Cards on the Table: Secure Card Games without a Trusted Party

Bryce Crouse and Mukunth

December 2024

Link to **GitHub repository**

1 Problem Description and Motivation

The idea of "Mental Poker" was first introduced by Rivest, Shamir, and Adelman in 1981 [1]. The problem they sought to solve was whether it was possible for two potentially dishonest players to play poker remotely, without using any cards, and without a trusted third-party. While their formulation only works for the simplest poker (five-card draw), the idea has since been expanded to playing secure, no-trust card games in general.

Due to the advent of the internet, secure no-trust card games have clear modern applications. Nowadays, card games like Poker and Blackjack are played on large websites that act as the "trusted third-parties" to ensure fairness in their games. This is suboptimal in some ways because players have to trust that these sites are operating fairly and are not influencing the game outcomes (for example, by colluding with players). Additionally, as a fee for operating, these websites take a "rake", an (often large) percentage of a poker pot, or some form of fee.

Having a widely implemented protocol for secure card games would allow players to play card games totally securely over the internet, bypassing these websites and requiring no trust on the players' parts.

2 Prior work and comparison

An overview of how "Mental Poker" could be achieved was proposed first in [1]. The main takeaway from this is the layout of the algorithm that should be used for commutatively encrypting the card representations. It also proceeds to give an example of such a commutative encryption utilizing modular exponentiation for which, the security is influenced to a high degree by the choice of parameters. Nevertheless, the given layout for how the system should encrypt was widely adopted, resulting in multiple different protocols.

Bárány and Füredi proposed a Mental Poker protocol [4] in 1983 that avoided the need for a commutative cryptosystem (unlike [1]). Instead, the protocol relied on players exchanging permutations of cards during the shuffling phase. The protocol assumes that players act independently. However, if two or more players collude, they can pool their knowledge to reconstruct the complete deck order and view the hands of non-colluding players. By combining their applied permutations, the colluding players can compute the cumulative effect of their contributions to the deck's shuffle.

Crépeau's protocol in 1985 [5] was designed to reduce the impact of coalitions and ensure the confidentiality of cards. It leveraged the probabilistic encryption scheme [3] proposed by Goldwasser and Micali to encrypt cards which relies on the hardness of the quadratic residuosity problem. However, the scheme in [3] itself was computationally expensive, as encrypting a single bit required multiple operations. The protocol also requires all players to reveal their secret permutation that they used to shuffle the deck, during the end of the game, to ensure fair play. To prevent this, Crépeau also proposed an extension of [5] in 1987 in [6], which involved interactive zero-knowledge arguments to prove a player's hand is valid. However, this protocol ended up being very slow in practice, mainly due to the Goldwasser-Micali encryption schemes and the interactive zero-knowledge arguments.

In 2003, Zhao, Varadharajan, and Mu proposed a protocol [7] designed to enable secure online poker games without the need for a trusted third-party. Compared to the modular exponentiation used in [1], this protocol utilizes ElGamal encryption for faster computation. However, with an attack [8] proposed by Castellà-Roca and Domingo-Ferrer in 2004, the player could learn the card values without knowledge of the encryption keys. This attack is under the assumption that the initial card values are publicly known (eg. 1, 2, 3, ..., 52), which is usually the case in practice.

Preventing such vulnerabilities is much more accessible in modern cryptography with advancements in elliptic curves and ZK-SNARK systems. More discussion about our protocol and its security in Section 3.

3 Detailed results

3.1 Description of Protocol

Our protocol is adapted from the protocol [2] proposed by Barnett and Smart. Although our game is only for two players, the protocol actually works for an arbitrary number of players and is generalizable to other games. We'll explain the sub-protocols first before formulating how they fit into the overall protocol and discussing security.

3.1.1 Setup

The players agree beforehand on a large prime-order elliptic curve group (we used secp256k1, the same curve that Ethereum uses) and a generator g. We will denote the n players $p_1, p_2, ..., p_n$. We use the multiplicative notation for operations in an elliptic curve group.

3.1.2 Key Generation

First, each player generates a key.

```
KeyGen for player $i$:

  1. Private Key $x_i \overset{R}{\leftarrow} \mathbb{Z}_q$
  2. Public Key $H_i = g^{x_i}$
  3. Publish $H_i$
```

A group public key $H = H_1 * H_2 * ... * H_n$ (aggregate key) is calculated as the product of all n individual public keys. This will be used later for decrypting cards.

