J. Math. Cryptol. **6** (2012), 39–68
DOI 10.1515/jmc-2012-0004

A fast mental poker protocol

Tzer-jen Wei and Lih-Chung Wang

Communicated by Kwangjo Kim

Abstract. In this paper, we present a fast and secure mental poker protocol. The basic structure is the same as Barnett & Smart's and Castellà-Roca's protocols but our encryption scheme is different. With this alternative encryption scheme, our shuffle is not only twice as fast, but it also has different security properties. As such, Barnett & Smart's and Castellà-Roca's security proof cannot be applied to our protocol directly. Nevertheless, our protocol is still provably secure under the DDH assumption. The only weak point of our protocol is that reshuffling a small subset of cards might take longer than Barnett & Smart's and Castellà-Roca's protocols. Therefore, our protocol is more suitable for card games such as bridge, most poker games, mahjong, hearts, or black jack, which do not require much partial reshuffling.

Keywords. Mental poker, DDH assumption.

2010 Mathematics Subject Classification. 94A60, 68M12.

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. There are very few assumptions about the behavior of adversaries in mental poker. Adversaries are typically allowed to have a coalition of any size and can conduct active attacks.

The main challenge is to design a secure mental poker protocol that is fast enough for practical needs. Numerous mental poker protocols have been proposed ([4,5,10–12,17,18,20,25,26,28,30,34–36]) and many of them are provably secure, but all commercial online poker rooms are still based on client-server architectures. Therefore, online players are assumed to trust the central server. However, it is not uncommon for poker players to question the integrity of online games, and in some cases their suspicions are justified. In the fall of 2007, a major employee cheating scandal was uncovered at a famous online poker room, Absolute Poker.

This study was partially supported by National Science Council of Taiwan (NSC-96-2221-E-259-008-MY3)

In 2008, a similar scandal occurred at another famous online poker room, UltimateBet (see [33]). These scandals highlight the need for efficient decentralized poker protocols. A fast mental poker protocol is proposed in this paper.

1.2 Previous research

The first mental poker protocol was proposed by Shamir, Rivest and Adleman in 1979 ([30]) and allowed only two players to play at a time. Unfortunately, it had a security flaw (see [25, 29]). The first secure mental poker protocol was proposed by Crépeau in 1987 ([18]). Since then, several other secure protocols have been developed ([4, 5, 10–12, 17, 18, 20, 25, 26, 28, 30, 34–36], see [10] for a survey).

Barnett & Smart's protocol was proposed in 2003 ([5]). It can be implemented by using either the ElGamal or the Paillier encryption scheme. However, Paillier encryption based version depends on Boneh–Franklin's protocol ([7, 8]), which is only secure under the assumption that the maximum coalition size of adversaries is $\frac{N-1}{2}$, where N is the total number of players. In this paper, we study circumstances in which active adversaries may have a coalition size of up to $N-1$. As such, in the rest of this paper, any mention of Barnett & Smart's protocol refers to the ElGamal-based version of this protocol.

Castellà-Roca's protocol was proposed in 2004 ([10]). The major difference between Castellà-Roca's protocol and Barnett & Smart's protocol is their encryption scheme. Castellà-Roca's shuffle is a bit faster than Barnett & Smart's.

1.3 DDH assumption

The security of our protocol depends on an intractability assumption, namely, *the Decisional Diffie–Hellman (DDH) assumption.*

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

- (g, g^a, g^b, g^{ab}), where a, b are uniformly random,
- (g, g^a, g^b, g^c), where a, b, c are uniformly random.

The DDH assumption is believed to be true for several 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 true on a prime-order elliptic curve E over the field $\mathrm{GF}(p)$, where p is prime and E has large embedding degree. Readers are referred to [6] for more details.

The DDH assumption is widely used in cryptography. There are many cryptographic primitives based on the DDH assumption. For example, ElGamal encryption scheme ([19]), Diffie–Hellman key exchange, and Cramer–Shoup cryptosystem ([15]). The security of Barnett & Smart and Castellà-Roca also depends on the DDH assumption.

1.4 Zero-knowledge argument

An *auxiliary-input zero-knowledge argument* ([3, 22, 24]) for a language is an interactive protocol executed by a prover and a verifier that satisfies the following conditions:

- The input is some instance i. The prover knows a certain secret related to x so that she can efficiently verify that i belongs to the language.
- Completeness: If i belongs to the language, the honest verifier will be convinced of this fact by the honest prover.
- Computational soundness: If i does not belong to L, then the probability that the honest verifier will be convinced by a cheating prover is negligible.
- Zero-knowledge: For every cheating verifier, there is a simulator, and given only the input i and no access to the prover, it can produce transcripts that are indistinguishable from the transcripts of the interaction between the honest prover and the cheating verifier.
- Cheaters, simulators and the honest prover and verifier are all "efficient" in slightly different ways. The honest provers and verifiers are both probabilistic polynomial time machines. The cheaters are auxiliary input probabilistic polynomial time machines. The simulators are auxiliary input expected probabilistic polynomial time machines when considering constant round protocols and are strict probabilistic polynomial time machines otherwise.

Let us fix ZKA to be an arbitrary auxiliary input zero-knowledge argument of equality of discrete logarithms (see [14, 21] for example, these are all constant round protocols). That is, if the prover knows $x = \log_a b = \log_c d$, then she can use ZKA to convince the verifier to accept the fact that $\log_a b = \log_c d$ without revealing any information about x. ZKA is used in our protocol various times.

1.5 Our results

We will present a fast and secure mental poker protocol. It is similar to Barnett & Smart's and Castellà-Roca's protocol. The difference between these protocols and ours is the card encryption and decryption procedure.

In Barnett & Smart and Castellà-Roca, each player uses two kinds of secrets to shuffle the deck. Loosely speaking, the first kind of secrets is used to turn the cards face down at the beginning of the game. When shuffling, players mix up the cards and use the second kind of secrets to hide the permutation of the cards. The second kind of secrets is different for each round of shuffling and then is never used again during the game. Only the first kind of secrets is needed to decrypt a card. Therefore, the security of the shuffle, card dealing and opening can be proved separately. Composition theorems can then be used to show that all these sub-protocols can be replaced by ideal function and hence the whole protocol is secure.

In our protocol, however, each player uses only one secret to simultaneously turn the cards face down and mix them up. The same secret is also used for decrypting cards.

Before further discussion, let us briefly describe the idea of deck encryption in our protocol. Let G be a cyclic group and $g \in G$ be 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 the DDH assumption, which we discuss below). A face-up deck of M cards can be considered as the set $\{a_i\}_{i \leq M}$. When a player, say Player j, wishes to shuffle the deck $\{a_i\}$, he privately chooses a secret x_j and then encrypts the deck as $\{a_i^{x_j}\}$ and the generator as g^{x_j}. When dealing a card, x_j is used to decode the card.

Each card encryption requires only one exponentiation in our protocol. In the Barnett & Smart and Castellà-Roca protocols, each card encryption requires two exponentiations. Therefore, our protocol is roughly twice as fast. Detailed comparison can be found in Section 4.

It is known that the DDH assumption implies that the following two distributions are indistinguishable (see [1,9]):

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

In other words, the DDH assumption implies that the "shuffled deck", $\{a_i^x\}$ is indistinguishable from random variables $\{b_i\}$. This evidence suggests that the deck is mixed up randomly.

The situation is very similar for Castellà-Roca's and Barnett & Smart's protocols. The DDH assumption also implies that the shuffled deck is indistinguishable from a random deck. In fact, since the secret factor x is not needed anymore for decrypting the cards, players can discard or forget the secret x after the shuffle. Therefore, the DDH assumption can be applied to prove that both of these protocols are secure.

However, this is not the case for our protocol. The secret x is needed for decrypting the cards. As a consequence, the DDH cannot be applied directly to our protocol. Therefore, we needed to comprehensively study the shuffle, card dealing and opening and find a way to reduce the whole protocol into the DDH problem. The proof is given in Section 3.

