Mental poker

Noel Nathan Planell Bosch

February 2025

1 Normal Game Example

We consider a game with four players: Alice, Bob, Tom, and Nico, where Alice acts as the dealer.

1.1 Key Generation

All players agree on a generator G. Using the algorithm

$$\texttt{Setup}(1\lambda),$$

each player generates a secret and public key pair (sk, pk):

$$\begin{aligned}
\text{Alice:} \quad & \texttt{Setup}(1\lambda) \quad \Rightarrow (pk_{\text{alice}},\, sk_{\text{alice}}) \\
\text{Bob:} \quad & \texttt{Setup}(1\lambda) \quad \Rightarrow (pk_{\text{bob}},\, sk_{\text{bob}}) \\
\text{Tom:} \quad & \texttt{Setup}(1\lambda) \quad \Rightarrow (pk_{\text{tom}},\, sk_{\text{tom}}) \\
\text{Nico:} \quad & \texttt{Setup}(1\lambda) \quad \Rightarrow (pk_{\text{nico}},\, sk_{\text{nico}})
\end{aligned}$$

Each player then broadcasts their public key along with a Schnorr identification proof to demonstrate knowledge of the corresponding secret key.

Next, an aggregated public key is computed:

$$H = pk_{\text{agg}} = pk_{\text{alice}} + pk_{\text{bob}} + pk_{\text{tom}} + pk_{\text{nico}}.$$

1.2 Creating the Deck of Cards

A standard deck consists of:

- 13 ranks: Ace, Two, Three, Four, Five, Six, Seven, Eight, Nine, Ten, Jack, Queen, and King.
- 4 suits: Diamond, Heart, Club, and Spade.

Thus, there are $4 \cdot 13 = 52$ cards.

In our protocol, each card is associated with an elliptic curve point and a corresponding plaintext (for example, the point m_i maps to "Nine of Clubs"). The resulting deck is represented as:

$$[(m_1, \text{A}\clubsuit),\, (m_2, 2\clubsuit),\, \ldots,\, (m_{52}, \text{K}\spadesuit)].$$

The dealer generates this deck. All other players then verify that the cards are valid (i.e., that each elliptic curve point is well-formed and lies on the curve).

1.3 Masking the Cards

For each card key m_i (with $i = 1, \ldots, 52$) from the dictionary, the dealer applies an initial mask. Using a fixed value $r = 1$, the masked representation of each card is computed as:

$$\begin{pmatrix} G \\ m_i + H \end{pmatrix} \quad \text{for } i = 1, \ldots, 52.$$

We denote these masked cards as:

$$[C_1, C_2, \ldots, C_{52}] = \left[\begin{pmatrix} C_a \\ C_b \end{pmatrix}_1, \begin{pmatrix} C_a \\ C_b \end{pmatrix}_2, \ldots, \begin{pmatrix} C_a \\ C_b \end{pmatrix}_{52} \right].$$

Here, the dealer is responsible for applying this initial masking.

1.4 Shuffling the Cards

To ensure that no player can deduce another player's cards, each player sequentially shuffles the deck. The procedure for the first shuffler (the dealer, Alice, in this example) is as follows:

1. **Permutation:** Generate a random permutation of the indices $\{0, 1, \ldots, 51\}$. For example, for a deck with 5 cards, a permutation might map:

$$[C_1, C_2, C_3, C_4, C_5] \to [C_1, C_4, C_5, C_2, C_3].$$

2. **Apply Permutation:** Rearrange the deck according to the generated permutation.

If only a permutation were applied, other players could trace the order of the cards. Therefore, after permuting, the shuffler also *re-masks* the deck. Denote the new deck as:

$$[C_1, C_4, C_5, C_2, C_3] \to [C'_1, C'_2, C'_3, C'_4, C'_5].$$

The re-masking is performed by generating a random number r' and computing:

$$\mathbf{C}' = \mathcal{E}'_H(\mathbf{C}, r') = \mathbf{C} + \begin{pmatrix} r'G \\ r'H \end{pmatrix} = \begin{pmatrix} C_a + r'G \\ C_b + r'H \end{pmatrix}.$$

Since every player applies a shuffle with re-masking in turn, after all shuffles the cards take the form:

$$\begin{pmatrix} (r_{\text{alice}} + r_{\text{bob}} + r_{\text{tom}} + r_{\text{nico}} + r_{\text{initial}})\, G \\ m + (r_{\text{alice}} + r_{\text{bob}} + r_{\text{tom}} + r_{\text{nico}} + r_{\text{initial}})\, H \end{pmatrix}.$$

To ensure correctness and prevent malicious shuffling, each shuffler must provide a zero-knowledge proof (specifically, a Chaum-Pedersen style proof for shuffle correctness, as described in Zero-Knowledge Argument for Correctness of a Shuffle (BayerGroth).

1.5 Revealing the Cards

Cards are dealt based on position. For example, the dealer (Alice) receives the first two cards, the next player receives the third and fourth, and so on.

Since each card $C = (C_a, C_b)$ is masked as

$$C_b = m + rH,$$

where

$$r = r_{\text{alice}} + r_{\text{bob}} + r_{\text{tom}} + r_{\text{nico}},$$

the decryption requires collaboration from all players.