3.1.3 Encrypting cards

We represent card values by $C = (a, b)$ where $a, b \in G$ denotes the value of the card or the encrypted ciphertext of a card. Before encrypting, the 52 cards $C_1, C_2, ..., C_{52}$ are initialized $C_1 = (1, g^1)$, $C_2 = (1, g^2)$, ..., $C_52 = (1, g^{52})$ (where 1 denotes the identity element in G).

```
encrypt($C$) (by player $i$):

  1. Sample masking factor $r \overset{R}{\leftarrow} \mathbb{Z}_q$
  2. $C_a = a * g^r$
  3. $C_b = b * {H_i}^r$
  4. produce proof ${\pi_i}^{encryption}$ for a valid encryption
  5. set $C = (C_a, C_b)$
  6. output ${\pi_i}^{encryption}$
```

3.1.4 Decrypting cards

```
decrypt($C$):
```

```
1. For each player $i$,
    (a) player $i$ computes $u_i = {C_a}^{x_i}$
    (b) player $i$ produces proof ${\pi_i}^{decrypt}$ for a valid $u_i$ (Schnorr Proof with witness $x_i$)
    (c) other players verify ${\pi_i}^{decrypt}$
2. $U = \prod_i u_i$
3. output $v = C_b/U$
```

Completeness: Under our definition, a card starts out with a unique value $(1, v)$. Assume we have i players who all encrypt the cards, where the ith player used their secret key x_i and a masking factor for this card of r_i. Let $R = \prod_i r_i$.

After being encrypted by all players, $(1, v)$ is mapped to $(g^R, v * H^R)$. But notice $U = \prod_i u_i = H^R$ and thus $C_b/U = v * H^R/H^R = v$.

3.1.5 Proof-of-shuffle

We utilized Groth16 to prove shuffle, as outlined in [9]. Let $deck_{init}$, $deck_{shuffled}$, $perm_{arr}$ be the initial deck, shuffled deck, and the permutation arrays respectively.

```
generateProof($deck_{init}$, $deck_{shuffled}$, $perm_{arr}$):
    1. generate Prove($w$) $\to \pi_{shuffle}$, that the prover knows $perm_{arr}$ such that, $deck_{init}[i] = deck_{shuffled}[perm_{arr}[i]]$
    2. output $\pi_{shuffle}$
```

Note that $deck_{init}$ should be private since it will be obtained after the prover masks the deck before it, and hence is known only to the prover.

The proof circuit is constructed in Circom 2.2.1, and checks for the following constraints:

```
For $i$, $j < 52$:
    1. $0 \leq perm_{arr}[i] < 52$
    2. $perm_{arr}[i] \neq perm_{arr}[j]$
    3. $deck_{init}[i] = deck_{shuffled}[perm_{arr}[i]]$
```

This proof ensures that the output of the shuffle is a valid permutation of the input, so the deck has not been "tampered with".

3.1.6 Game

The game we implemented is simple. There is a player and a dealer. The deck is shuffled, then the dealer draws the top card and holds it face-down. The dealer draws and reveals the next card of the deck. Then, the player may choose to have the dealer draw and reveal an additional card. This continues until the player chooses to stop. When the player chooses to stop, the dealer reveals their face-down card and adds its value to the sum of the cards that are already revealed. If the sum of all cards is between 15 and 21 (inclusive), the player wins. Otherwise, the player loses.

3.1.7 Workflow

After setup, the dealer encrypts the cards and shuffles them, outputting proofs for both. The player verifies these proofs. Then, the dealer sends the deck (which is now encrypted and shuffled) to the player. The player shuffles and encrypts the deck, with the dealer verifying the validity of these actions, then sends the final deck back to the dealer. Now the game begins.

Note that the dealer and player both have a 'copy' of the exact same deck, including ordering. The dealer sets aside the first card and deals the player the second card. The dealer and player respectively reveal their (card-specific) decryption keys u_i so that this card can be decrypted and its value revealed. If the player chooses to 'hit', the next card is dealt and the dealer and player again reveal their decryption keys to reveal the card. This process is repeated until the player chooses to 'stand', at which point the dealer and player reveal their decryption keys for the dealer's hidden card and sum the cards to check whether the total is between 15 and 21.