The encryption and decryption of our protocol is simpler than the protocols of Barnett & Smart and Castellà-Roca. In our protocol, only one exponentiation is required to encrypt a card, and as a result, shuffling a full deck can be completed twice as fast as previous protocols.

The only weakness of our approach is that reshuffling a small pack of cards might be slower compared to other protocols. To reshuffle a small pack of cards using Barnett & Smart's and Castellà-Roca's protocols, players only need to encrypt a subset of the cards (with the second kind of secrets). Because only the first kind of secrets is needed to decrypt a card, players can use the same way to open a card no matter how the card has been shuffled. However, in our protocol, in order to decrypt a card, players need to use the secret x to mix up the deck. Every shuffle uses a new private key x to re-encrypt the cards. So if a card is being reshuffled several times, players need to use all secret keys x for these shuffles in order to decrypt the card. Therefore, when reshuffling a small part of the deck, players need to encrypt the rest of the deck with the same private key x as well, so that all cards can be decrypted with same keys. Consequently, the computational cost of a partial reshuffle is comparable to the cost of a full reshuffle. Therefore, our protocol is less suitable for card games such as old maid, I doubt and slapjack, which require many partial shuffle. Partial shuffling is not used in most poker games, bridge games, go fish, or blackjack. These card games benefit from using our protocol. Games like uno and ninety-nine requires partial shuffles only once in a while. Our protocol would likely induce some speed gain, but further study on the statistical behavior of these games is required to determine a definite conclusion.

2 Protocol description

2.1 Overview

The basic structure and usage of our protocol is the same as those of Barnett & Smart and Castellà-Roca. Detailed considerations and theoretical description can be found in [5].

The protocol can be divided into four parts: Deck Preparation (Protocol 1), Shuffle (Protocol 3), Card Drawing (Protocol 5) and Card Opening (Protocol 6).

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

The deck can be shuffled by players using Shuffle. When dealing cards, players can draw cards from the shuffled deck by using Card Drawing. The use of Card

Opening allows a player to show his cards to other players. Dealing a community card can be simulated by Card Drawing and Card Opening.

2.2 Deck Preparation

Fix a family of cyclic groups Γ that satisfies the DDH assumption. Assume that a group $G \in \Gamma$ of arbitrary large order can be efficiently found and the group operation of G can be computed efficiently. For example, G can be the group of quadratic residues modulo a large safe prime or a prime-order elliptic curve E over the field GF(p), where p is prime and E has large embedding degree. For further consideration of the efficiency of G and Γ, please see [16, Section 4.1]. Lastly, fix a group $G \in \Gamma$ with a large order p.

Consider if 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 is used to prepare the deck.

Protocol 1: Deck Preparation

(i) Players generate distinct independent random generators $a_i \in G$ for every $0 \leq i \leq M$ using Protocol 2 (see below).

(ii) $(a_i)_{0 \leq i \leq M} = a_0, a_1, a_2, \ldots, a_M$ is the prepared deck of M cards.

Protocol 2: Generate a random element

(i) For $j = 1, \ldots, N$, Player j does the following:

  (a) randomly choose generators $g_j, h_j \in G$ and randomly choose $0 < \lambda_j < n$,

  (b) let $g_j' = g_j^{\lambda_j}$,

  (c) broadcast g_j, g_j', h_j.

(ii) For $j = N, \ldots, 1$, Player j does the following:

  (a) let $h_j' = h_j^{\lambda_j}$,

  (b) broadcast h_j',

  (c) use ZKA (see 1.4) to convince other players that $\log_{g_j} g_j' = \log_{h_j} h_j'$.

(iii) Return the result $h = \prod h_j'$.

In Protocol 1, $(a_i)_{0\leq i\leq M}$ can be considered as the "face up" representation of the deck; a_0 is used as the "base" of the deck and for every $i \geq 1$, Card i is represented by a_i.

At step 1, players can choose any algorithm to generate random $a_{0\leq i\leq M}$ instead of Protocol 2, as long as the algorithm satisfies Theorem 3.3. Protocol 2 is our reference implementation based on the DDH assumption. When playing against passive adversary, Theorem 3.3 implies that $(a_i)_{0\leq i\leq M}$ are indistinguishable from genuine independent uniform random variables. In contrast, when against active adversaries, some adversaries may refuse to follow the protocol, so the protocol may not always successfully generate $(a_i)_{0\leq i\leq M}$. If a player fails to follow the protocol, then it counts as cheating and the player loses the game. Theorem 3.3 basically says that the result generated by Protocol 2 is indistinguishable from genuine random variables, but we may not always see the result because sometimes the protocol will not be successfully executed. We shall prove Theorem 3.3 in Section 3.2.

2.3 Shuffle a deck

We state that a deck is *properly shuffled* if the content of the deck has not changed with the exception of the order of the cards. The following is a formal definition.

Definition 2.1. A deck $(b_i)_{0\leq i\leq M}$ is *properly shuffled* from $(a_i)_{0\leq i\leq M}$ if there is an $x \neq 0$ and a permutation π, such that $\pi(0) = 0$ and $b_i = a^x_{\pi(i)}$ for all i.

Given a properly shuffled deck

$$(b_i)_{0\leq i\leq M} = b_0, b_1, b_2, \ldots, b_M,$$

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.

To shuffle the deck so that only the shuffler knows the permutation π, the shuffler first encrypts the deck as $(a_i^x)_{0\leq i\leq M}$ with a secret x. The encrypted deck $(a_i^x)_{0\leq i\leq M}$ can be considered as a "face down" representation of the deck. Then the shuffler will mix the cards up, so that the shuffled deck becomes

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

where π is a permutation. Clearly, the resulting deck is properly shuffled. The shuffler can use Protocol 4 to convince other players that the result of his shuffle is proper without revealing the permutation π.

Players can cooperatively shuffle a deck by using the procedure mentioned above in turn. Protocol 3 gives a detailed description of the Shuffle protocol.

Protocol 3: Shuffle

(i) Let $B_0 = (b_{0,i})$, where $b_{0,i} = a_i$.

(ii) For $j = 1, \ldots, N$, one by one, Player j does the following:

- (a) randomly choose a secret integer $0 < x_j < p$,
- (b) randomly choose a permutation π_j of $(0, 1, 2, \ldots, M)$ such that $\pi_j(0) = 0$,
- (c) compute $B_j = (b_{j,i})$, where $b_{j,i} = b_{j-1,\pi(i)}^{x_j}$,
- (d) broadcast B_j to other players,
- (e) use Protocol 4 to convince other players that B_j is properly shuffled from B_{j-1}.

(iii) $B = B_N = (b_{N,i})_{1 \le i \le M}$ is the shuffled deck.

Protocol 4 (Shuffle Verification) is a zero-knowledge proof and if Player j does not shuffle properly, then the probability of other players accepting Player j's shuffle is at most $2^{-K} + \epsilon$, where ϵ is negligible. Barnett & Smart's and Castellà-Roca's protocol also use the same scheme to verify the shuffle.

2.4 Card Drawing and Opening

Fix an arbitrary efficient auxiliary input zero-knowledge argument of equality of discrete logarithms (see [14] for example). Player j_0 can use Protocol 5 to draw a card from shuffled deck B.

After Player j_0 has drawn a Card i, he can use Protocol 6 to reveal the card to the other players.

2.5 Re-shuffle and partial shuffle

Because of the arithmetic properties of exponentiation and multiplication, the shuffled deck can be reshuffled by using Protocol 3 with a different set of private keys. Suppose a deck is first shuffled with private keys $(x_j)_{1 \le j \le N}$. Then players can reshuffle the deck by using the shuffled deck as the input B_0 of Protocol 3 with a different set of private keys $(x'_j)_{1 \le j \le N}$. The resulting deck can be considered as a deck shuffled with private keys $(x_j x'_j)_{1 \le j \le N}$. A card can be opened and drawn using Protocol 6 and Protocol 5 with private keys $(x_j x'_j)_{1 \le j \le N}$.