For a card C_i (say, one assigned to Alice), each other player i computes a token:

$$D_i = sk_i \cdot C_a,$$

and sends D_i to Alice. Alice then computes her own:

$$D_{\text{alice}} = sk_{\text{alice}} \cdot C_a.$$

Finally, Alice recovers her card by subtracting the sum of all tokens from C_b:

$$C_b - \Big(D_{\text{alice}} + D_{\text{bob}} + D_{\text{tom}} + D_{\text{nico}}\Big) = \big(m + rH\big) - \Big((sk_{\text{alice}} + sk_{\text{bob}} + sk_{\text{tom}} + sk_{\text{nico}}) \cdot (rG)\Big) = m.$$

Because only Alice knows her own contribution D_{alice}, no other player can learn her card.

1.6 Verification Using Chaum-Pedersen Proofs

Verification is performed in three steps:

1. Masking Verification

To verify that a masked value is consistent with the plaintext M, one proves that there exists an r such that

$$g^r = C_a \quad \text{and} \quad h^r = C_b - M.$$

This Chaum-Pedersen proof ensures that the pair $(C_a,\, C_b - M)$ is correctly computed using the secret r.

2. Re-masking Verification

To verify that the re-masking was applied correctly (i.e., that the transition from (C_a, C_b) to (C'_a, C'_b) is valid), one proves that:

$$g^{r'} = C'_a - C_a \quad \text{and} \quad h^{r'} = C'_b - C_b.$$

This demonstrates that a new mask r' was added correctly.

3. Decryption Verification

Finally, to confirm that the decryption token D_i was correctly generated with secret exponent sk_i, one proves that:

$$g^{sk_i} = C_a^{sk_i}.$$

Since $g^{sk_i} = pk_i$ and $C_a^{sk_i} = D_i$, this proof verifies that C_a corresponds to the public key pk_i without revealing the secret sk_i.

2 Reshuffle Game Example

In some situations—such as in Texas Hold'em poker where five community cards are revealed in later rounds—the game may stall if a player refuses to collaborate or loses connection, as all players must jointly participate in revealing the community cards (every player must supply their token D_i).

The idea of using an $(l-n)$-out-of-l threshold encryption was discarded, since n malicious players collaborating could potentially learn both their own and their opponents' cards.

Proposed Solution for Non-cooperative Players

If a player does not cooperate, the deck is reshuffled without that player, and the game continues with a new deck. However, players who have already been dealt cards must remove their cards from the deck before the reshuffle. This introduces two challenges:

1. **Privacy Issue:** When an honest player discards their cards, other players may learn which cards were removed.
2. **Abuse Issue:** A dishonest player might remove cards that are unfavorable to them.

Addressing the Privacy Issue

Before shuffling, once the cards have been masked, a round of re-masking is performed. To remove a card, the player computes:

$$\mathbf{C}' = \mathbf{C} + \begin{pmatrix} r'G \\ r'H - m \end{pmatrix} = \begin{pmatrix} C_a + r'G \\ C_b + r'H - m \end{pmatrix}.$$

After cancellation, we have:

$$\begin{pmatrix} rG + r'G \\ m + rH + r'H - m \end{pmatrix} = \begin{pmatrix} (r+r')G \\ (r+r')H \end{pmatrix}.$$

If a card is *not* removed, the re-masking proceeds normally:

$$\mathbf{C}' = \mathbf{C} + \begin{pmatrix} r'G \\ r'H \end{pmatrix} = \begin{pmatrix} C_a + r'G \\ C_b + r'H \end{pmatrix}.$$

Ensuring Honest Removal (Zero-Knowledge Proof)

Since subtracting m to cancel a card makes it impossible to use the standard Chaum-Pedersen proof, we must ensure that a player is only removing a card they actually hold. To that end, we propose using a zero-knowledge proof (assumed to be a ZNARK protocol). Let:

- $\overline{C} = (\overline{C}_a, \overline{C}_b)$ denote one of the two cards held by the player. $\begin{pmatrix} \overline{C_a} = \alpha G \\ \overline{C_b} = \overline{m} - \alpha H \end{pmatrix}$

The protocol is defined as follows:

$$\mathrm{ZK}(r',\, sk,\, pk,\, \overline{C}_a,\, \overline{C}_b,\, H,\, g,\, G,\, m,\, C_a,\, C_b) :$$

Check: $pk = sk \cdot g$,
Compute $\overline{m} = \overline{C}_b - sk \cdot \overline{C}_a$,
if $\overline{m} \neq m$, then return $\begin{pmatrix} C_a + r'G \\ C_b + r'H \end{pmatrix}$;

else, return $\begin{pmatrix} C_a + r'G \\ C_b - m + r'H \end{pmatrix}$.

Here, r' and sk remain private, while pk, $\overline{C}_a$, $\overline{C}_b$, H, g, G, m, C_a, and C_b are public.

This zero-knowledge proof ensures that a player can only remove a card they actually hold, thereby preventing any malicious behavior.