3.2 Discussion of Security

There are constraints that are proposed to determine the validity of a mental poker scheme [5]. They are:

1. Uniqueness of cards
2. Absence of a trusted third-party
3. Uniform random distribution of cards
4. Complete confidentiality of cards
5. Cheating detection with a very high probability
6. Minimal effect of coalitions
7. Complete confidentiality of strategy

'Uniqueness of cards' and 'absence of a trusted third-party' both follow immediately; we'll discuss the rest.

3.2.1 Uniform random distribution of cards

This is guaranteed by the fact that all players get the chance to shuffle. As long as all shuffles are valid (which is guaranteed) and at least one player performs a random shuffle, the ending distribution will be random.

3.2.2 Complete confidentiality of cards

This is guaranteed because the cards are encrypted by all players. Imagine trying to access a card's value while not having one player's decryption key for that card. This would be equivalent to having the value $g^r, v * H_i^r$ and trying to recover v. This problem is intractable under the discrete log hardness assumption.

3.2.3 Cheating detection with a high probability

Cheating can take multiple forms. One way would be to try to decrypt a card to some other value than its original value. This would require coming up with a value k such that $v_1 * H_i^r = v_2 * H_i^k$ which again amounts to solving the discrete log problem.
Another form of cheating could involve attempting to fraudulently shuffle, i.e. tampering with the deck. This should be detected with high probability due to the proof-of-shuffle SNARKs. Similarly, attempts to dishonestly encrypt or decrypt the cards will be detected with high probability by verifiers.
Perhaps players could attempt to simply ignore certain players' encryptions/shuffles of the deck; in class, it was suggested that colluding players could simply ignore one player's shuffle and encryption step. However, this would imply that the deck had not been encrypted by that player. Because the card decryption step is built assuming ALL players have participated in encrypting the deck when the players move to decrypt the cards, the resulting decryption value would be some random garbage value rather than the card's original value. And the non-colluding player would be able to tell that someone has been acting dishonestly.
While we have covered the main possible cheating avenues, you may feel unconvinced that there is *no* possible way to cheat. While we can prove certain parts of the protocol secure, it's very difficult to prove that the protocol as a whole is; indeed, this is the reason that other protocols and implementations have been wrong in the past. Proving the security of the whole system would be a great challenge and could be an interesting direction for future work.

3.2.4 Minimal effect of coalitions

This is given by 3.2.1 and 3.2.2. Even if $n-1$ out of n players are colluding, they cannot shuffle the deck dishonestly as long as the remaining player gives a random shuffle. And without the nth player's decryption keys, the colluding players cannot learn any additional information about any cards.

3.2.5 Complete confidentiality of strategy

This requirement enforces that some cards may not be revealed during or after a game. For example, in poker one doesn't need to reveal their hand after a successful bluff. This is given by our protocol since cards are only revealed with all players' consent. Thus, certain cards can stay hidden after a game is finished.
This requirement is indirectly responsible for the need for proofs-of-correct shuffle in our scheme. If all cards *could* be revealed after a game, one would only need to look at the revealed deck after the game to detect deck tampering from dishonest shuffles. But since this is not possible, we need 'proofs of shuffle' to avoid cheating. This requirement was actually not fulfilled in the original Mental Poker protocol put forth in [1], as their scheme always involved all cards being revealed at the end.

3.3 Performance Numbers

The primary steps in the protocol that are expected to dominate the runtime are the shuffle proof generation and verification steps. On a consumer-grade laptop, the proof generation step took around 575ms on average. This does not include the Groth16 setup and the generation of proving and verification keys. The proof verification step took around 520ms on average, which is expected. The witness generation took 560ms on average.
It is also to note that these steps pertaining to the proofs happen only i times in the game, where i corresponds to the player count since the game requires i players to shuffle and no more than that. For real-world applications, i tends to be small, allowing for efficient & secure games.
The proof and witness sizes should also be taken into consideration when dealing with ZK-SNARKs. In our case, both these sizes are very minimal, with the proof size being around 800 bytes on average, and the witness size being 140 bytes.

4 Impact of results

Our implementation of the discussed protocol is found to be efficient while also ensuring privacy, where the card representations are secure. This project aimed to take a step towards creating a 'plug-and-develop' utility for this protocol, eventually allowing other developers to employ the different functions of the protocol for their own card games. We believe we have succeeded in providing a fundamental implementation for such a utility.