Players can also partially shuffle a deck, that is, only a certain part of the deck is being shuffled and the remainder of the deck is left undisturbed. The partial

Protocol 4: Shuffle Verification

(i) Player j randomly chooses integers $0 < y_1, y_2, \ldots, y_K < p$.

(ii) Player j randomly chooses permutations $\pi'_1, \pi'_2, \ldots, \pi'_K$ of $(0, 1, 2, 3, \ldots, M)$.

(iii) Player j computes $C_k = (c_{k,i})_{0 \le i \le M}$, where $c_{k,i} = b^{y_k}_{j,\pi'_k(i)}$ for $k = 1, 2, \ldots, K$.

(iv) For each $k = 1, 2, \ldots, K$:

- (a) Player j broadcasts C_k to other players.
- (b) Other players cooperatively generate a bit e_k by running Protocol 2 and taking the parity bit of the result.
- (c) Send e_k to Player j.
- (d) If $e_k = 0$, Player j broadcasts y_k, π'_k and every player computes $d_{k,i} = b^{y_k}_{j,\pi'_k(i)}$ for every i.
- (e) If $e_k = 1$, Player j broadcasts $x_k y_k$, $\pi_j \pi'_k$ and every player computes $d_{k,i} = b^{x_k y_k}_{j-1,\pi_j \pi'_k(i)}$ for every i.
- (f) If $d_{k,i} \neq c_{k,i}$ for any i, then Player j does not pass the verification.

(v) Player j passes the shuffle verification.

Protocol 5: Card Drawing

(i) Player j_0 picks a $c_0 \in B$.

(ii) For $j = 1, 2, \ldots, N$, one by one, Player j does the following:

- (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 ZKA to convince other players that $\log_{c_j} c_{j-1} = \log_{b_{j-1,0}} b_{j,0}$.

(iii) Player j_0 computes $c = c_N^{x_{j_0}^{-1}}$ and finds the i for which $a_i = c$.

(iv) Card i is the card Player j_0 drew.

Protocol 6: Card Opening

Player j_0 claims that he has Card i and uses ZKA to prove that $\log_{a_i} c_N = \log_{b_{j_0-1,0}} b_{j_0}$.

shuffle procedure is the same as Protocol 3, except that the possible choices of random permutations $(\pi_j)_{1\leq j\leq N}$ are different because only part of the deck needs to be shuffled. Let $(c_i)_{0\leq i\leq M}$ be a shuffled deck, $S \subseteq \{1, 2, \ldots, M\}$. If we want to reshuffle the part of the deck that contains only the i-th card for all $i \in S$, then we should restrict our choice of random permutations π_j to satisfy $\pi_j(i) = i$ for all $i \notin S$. The same restriction applies to $(\pi'_k)_{1\leq k\leq L}$ in Protocol 4.

3 Security analysis

3.1 Secure mental poker

Let us first consider the ideal model of a card game.

Definition 3.1. A *virtual deck of M cards* is a permutation π of $\{1, 2, \ldots, M\}$. If $\pi(i) = m$, then we say that the i-th card in the virtual deck is Card m.

Fix an arbitrary card game rule. An *ideal game* is a card game played by N players with a virtual deck such that:

(i) The virtual deck is held by a trusted third party.

(ii) Players and the trusted third party can communicate with each other via private channels and a broadcast channel. These channels are reliable and safe, such that messages can not be altered and senders can be identified.

(iii) The third party can fully or partially shuffle the virtual deck by replacing the permutation π with $\pi \circ \pi'$ when necessary, where π' is a uniformly random permutation of a subset of $\{1, 2, \ldots, M\}$.

(iv) The trusted third party can reveal $\pi(i)$ to some or all players for some i when necessary.

(v) Each player is allowed to *forfeit* at any time in an ideal game. The game terminates when a player forfeits.

The goal of a mental poker protocol is to allow players to play a card game that resembles an ideal game over a network without the need of a third party. The role of the trusted third party is replaced by a multiparty protocol. The network is assumed to be safe and reliable. Players can broadcast messages to all players and can send private messages to any player via the network. When a safe and reliable network is not available, players should encrypt, sign and timestamp their messages to establish practically safe and reliable communication over the network.

However, it is an important note that secure mental poker protocols cannot entirely prevent coalition. Since the ideal model allows players to communicate with each other secretly, players can easily form coalitions. Blocking or restricting private and secret channels between players is beyond the scope of this paper. What a mental poker protocol can do is minimize the effect of coalition (see [17]), i.e., so that is no worse than the coalition in an ideal game.

We use the terminology "hand history" to denote the complete transcript of the ideal game corresponding to the mental game. The transcript of a mental game consists of two parts: the card game part and the cryptography part. Hand history can be considered as the card game part of the transcript of a mental game. The event of a player being caught cheating in a mental game is identify to the event of the player forfeiting in the ideal game.

To prove the security of our protocol, we shall show that for any given strategy in a mental game, a strategy for the ideal game can be efficiently constructed such that the hand history for the mental game and the ideal game are indistinguishable. A formal definition is given below.

Definition 3.2. Let α be a real value function. Two distribution ensembles $\{D_n\}$ and $\{E_n\}$ are said to be *α-indistinguishable* if

$$\left| \Pr_{x \leftarrow D_n} (T(x) = 1) - \Pr_{x \leftarrow E_n} (T(x) = 1) \right| < \alpha$$

for any polynomial time machine T for large enough n.

Let α be a real value function of the security parameter n. Fix a subset Z of players for a mental game. Assume that all other players are honest and they play the mental game using the same card game strategy as when playing an ideal game. A mental poker protocol is said to be *α-secure against adversary Z* if for any polynomial time mental game strategy S of Z, we can efficiently find an expected polynomial time ideal game strategy S' of Z, so that the game history of the mental game and the ideal game are α-indistinguishable.

A mental game is said to be *α-secure* if it is secure against any structure of adversary as long as there is at least one honest player. If a mental game is n^{-k}-secure for every $k \in \mathbb{N}$, then the mental game is said to be *secure*.

Intuitively speaking, Definition 3.2 states that whatever the malicious players can do in a mental game, they can also do in an ideal game. The transcript of a mental game contains more information than the hand history. However, if the mental game is secure, the malicious players cannot use the additional information efficiently to improve their game, analyze an opponent's strategy, or make any kind of significant changes. In particular, Definition 3.2 easily meets all of Crépeau's requirements for secure mental poker ([17]). For instance, one of Crépeau's requirements is to "minimize the effect of coalitions". As stated previously, there is no way to prevent coalitions entirely, but secure mental poker protocols are able to minimize the effect of coalitions. Definition 3.2 implies this requirement in the sense that coalitions in a secure mental game can take no more advantage than they can in an ideal game.

Remark. Please note that although both S and S' can be considered as feasible strategies, they are feasible in different notion of efficiency. Ideally, we would like the ideal game strategy S' to be a strict polynomial time strategy, but our construction of S' requires running the simulator of ZKA. As far as we know, all practical ZKA have expected polynomial time simulators. This is because constant round zero-knowledge protocols do not have polynomial time blackbox simulators and extractors ([2]). If a ZKA with strict polynomial time simulator is used instead, then the strategy S' constructed in our proof is strict polynomial time. However, known efficient ZKA (like [14,21]) are all constant round protocols with blackbox simulators. Since efficiency is one of our main concerns, we shall allow constant round ZKA with blackbox simulators to be used.

Alternatively, it is also possible to allow both S and S' in a wider class of efficient machine, see [23,27] for more detail.

3.2 Random element generator

Theorem 3.3. *There is an expected polynomial time simulator of Protocol* 2 *and a uniform random variable r such that:*

(i) *The transcript generated by the simulator is indistinguishable from the transcript generated by Protocol* 2.

