as the zero-knowledge argument. Thus, $z = (2N - 1)(2e + m)$. We have the following table.

Table 5. Total computational cost when using Chaum-Pedersen

	Opening + Drawing
Our protocol	$\left(4N^2 - N\right)e + \left(2N^2 - N\right)m$
Castellà-Roca	$\left(4N^2 - 1 + \frac{M}{2}\right)e + \left(2N^2 - N\right)m$
Barnett-Smart	$\left(4N^2 - N\right)e + \left(3N^2 - N\right)m$

Note that theoretically, Chaum-Pedersen's protocol is only known to be honest verifier zero-knowledge. However, it is widely used and it serves well for a rough estimation of empirical execution time. We have the following table.

Table 6. Total computational cost (seconds) when using Chaum-Pedersen

	Opening + Drawing (512 bits)	Opening + Drawing (1024 bits)
Our protocol	0.139	0.945
Castellà-Roca	0.154	1.047
Barnett-Smart	0.139	0.946

The computational cost of ours and Barnett-Smart are roughly the same, while Castellà-Roca is about 10% slower. The speed of Card Drawing and Opening of these protocols seems to be acceptable for practical use.

5. Conclusion

Our protocol is proved to be secure in Section 3 under DDH assumption. Theorem 2roughly states that cheating will be detected and other than that, the mental game is indistinguishable from the ideal game.

However, there are limitations of the security proof. For example, we assume the cheater loses if he is caught on cheating for every event A. To make this assumption practical, the penalty and compensation of cheating should be high enough. Moreover, the execution time of Game 8 is longer than Game 0. We implicitly assume the difference is insignificant.

Our protocol is fast. Considering the advance of computer hardware, efficient protocols like Castellà-Roca and Barnett-Smart may become fast enough to be practical in a few years. Our protocol is even faster, requires only half of the computing power to achieve same performance. We didn't discuss the communication costs of our protocol. However, for it can be easily verify that the communication cost of our protocol is also cheaper, roughly half as much compares to other protocols.

We hope our contribution can shorten the gap between theoretical study and the practical application of mental poker.

References

[1] F. Bao, R.H. Deng, and H. Zhu. Variations of diffie-hellman problem. *Lecture Notes in Computer Science*, 2836:301–312, 2003.

[2] I. Barany and Z. Furedi. Mental poker with three or more players. *Information and Control*, 59(1-3):84–93, 1984.

[3] A. Barnett and N. Smart. Mental poker revisited. *Proc. Cryptography and Coding*, 2898:370–383, 2003.

[4] D. Boneh and M. Franklin. Efficient generation ofshared RSA keys. *Advances in Cryptology-Crypto'97, Lecture Notes in Computer Science*, 1294:425–439, 1997.

[5] D. Boneh and M. Franklin. Efficient Generation of Shared RSA Keys. *Journal of the ACM*, 48(4):702–722, 2001.

[6] Dan Boneh. The decision diffie-hellman problem. *Lecture Notes in Computer Science*, 1423:48–63, 1998.

[7] J. Castella-Roca. Contributions to Mental Poker. *Thesis*, 2005.

[8] J. Castella-Roca, J. Domingo-Ferrer, and F. Sebe. On the Security of a Repaired Mental Poker Protocol. *Proceedings of the Third International Conference on Information Technology: New Generations (ITNG'06)-Volume 00*, pages 664–668, 2006.

[9] J. Castella-Roca, F. Sebe, and J. Domingo-Ferrer. Dropout-Tolerant TTP-Free Mental Poker. *LECTURE NOTES IN COMPUTER SCIENCE*, 3592:30, 2005.

[10] D. Chaum and T.P. Pedersen. Wallet databases with observers. *Preproceeding of Crypto*, 92:3–1, 1992.

[11] R. Cramer, I. Damgard, and P. MacKenzie. Efficient zero-knowledge proofs of knowledge without intractability assumptions. *Lecture Notes in Computer Science*, 1751:354–372, 2000.

[12] R. Cramer, I. Damgard, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. *Advances in Cryptology¡XCRYPTO*, 839:174–187, 1994.

[13] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. *Crypto*, 98:13–25, 1998.

[14] R. CRAMER and V. SHOUP. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM journal on computing*, 33(1):167–226, 2004.

[15] C. Crepeau. A zero-knowledge Poker protocol that achieves confidentiality of the players' strategy or How to achieve an electronic Poker face. *Proceedings on Advances in cryptology—CRYPTO'86*.

[16] C. Crepeau. A secure poker protocol that minimizes the effect of player coalitions. *Advances in Cryptology: Proceedings of Crypto*, 85:73–86, 1986.

[17] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 10–18, 1985.

[18] S. Fortune and M. Merrit. Poker Protocols, Advances in Crytology-CRYPTO 84 Proceedings, 1985.

[19] O. Goldreich. On expected probabilistic polynomial-time adversaries: A suggestion for restricted definitions and their benefits. *Lecture Notes In Computer Science*, 4392:174, 2007.

[20] S. Goldwasser and S. Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 365–377, 1982.

[21] P. Golle. Dealing Cards in Poker Games. *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05)-Volume I-Volume 01*, pages 506–511, 2005.

[22] J. Katz and Y. Lindell. Handling expected polynomial time strategies in simulation-based security proofs. *Theory of Cryptography: Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005: Proceedings*, 2005.

[23] K. Kurosawa, Y. Katayama, and W. Ogata. Reshuffleable and laziness tolerant mental poker game. *IEICE Trans. Fundamentals E*, 80:72–78, 1997.

[24] RJ Lipton. How to cheat at mental poker. *Technical Report, Computer Science Dept., Berkeley, CA, August*, 1979.

[25] A. Shamir, R.L. Rivest, and L.M. Adleman. Mental Poker. 1979.

[26] V. Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR ePrint Report*, 332, 2004.

[27] M. Yung. Cryptoprotocols: Subscription to a public key, the secret blocking and the multi-player mental poker game. *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 439–453, 1985.

[28] W. Zhao and V. Varadharajan. Efficient TTP-free mentalpokerprotocols. *Proceedings of ITCC*, pages 745–750, 2005.

[29] W. Zhao, V. Varadharajan, and Y. Mu. A secure mental poker protocol over the Internet. *C. Johnson, P. MontagueandC. Steketee, eds., Australasian Information Security Workshop*, 21:105–109, 2003.