5 Future directions

One interesting direction would be to enforce the outcomes of games on the blockchain. Smart contracts could be written which could verify the outcomes of the games, where every player digitally signs the result and sends them over to the smart contract. These smart contracts could even handle betting for players, and automatically transfer money based on the outcomes of the game. A secure implementation for this could allow for totally decentralized, secure, and fair games.

Another important problem to address is proving the security of the whole protocol, for instance, ensuring that the subroutines are interconnected securely. As discussed in 3.2.3, while individual subroutines are proven to be secure, it is difficult to prove the protocol as a whole cannot be exploited in *any* way at all.

Code quality could be further improved by taking into consideration the different versions of Circom since the circuit syntax tends to heavily vary with each version.

References

[1] Shamir, Adi, Rivest, Ronald L., and Adleman, Leonard M. "Mental Poker." *Springer EBooks*, 1 Jan. 1981, pp. 37–43. Available at: `https://link.springer.com/chapter/10.1007/978-1-4684-6686-7_5`. DOI: 10.1007/978-1-4684-6686-7_5. Accessed 8 Dec. 2024.

[2] Barnett, Adam, and Smart, Nigel P. "Mental Poker Revisited." *Lecture Notes in Computer Science*, 1 Jan. 2003, pp. 370–383. Available at: `https://api.semanticscholar.org/CorpusID:27461759`. DOI: 10.1007/978-3-540-40974-8_29. Accessed 8 Dec. 2024.

[3] Goldwasser, Shafi, and Micali, Silvio. "Probabilistic Encryption & How To Play Mental Poker Keeping Secret All Partial Information." *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, 1982, pp. 365–377. Available at: `https://dl.acm.org/doi/10.1145/800070.802212`. DOI: 10.1145/800070.802212. Accessed 8 Dec. 2024.

[4] Bárány, Imre, and Füredi, Zoltán. "Mental Poker with Three or More Players." *Information and Control*, vol. 59, no. 1-3, 1983, pp. 84–93. Available at: `https://www.sciencedirect.com/science/article/pii/S0019995883800308`. DOI: 10.1016/S0019-9958(83)80030-8. Accessed 8 Dec. 2024.

[5] Crépeau, Claude. "A Secure Poker Protocol That Minimizes the Effect of Player Coalitions." *Lecture Notes in Computer Science*, 1985, pp. 73–86. Available at: `https://link.springer.com/chapter/10.1007/3-540-39799-X_8`. DOI: 10.1007/3-540-39799-x_8. Accessed 8 Dec. 2024.

[6] Crépeau, Claude. "A Zero-Knowledge Poker Protocol that Achieves Confidentiality of the Players' Strategy or How to Achieve an Electronic Poker Face." *Advances in Cryptology — CRYPTO '86*, edited by A. M. Odlyzko, Lecture Notes in Computer Science, vol. 263, Springer, 1987, pp.

239–247. Available at: https://link.springer.com/chapter/10.1007/3-540-47721-7_18. DOI: 10.1007/3-540-47721-7_18. Accessed 8 Dec. 2024.

[7] Zhao, Weiliang, Varadharajan, Vijay, and Mu, Yi. "A Secure Mental Poker Protocol Over The Internet." *Australasian Information Security Workshop*, 2003, pp. 105–109. Available at: https://dl.acm.org/doi/pdf/10.5555/827987.828000. Accessed 8 Dec. 2024.

[8] Castella-Roca, Jordi, and Domingo-Ferrer, Josep. "On the Security of an Efficient TTP-Free Mental Poker Protocol." *International Conference on Information Technology: Coding and Computing (ITCC 04)*, Volume 2, April 5-7, 2004, Las Vegas, Nevada, USA, IEEE Computer Society, pp. 781-784. Available at: https://ieeexplore.ieee.org/document/1286753. DOI: 10.1109/itcc.2004.1286606. Accessed 8 Dec. 2024.

[9] Bayer, Stephanie, and Groth, Jens. "Efficient Zero-Knowledge Argument for Correctness of a Shuffle." *Lecture Notes in Computer Science*, 1 Jan. 2012, pp. 263–280. Available at: https://link.springer.com/chapter/10.1007/978-3-642-29011-4_17. DOI: 10.1007/978-3-642-29011-4_17. Accessed 8 Dec. 2024.