(ii) *r is independent to $\{g_j, g_j', h_j\}_{j \leq N}$.*

(iii) *Let h be the output of the simulator. Then* $\Pr(h$ *is generated and* $r \neq h)$ *is negligible.*

Proof. Let us consider Protocol 7 as simulator of Protocol 2.

In Protocol 7, h_1' is determined at step (i) because there is only one possible h_1' that satisfies

$$\log_{g_1} g_1' = \log_{h_1} h_1'$$

and only this h_1' can be accepted by ZKA (except for negligible probability). Similarly, h_3' is determined at step (iii). Let $r_j = h_j^{s_j}$ where $s_j = \log_{g_j} g_j'$ if g_j, g_j', h_j

Protocol 7: Generate a random element h

(i) The adversary broadcasts g_1, g_1', h_1.

(ii) Player 2 broadcasts random g_2, g_2', h_2.

(iii) The adversary broadcasts g_3, g_3', h_3.

(iv) The adversary broadcasts h_3' and uses ZKA to prove that $\log_{g_3} g_3' = \log_{h_3} h_3'$.

(v) Player 2 broadcasts a random h_2' and runs the simulator of ZKA on g_2, g_2', h_2, h_2' and generates the transcript.

(vi) The adversary broadcasts h_1' and uses ZKA to prove that $\log_{g_1} g_1' = \log_{h_1} h_1'$.

(vii) Return the result $h = h_1' h_2' h_3'$.

At step (i), (iii), (iv), (vi), the adversary should act in the same way as she would in Protocol 2.

are generated and $r_j = 1$ otherwise. Let $r = r_1 h_2' r_3$; r is a uniform random variable independent to g_j, g_j', h_j and the probability that h is generated and $h \neq r$ is negligible.

However, since the adversary may fail to pass ZKA at step (iv) and (vi) or refuse to execute step (i) or (iii), there is some probability that h is not generated and the adversary forfeits the game. Therefore, for any polynomial time machine T, we have

$$\begin{aligned}\Pr(T(s', h) = 1) &= \Pr(T(s', h_1' h_2' h_3') = 1 \wedge h \text{ is generated}) \\ &\leq \Pr(T(s', r) = 1) + \epsilon,\end{aligned}$$

where ϵ is negligible. The transcript of Protocol 7 and the transcript of Protocol 2 are indistinguishable, therefore, the element h generated by Protocol 2 also satisfied that

$$\Pr(T(h) = 1) \leq \Pr(T(r) = 1) + \epsilon$$

for every polynomial time machine T, uniformly random r, and negligible ϵ. □

By Theorem 3.3, h can be viewed as a random variable indistinguishable from a uniform random variable, except that there is some probability that adversary may forfeits before the element is generated.

3.3 Security proof

Let $n = \log_2 |G|$ be the security of the mental game, where G is the cyclic group used in the mental poker protocol.

Theorem 3.4. *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). Then our mental poker protocol (Protocol 1, 3, 5, 6) is secure.*

Proof. For the sake of simplicity, assume that there are three players in the card game and Player 2 is the only honest player. Since coalition is allowed, we can assume that Player 1 and Player 3 are both controlled by the same adversary. The proof of the general case is essentially the same.

In order to describe our proof, we shall introduce a sequence of games so that we are able to present an otherwise complicate security proof in a simplistic manner (see [31] for more information). These games are played by the honest player and an adversary. Game 0, is the mental game and last game, Game 8, is essentially the ideal game.

Fix a polynomial time strategy S_0 for the adversary for Game 0. We shall show that there is a correspondence strategy S_k for Game k such that the hand history of Game k is indistinguishable from that of Game $k-1$.

Game 0. The honest player and the adversary play the card game using our mental poker protocol. If cheating is detected, then the adversary forfeit the game.

Game 1. Game 1 and Game 0 are the same, except that whenever Player 1 or Player 3 pass Shuffle Verification (Protocol 4), we attempt to extract x_1 and x_3 from the adversary using Protocol 8. In the case that x_1 or x_3 cannot be extracted, then Player 2 forfeits the game.

Note that if x_j can be extracted, then B_j is properly shuffled, but the converse may not be true.

Let A_1 be the event that we fail to extract x_1 or x_3 and hence Player 2 forfeits Game 1. Game 1 and Game 0 are otherwise the same unless A_1 occurs. We shall compute $\Pr(A_1)$.

By Theorem 3.3, we have $\Pr(e_k = 1) < \frac{1}{2} + \epsilon$ and $\Pr(e_k = 0) < \frac{1}{2} + \epsilon$ for some negligible ϵ. Let j be either 1 or 3. Let p_k be the probability that Player j is going to pass the Shuffle Verification when the k-th round of Shuffle Verification is just about to begin. We want to show that the probability of Player j passing the k-th round of verification and x_j cannot be extracted is no more than $\frac{1}{2} + \epsilon$. If $p_k < \frac{1}{2} + \epsilon$, then the probability of Player j passing the k-th round of verification is no more than $\frac{1}{2} + \epsilon$. Assume $p_k \geq \frac{1}{2} + \epsilon$. The probability that Player j passes

Protocol 8: Extract x_j

Let (e_k) be the bits generated at Step (iv) (b) of Shuffle Verification. For each $k = 1, 2, \ldots, K$,

(i) Rewind the adversary back to Step (iv) (b) for k of Shuffle Verification.

(ii) Run Steps (iv) (b)–(f) of Shuffle Verification and let e'_k be the random bit generated at Step (iv) (b).

(iii) Run Steps (i) and (ii) for n times.

(iv) Steps (iv) (b)–(f) of Shuffle Verification have been executed for n times. If in one of these executions, e'_k is generated and $e'_k \neq e_k$, then both y_k and $x_j y_k$ are known and we can compute $x_j = x_j y_k / y_k$.

the k-th round of verification and $e'_k \neq e_k$ for all of n executions of Step (i), (ii) in Protocol 8 is at most

$$p_k\left(1 - \left(p_k - \frac{1}{2} - \epsilon\right)\right)^n.$$

By using calculus method for finding extreme values, we have

$$p_k\left(1 - \left(p_k - \frac{1}{2} - \epsilon\right)\right)^n < \frac{1}{2} + \epsilon$$

when $1 \geq p_k > \frac{1}{2} + \epsilon$ and n is large enough. So the probability that Player j passes all K rounds of the Shuffle Verification and x_j cannot be extracted is less than $(\frac{1}{2} + \epsilon)^K$.

Let TimeBound be the polynomial time bound of Game 0. So Protocol 8 can only be executed for less than TimeBound times in Game 1. We have

$$\Pr(A_1) \leq \text{TimeBound} \cdot \left(\frac{1}{2} + \epsilon\right)^K,$$

which is negligible.

Therefore, except for a negligible probability, hand histories of Game 0 and Game 1 are the same.

Game 2. Same as Game 1, but we use the knowledge of x_1, x_3 to detect cheating when drawing or opening cards. That is, in addition to the zero-knowledge argument at Step (ii) (d) in Card Drawing (Protocol 5) and Card Opening (Protocol 6), we also check whether $\log_{c_j} c_{j-1} = x_j$ directly for $j = 1, 3$ if $j \neq j_0$. If $\log_{c_j} c_{j-1} \neq x_j$, then the adversary forfeits the game.

Let A_2 be the event that the zero-knowledge argument used in Card Drawing (Protocol 5) and Card Opening (Protocol 6) fails to detect $\log_{c_j} c_{j-1} \neq x_j$. Game 1 and Game 2 only differ when event A_2 occurs. By the definition of soundness, $\Pr(A_2)$ is negligible.

Game 3. Same as Game 2, except that Player 2 uses a different method to decrypt cards at Step (ii) (a) and (iii) of Card Drawing (Protocol 5) and in Card Opening (Protocol 6). Using the notation of Protocol 5, suppose $c_0 = b_{N,\pi_3\pi_2\pi_1(i)}$, Player 2 can use the knowledge of x_1, x_3 (and the transcript of shuffle) to recover π_1, π_3 and i efficiently. If $j_0 \neq 2$, instead of computing

$$c_2 = c_1^{x_2^{-1}},$$

Player 2 computes

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

at Step (ii) (a) of Protocol 5.

If $j_0 = 2$, instead of computing

$$c = c_3^{x_2^{-1}},$$

Player 2 computes $c = a_i$ at Step (iii) of Protocol 5.

Similarly, when opening a card, Player 2 use x_1, x_3 to recover π_1, π_3 and hence i without using x_2.

Note that this only a conceptual change and does not alter the values of i, c_2 and c, and is merely an alternative way of computing these values without using x_2. The hand histories of Game 2 and Game 3 are exactly the same.

Game 4. Same as Game 3, except that

(i) At Step (ii) (d) of Card Drawing (Protocol 5), players do not execute ZKA to prove

$$\log_{c_2} c_1 = \log_{b_{1,0}} b_{2,0}.$$

Instead, we use the simulator of ZKA to generate a transcript that is indistinguishable from the real transcript.

(ii) When Player 2 opens a card, players runs ZKA to prove

$$\log_{a_i} c_N = \log_{b_{j_0-1,0}} b_{j_0}.$$

Instead, we use the simulator of ZKA to generate a transcript that is indistinguishable from the real transcript.

(iii) Protocol 2 is replaced by Protocol 7.

(iv) Players do not use Shuffle Verification(Protocol 4) to verify Player 2's shuffle at Step (ii) (e) of Shuffle (Protocol 3). Instead, we use the following expected polynomial time simulator to generate a transcript:

Protocol 9: Simulator for Shuffle Verification

```
For each $k = 1, 2, \dots, K$,
  (a) $r_k \leftarrow 1$.
  (b) Choose a random bit $e'$, a random $0 < y < n$ and a random permutation $\pi'$.
  (c) If $e' = 0$, compute $(c_i)_{0 \le i \le M}$, where $c_i = b^y_{j,\pi'(i)}$.
  (d) If $e' = 1$, compute $(c_i)_{0 \le i \le M}$, where $c_i = b^y_{j-1,\pi'(i)}$.
  (e) Use the adversary as a blackbox. Rewind and run Step (iv) (b) of Shuffle Verification to generate a random bit $e$ by treating $(c_i)_{0 \le i \le M}$ as $C_k$.
  (f) $r_k \leftarrow r_k + 1$.
  (g) If $r_k > n$, then stop the simulation and Player 2 forfeits the game.
  (h) If $e$ is not generated and $e' = 0$, go to Step (a).
  (i) If $e$ is generated and $e \neq e'$, go to Step (a).
  (j) Write $(c_i)_{0 \le i \le M}$, the transcript generated in Step (ii) (d), $y$, $\pi'$ into the transcript.
  (k) If $e$ is not generated, then the adversary shall forfeit.
```

Therefore e' is independent to (c_i) and the distribution of (c_i) is identical to the real one. Thus, if we did not truncate the simulation (i.e. skip Step (g)), then the distribution of the transcript generated by Protocol 9 should be identical to the distribution of Protocol 4. The probability of the simulation being truncated is no more than $K2^{-n}$, which is negligible. So the transcript generated by the simulator is indistinguishable from the transcript generated by Shuffle Verification.

Since these changes generated indistinguishable transcripts, the transcripts of Game 3 and Game 4 are indistinguishable.

Game 5. Same as Game 4 except that we use a different method to generate

$$B_2 = (b_{2,i})_{0 \le i \le M}$$

in Shuffle (Step (ii) (a)–(c) of Protocol 3). Player 2 does not execute Step (ii)(a)–(c) of Protocol 3. Instead, recall that $(a_i)_{0 \le i \le M}$ is the face up deck generated in Deck

Preparation (Protocol 1). We generate a random x and let $f_i = a_i^x$. Then use the knowledge of x_1 to recover π_1 and compute

$$b_{2,i} = f_{\pi_2 \circ \pi_1(i)},$$

where π_2 is a random permutation such that $\pi_2(0) = 0$.

The distributions of B_2 in Game 4 and Game 5 are identical. The only change made was the way we generated B_2 without using x_2, due to the fact that x_2 is not needed elsewhere in the game anymore. Therefore, the transcripts of Game 5 and Game 4 are the same.

Game 6. Same as Game 5, except that we generate uniformly random f_i and does not generate x.

By Theorem 3.3, the face up deck $(a_i)_{0 \le i \le M}$ can be viewed as genuine random variables (except for some negligible differences), but there may be some chance that the adversary may forfeit so that the face up deck cannot be actually generated. Therefore, the DDH assumption implies that

$$(a_0, a_1, \ldots, a_M, a_0^x, a_1^x, \ldots, a_M^x)$$

and

$$(a_0, a_1, \ldots, a_M, f_0, f_1, \ldots, f_M)$$

are indistinguishable.

Thus, the transcripts of Game 5 and Game 6 are indistinguishable.

Game 7. Game 7 is the same as Game 6, except that instead of letting

$$b_{2,i} = f_{\pi_2(i)}$$

when Player 2 shuffles the deck, we compute $b_{2,i} = f_i$. Player 2 still generates π_2, which is used for card drawing (see the description of Game 3). This is only a conceptual change to emphasize that π_2 is information theoretically secure. The transcripts of Game 6 and Game 7 have the same distribution.

Before we describe Game 8, let us summarize the changes that have been made so far.

The adversary and the honest player uses following protocols to play Game 7 (the adversary uses the strategy of Game 0 to play corresponding fragment of the protocol).

Deck Preparation: Same as Protocol 1 but Protocol 2 is replaced by the simulator (Protocol 7).

After the shuffle, since x_1 and x_3 are known, π_1 and π_3 can be easily recovered. Let $\pi = \pi_3 \pi_2 \pi_1$. Use Protocol 11 to draw cards.

Protocol 10: Game 7 Shuffle

(i) Let $B_0 = (b_{0,i})$, where $b_{0,i} = a_i$.

(ii) Run Step (ii) of Shuffle (Protocol 3) to generate B_1.

(iii) Rewind the game to extract x_1 from the adversary (Protocol 8).

(iv) Generate a random $B_2 = (b_{2,i})_{0 \le i \le M}$.

(v) Simulate the Shuffle Verification and generate an indistinguishable transcript (Protocol 9).

(vi) Run Step (ii) of Shuffle (Protocol 3) to generate B_3 from B_2.

(vii) Rewind the game to extract x_3 from the adversary (Protocol 8).

(viii) Let $B = (b_{N,i})_{1 \le i \le M}$ be the shuffled deck.

Moreover, Player 2 privately generates a permutation π_2 such that $\pi_2(0) = 0$.

Protocol 11: Game 7 Drawing

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

(i) Player 2 computes $i = \pi^{-1}(i')$

(ii) If $j_0 \neq 2$ (when the adversary draws the card):

- (a) Run Steps (ii) (a)–(d) of Card Drawing (Protocol 5) to generate c_1.
- (b) Player 2 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 of ZKA to generate a transcript.
- (c) Run Steps (ii) (a)–(d) of Card Drawing (Protocol 5) to generate c_3.
- (d) The adversary can run Steps (iii) and (iv) of Card Drawing (Protocol 5) to find i.

(iii) If $j_0 = 2$ (when Player 2 draws the card):

- (a) Run Step (ii) of Card Drawing (Protocol 5).
- (b) The card drawn by Player 2 is Card i.

Protocol 12: Game 7 Opening

The adversary uses Card Opening (Protocol 6) to open cards. Player 2 opens the card by showing i and then uses the simulator of ZKA to generate a transcript.

Game 8. Game 8 is the ideal game. The adversary, the honest player and a trusted third party play the card game using the following protocol.

- Shuffle: The trusted party randomly chooses a permutation π.
- Drawing: Player j_0 picks a number $i_0 \leq M$. The trusted party sends $\pi^{-1}(i_0)$ to Player j_0.
- Opening: When a player wish to open a face down card i_0, the trusted third party announces $\pi^{-1}(i_0)$.

The honest player uses the same card game strategy for Game 0 to play Game 8. The adversary uses the partial information of π that the trusted third party sent to her and the real hand history of Game 8 to simulate a corresponding Game 7. Then she copies her plays in the simulation to play Game 8. If any player in the simulated game forfeits, then the adversary forfeits Game 8.

This is only a conceptual change. The algorithm is the same, and only some parts of the protocol are assigned to and executed by different participants. The shuffle is completely controlled by the trusted third party. The honest player only needs to be concerned with his part of the pure card game. The rest of the work is executed and simulated by the adversary. Secret keys and random bits like x_j, e_j etc, are no longer needed by the ideal card game. These variables are only used in simulation in order to decide the card game strategy of the adversary.

Therefore, the hand histories of Game 8 and Game 7 are the same. □

The parameter K is usually viewed as a constant independent to the security parameter n in the context of mental poker. We have the following corollaries.

Corollary 3.5. *Let $K = K_0$ be a positive constant independent to n. Then our protocol is secure against passive adversaries.*

Proof. We shall prove that our protocol is secure if the adversary always responds to the Protocol 4 properly, that is,

$$\Pr(\text{the adversary being caught cheating in Protocol 4}) = 0.$$

The hand histories of following games are indistinguishable.

Game 1. Players play mental poker with security parameter n and $K = K_0$.

Game 2. Players play mental poker with security parameter n and $K = K_0 + n$. The adversary uses the same strategy as Game 1 to play Game 2 (ignoring the last n rounds of Shuffle Verifications).

Since the adversary always passes the last n rounds of Shuffle Verifications, the distribution of the hand history is exactly the same as Game 1.

By Theorem 3.4, Game 2 is secure, so Game 1 is also secure. □

Corollary 3.6. *Let K be a positive constant independent to n. Then our protocol is $(2^{-K} + \epsilon)$-secure for some negligible function ϵ of n.*

Proof. We only need to prove that the protocol is $(\frac{1}{2} + 2n^{-q})^K$-secure for every positive q.

Fix a positive q and consider the following two games.

Game 1. Players play mental poker with security parameter n and $K = K_0$.

Game 2. Players play mental poker with security parameter n and

$$K = K_0 + nK_0.$$

Except for the last nK_0 rounds of Shuffle Verification, the adversary uses the same strategy to play Game 2.

For the last nK_0 rounds of the verification, when being the verifier, the adversary executes the protocol honestly. When being the prover, if he passes the first K_0 rounds of Shuffle Verification, then the adversary uses Protocol 13 to extract his own x_j. If x_j is successfully extracted, then he runs the last nK_0 rounds of Shuffle Verification honestly. Otherwise, he forfeits.

Game 1 and Game 2 are the same except for the event that the adversary passes the first K_0 rounds of the Shuffle Verification but cannot extract his own x_k.

Suppose the adversary is going to use Shuffle Verification to prove his shuffle in Game 1. Let p be the probability that he passes all K_0 rounds of verification. Suppose

$$p \geq \left(\frac{1}{2} + n^{-q}\right)^{K_0}.$$

Then $p_k \geq \frac{1}{2} + n^{-q}$ for some k, where p_k is the probability that the adversary passes k-th round of the verification. By Theorem 3.3, the probability that x_j cannot be extracted is

$$\left(1 - \left(p_k - \frac{1}{2} - \epsilon'\right)\right)^{n^{2q}}$$

Protocol 13: Extract x_j

Let (e_k) be the bits generated at Step (iv) (b) of Shuffle Verification.
For each $k = 1, 2, \ldots, K$,

(i) Rewind the adversary back to Step (iv) (b) for k of Shuffle Verification.

(ii) Run Steps (iv) (b)–(f) of Shuffle Verification and let e'_k be the random bit generated at Step (iv) (b).

(iii) Run Steps (i) and (ii) n^{2q} times.

(iv) Steps (iv) (b)–(f) of Shuffle Verification have been executed for n^{2q} times. If in one of these executions, e'_k is generated and $e'_k \neq e_k$, then both y_k and $x_j y_k$ are known and we can compute $x_j = x_j y_k / y_k$.

(Note that this protocol is the same as Protocol 8, except that Steps (i), (ii) loop n^{2q} times.)

for some negligible ϵ'. For large enough n,

$$\left(1 - \left(p_k - \frac{1}{2} - \epsilon'\right)\right)^n > \left(1 - \frac{1}{2}n^{-q}\right)^{n^{2q}},$$

which is negligible.

If $p < (\frac{1}{2} + n^{-q})^{K_0}$, then for probability $(\frac{1}{2} + n^{-q})^{K_0}$, the adversary forfeits and the game terminates.

Therefore the probability that the hand history of Game 1 and Game 2 are different is less than $(\frac{1}{2} + n^{-q})^{K_0}$ plus some negligible function. Since Game 2 is secure, it follows that Game 1 is $(\frac{1}{2} + 2n^{-q})^{K_0}$-secure. □

4 Efficiency analysis

4.1 Computational cost

In this section, we compare the computational cost (time) of our protocol to similar protocols, namely, the protocols of Castellà-Roca ([10]) and Barnett & Smart ([5]).

All these protocols are discrete logarithm based. The most time consuming operations in these protocols are exponentiation and ZKA. So that we can compare our results with the work of [10], the computational cost of multiplication was also considered. The computational cost of other operations was assumed to be much lower and can be ignored. Denote z, e, m as 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 Shuffle is compared in Section 4.2, and the cost of Card Opening and Drawing is compared in Section 4.2.

To give some idea of empirical execution time and how practical these protocols might be, we have also made some estimations of execution time in Section 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). Table 1 compares the computational cost of Shuffle. For the calculation of the computational cost of Castellà-Roca's and Barnett & Smart's protocols we refer to [10].

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

Table 1. Computational cost for Shuffle.

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

The main weakness of our protocol related to partially reshuffling a deck. For Castellà-Roca's and Barnett & Smart's protocol, the time cost to shuffle a subset containing M' cards is $\frac{M'}{M}$ times the cost of shuffling a full deck. In contrast, the time cost of a partially shuffle is the same as shuffling a full deck using our protocol. So for card games like old maid, I doubt and slapjack, which require many partial shuffles, our protocol is more likely to be slower. There are some apparent ways to speed up the partial reshuffle when implement our protocol. Nevertheless, it is safe to say that our protocol is more suitable for card games like bridge, most poker games, mahjong, hearts, or black jack, which do not requires many partial reshuffles,

4.3 Card Opening and Drawing

The cost of Card Opening and Drawing are much lower compared to Shuffle costs. Table 2 compares the computational cost of Card Opening and Card Drawing.

	Card Opening	Card Drawing
Our Protocols	z	$(N-1)z + Ne$
Castellà-Roca	$z + (N-1)e$	$(N-1)z + (N + \frac{M}{2})e$
Barnett & Smart	$z + N(N-1)m$	$(N-1)z + Ne + Nm$

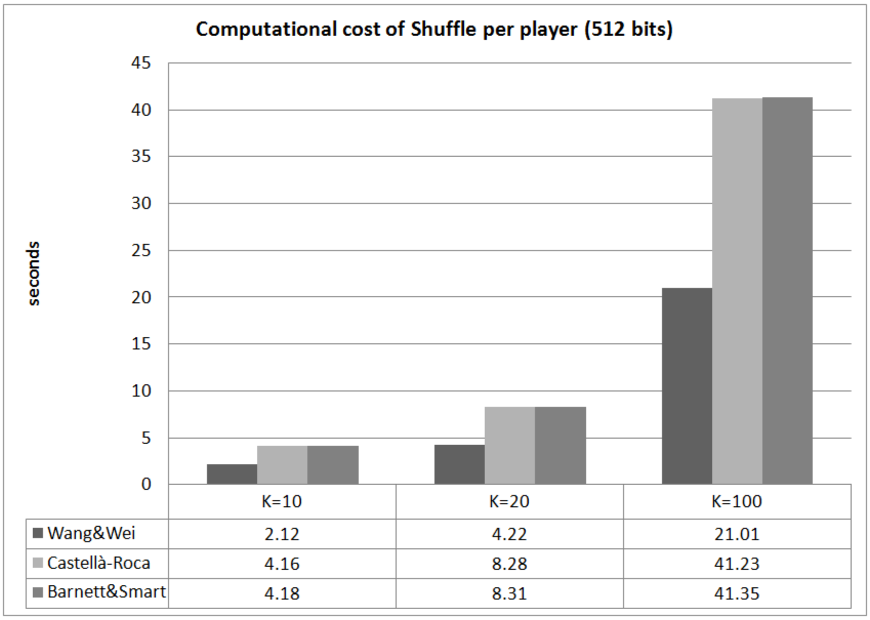
Table 2. Total computational cost for drawing and opening.

Our protocol is slightly faster than the other protocols. If the computation cost m of multiplication is ignored, then the computational cost of our protocol and Barnett & Smart's protocol 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.

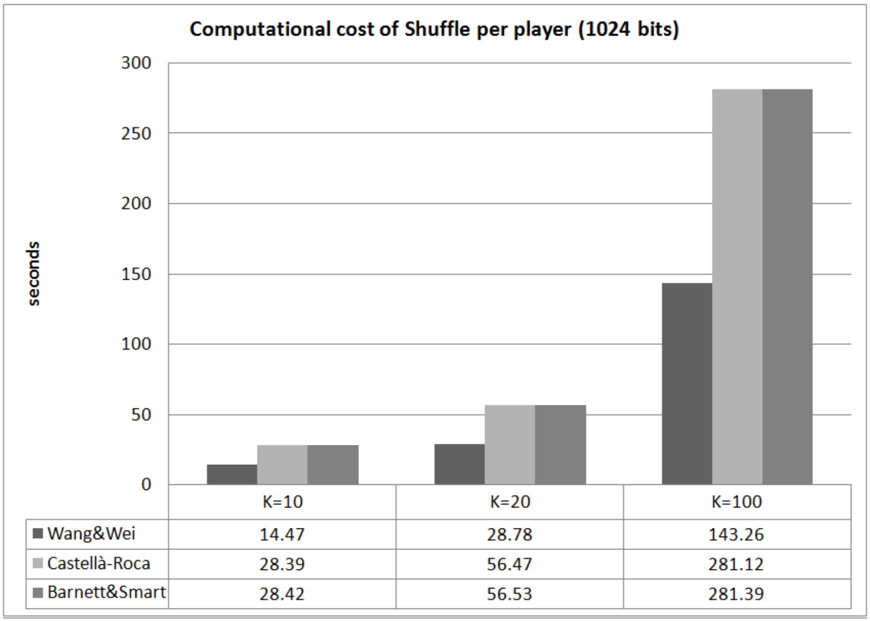
On an AMD X2 3800+ 2Ghz, which is fairly mediocre by today's PC hardware standards, e and m are about 4.4×10^{-4} and 1.3×10^{-6} seconds for 512 bits integers (when using both cores). The computational cost based on this setting can be found in Figure 1.



	K=10	K=20	K=100
■ Wang&Wei	2.12	4.22	21.01
■ Castellà-Roca	4.16	8.28	41.23
■ Barnett&Smart	4.18	8.31	41.35

Figure 1. Computational cost (seconds) for each player (512 bits).

On the same machine, e and m is about 3×10^{-3} and 3×10^{-6} seconds for 1024 bits integers. The computational cost based on this setting can be found in Figure 2.



	K=10	K=20	K=100
■ Wang&Wei	14.47	28.78	143.26
■ Castellà-Roca	28.39	56.47	281.12
■ Barnett&Smart	28.42	56.53	281.39

Figure 2. Computational cost (seconds) for each player (1024 bits).

The difference between the Castellà-Roca and Barnett & Smart protocols are less than 1% and our protocol is roughly twice as fast.

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

When using 1024 bits primes and

$$K = 100,$$

all protocols are too slow.

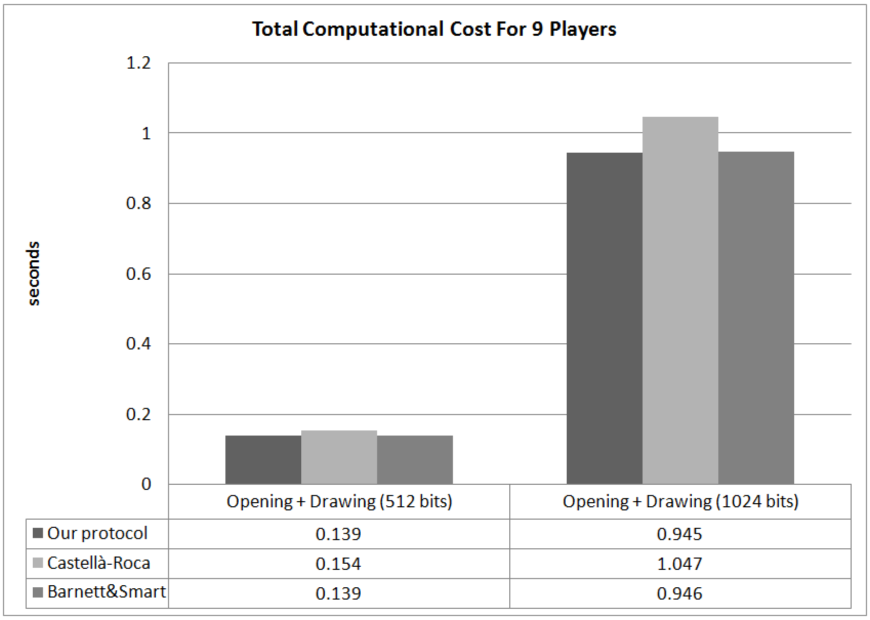
To estimate the execution time of Card Opening and Card Drawing, assume using the Chaum–Pedersen's protocol (see [13]) as the zero-knowledge argument. Thus,

$$z = (2N - 1)(2e + m).$$

Table 3 and Figure 3 show the total computational cost when the Chaum–Pedersen protocol is used.

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

Table 3. Total computational cost using the Chaum–Pedersen protocol.



	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

Figure 3. Total computational cost (seconds) using the Chaum–Pedersen protocol.

The computational cost of our protocol and Barnett & Smart's protocol are roughly the same, while Castellà-Roca's protocol is approximately 10% slower. The speed of Card Drawing and Opening of these protocols seem to be acceptable for practical use.

Theoretically, the 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. In fact, it is used in [10] for computing computational cost. The findings in Table 3 and Figure 3 are comparable with previous research results. Zero-knowledge arguments proposed in [14,21] are secure against general verifiers and have a very small constant factor of computational overhead compared to the Chaum–Pedersen's protocol.

5 Conclusion

Our protocol was proved to be secure in Section 3 under the DDH assumption. Theorem 3.4 roughly states that cheating will be detected and other than that, the mental game is indistinguishable from the ideal game.

However, please note that our security proof is based on some assumptions. For example, we assume the cheater loses if he is caught cheating. To make this assumption practical, the penalty of cheating should be high enough, so that cheating is not an option. Moreover, even if a ZKA with a strict polynomial time simulator is used, the execution time of Game 8 is longer than Game 0. We implicitly assume that the difference is insignificant.

Our protocol has been shown to be quite 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. However, now our protocol is much faster for many card games, and requires only half the computing power to achieve the same performance level. We did not discuss the communication costs of our protocol in this paper. However, it can be easily verified that the communication cost of our protocol is also lower; roughly half the cost compared to other protocols, with the exception of the partial shuffle. Based on the results of this paper, we have also developed a modified version of this protocol, which has much lower communication costs, at the expense of small computation overhead (see [32]).

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

Bibliography

[1] F. Bao, R. H. Deng and H. Zhu, Variations of Diffie–Hellman Problem, in: *Information and Communications Security – ICIS 2003* Lecture Notes in Computer Science 2836, Springer-Verlag, Berlin (2003), 301–312.

[2] B. Barak and Y. Lindell, Strict polynomial-time in simulation and extraction, in: *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, ACM Press, New York (2002), 484–493.

[3] B. Barak, Y. Lindell and S. Vadhan, Lower bounds for non-black-box zero knowledge, in: *Proceedings of the 44th Annual Symposium on Foundations of Computer Science – FOCS '03*, IEEE Computer Society, Los Alamitos (2003), 384–393.

[4] I. Barany and Z. Furedi, Mental poker with three or more players, *Inf. Control* **59** (1984), 84–93.

[5] A. Barnett and N. Smart, Mental poker revisited, in: *Cryptography and Coding*, Lecture Notes in Computational Science 2898, Springer-Verlag, Berlin (2003), 370–383.

[6] D. Boneh, The decision Diffie–Hellman problem, in: *Algorithmic Number Theory*, Lecture Notes in Computer Science 1423, Springer-Verlag, Berlin (1998), 48–63.

[7] D. Boneh and M. Franklin, Efficient generation of shared RSA keys, in: *Advances in Cryptology – Crypto '97*, Lecture Notes in Computer Science 1294, Springer-Verlag, Berlin (1997), 425–439.

[8] D. Boneh and M. Franklin, Efficient generation of shared RSA Keys, *J. ACM* **48** (2001), 702–722.

[9] E. Bresson, Y. Lakhnech, L. Mazaré and B. Warinschi, A generalization of DDH with applications to protocol analysis and computational soundness, in: *Advances in Cryptology – Crypto 2007*, Lecture Notes in Computer Science 4622, Springer-Verlag, Berlin (2007), 482–499.

[10] J. Castellà-Roca, *Contributions to mental poker*, Ph.D. Thesis, Autonomous University of Barcelona, Doctoral Programme in Computer Science and Artificial Intelligence, 2005.

[11] J. Castellà-Roca, J. Domingo-Ferrer and F. Sebe, On the security of a repaired mental poker protocol, *Third International Conference on Information Technology: New Generations – ITNG '06*, IEEE Computer Society, Los Alamitos (2006), 664–668.

[12] J. Castellà-Roca, F. Sebe and J. Domingo-Ferrer, Dropout-tolerant TTP-free mental poker, in: *Trust, Privacy and Security in Digital Business*, Lecture Notes in Computer Science 3592, Springer-Verlag, Berlin (2005), 30–40.

[13] D. Chaum and T. P. Pedersen, Wallet databases with observers, in: *Advances in Cryptology – Crypto '92*, Lecture Notes in Computer Science 740, Springer-Verlag, Berlin (1993), 89–105.

[14] R. Cramer, I. Damgard and P. MacKenzie, Efficient zero-knowledge proofs of knowledge without intractability assumptions, in: *Public Key Cryptography– PKC 2000*, Lecture Notes in Computer Science 1751, Springer-Verlag, Berlin (2000), 354–372.

[15] R. Cramer and V. Shoup, A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack, in: *Advances in Cryptology – Crypto '98*, Lecture Notes in Computer Science 1462, Springer-Verlag, Berlin (1998), 13–25.

[16] R. Cramer and V. Shoup, Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack, *SIAM J. Comput.* **33** (2004), 167–226.

[17] C. Crepeau, A secure poker protocol that minimizes the effect of player coalitions, in: *Advances in Cryptology – Crypto '85*, Lecture Notes in Computer Science 218, Springer-Verlag, Berlin (1986), 73–86.

[18] C. Crepeau, A zero-knowledge poker protocol that achieves confidentiality of the players' strategy or how to achieve an electronic poker face, in: *Advances in Cryptology – Crypto '86*, Lecture Notes in Computer Science 263, Springer-Verlag, Berlin (1987), 239–247.

[19] T. El Gamal, A public key cryptosystem and a signature scheme based on discrete logarithms, in: *Advances in Cryptology – Crypto '84*, Lecture Notes in Computer Science 196, Springer-Verlag, Berlin (1985), 10–18.

[20] S. Fortune and M. Merrit, Poker protocols, in: *Advances in Crytology – Crypto '84*, Lecture Notes in Computer Science 196, Springer-Verlag, Berlin (1985), 454–464.

[21] O. Goldreich, How to construct constant-round zero-knowledge proof systems for NP, *J. Cryptology* **9** (1996), 167–189.

[22] O. Goldreich, Zero-knowledge twenty years after its invention, Cryptology ePrint Archive, Report 2002/186, `http://eprint.iacr.org/2002/186`.

[23] O. Goldreich, On expected probabilistic polynomial-time adversaries: A suggestion for restricted definitions and their benefits, in: *Theory of Cryptography – TCC 2007*, Lecture Notes in Computer Science 4392, Springer-Verlag, Berlin (2007), 174–193.

[24] O. Goldreich and Y. Oren, Definitions and properties of zero-knowledge proof systems, *J. Cryptology* **7** (1994), 1–32.

[25] S. Goldwasser and S. Micali, Probabilistic encryption & how to play mental poker keeping secret all partial information, in: *Proceedings of the 14th Annual ACM Symposium on Theory of Computing*, ACM Press, New York (1982), 365–377.

[26] P. Golle, Dealing cards in poker games, in: *Proceedings of the International Conference on Information Technology: Coding and Computing – ITCC 2005*, IEEE Computer Society, Los Alamitos (2005), 506–511.

[27] J. Katz and Y. Lindell, Handling expected polynomial-time strategies in simulation-based security proofs, in: *Theory of Cryptography – TCC 2005*, Lecture Notes in Computer Science 3378, Springer-Verlag, Berlin (2005), 128–149.

[28] K. Kurosawa, Y. Katayama and W. Ogata, Reshuffleable and laziness tolerant mental poker game, *IEICE Trans. Fundam. E* **80** (1997), 72–78.

[29] R. J. Lipton, How to cheat at mental poker, Technical report, Computer Science Department, Berkeley University, 1979.

[30] A. Shamir, R. L. Rivest and L. M. Adleman, Mental poker, Technical report LCS/TM-125, Massachusetts Institute of Technology, April 1979.

[31] V. Shoup, Sequences of games: A tool for taming complexity in security proofs, Cryptology ePrint Archive, Report 2004/332, `eprint.iacr.org/2004/332`.

[32] T. Wei, Communication efficient shuffle for mental poker protocols, *Inf. Sci.* **181** (2011), 5053–5066.

[33] Wikipedia, Online poker, `http://en.wikipedia.org/wiki/Online_poker#Insider_cheating` (as of February 3, 2012, 10:50 GMT).

[34] M. Yung, Cryptoprotocols: Subscription to a public key, the secret blocking and the multi-player mental poker game, in: *Advances in Cryptology – Crypto '84*, Lecture Notes in Computer Science 196, Springer-Verlag, Berlin (1985), 439–453.

[35] W. Zhao and V. Varadharajan, Efficient TTP-free mental poker protocols, in: *International Conference on Information Technology: Coding and Computing – ITCC 2005*, IEEE Computer Society, Los Alamitos (2005), 745–750.

[36] W. Zhao, V. Varadharajan and Y. Mu, A secure mental poker protocol over the internet, in: *ACSW Frontiers 2003. Proceedings of the First Australasian Information Security Workshop (AISW 2003)*, ACS, Adelaide (2003), 105–109.

Received September 2, 2010; revised April 18, 2012; accepted April 18, 2012.

Author information

Tzer-jen Wei, Department of Applied Mathematics, National Dong Hwa University, Shoufeng, Hualien 97401, Taiwan.
E-mail: tjw@mail.ndhu.edu.tw

Lih-Chung Wang, Department of Applied Mathematics, National Dong Hwa University, Shoufeng, Hualien 97401, Taiwan.
E-mail: lcwang@mail.ndhu.edu.tw