Secure Trick-Taking Game Protocols
How to Play Online Spades with Cheaters

Xavier Bultel[1] and Pascal Lafourcade[2]

[1] Univ Rennes, CNRS, IRISA
[2] University Clermont Auvergne, LIMOS, France

Abstract. Trick-Taking Games (TTGs) are card games in which each player plays one of his cards in turn according to a given rule. The player with the highest card then wins the trick, *i.e.,* he gets all the cards that have been played during the round. For instance, Spades is a famous TTG proposed by online casinos, where each player must play a card that follows the leading suit when it is possible. Otherwise, he can play any of his cards. In such a game, a dishonest user can play a wrong card even if he has cards of the leading suit. Since his other cards are hidden, there is no way to detect the cheat. Hence, the other players realize the problem later, *i.e.,* when the cheater plays a card that he is not supposed to have. In this case, the game is biased and is canceled. Our goal is to design protocols that prevent such a cheat for TTGs. We give a security model for secure Spades protocols, and we design a scheme called SecureSpades. This scheme is secure under the Decisional Diffie-Hellman assumption in the random oracle model. Our model and our scheme can be extended to several other TTGs, such as Belotte, Whist, Bridge, etc.

1 Introduction

The first card games originate around the 9th century, during the Tang dynasty. Today, they are played all around the world, and a multitude of different games exist. For instance, Poker is probably the most famous gambling card game. Thanks to the Internet, many web sites implement online card game applications, where players meet other players. Cards games websites require some security guarantees, such as secure access for payment, robust software, trusted servers, and cheating detection protocols. These guarantees are crucial for the reputation of the web site in the card game community.

Spades is a famous online gambling card game. It is a *trick-taking game*: at each round, players take turns playing, then the player that plays the highest card wins the *trick*, *i.e.,* all cards that have been played this round. Moreover, if it is possible, then players must play a card that follows the suit of the first card played in the round, otherwise they can play any other card. However, if a player cheats by playing a card of another suit while he has some cards of the leading suit, there is no way to detect it immediately. The other players will detect the cheat later, if the cheater plays a card of the leading suit. As a result, the game is biased, because players revealed some of their cards, hence players cannot replay the game, which must be canceled. Cheaters often get a penalty, but Spades is a team game, hence the cheater's partner is also punished, even if he is not an accomplice. It is even more unfair if the partners do not know each other and/or do not trust each other, which is the case in online games, where teams are chosen by the server.

To avoid this problem, online Spades web sites use a trusted server that manages the game. This server deals the cards, and prevents players from cheating, which means it knows all the cards of each player. However, having a trusted server is a strong security hypothesis, because if some players corrupt the server, then the security properties do not longer hold.

Our motivation is to design a cryptographic scheme, called SecureSpades, that allows the players to check that the other players do not cheat, whithout revealing any information about the cards of each player, and without any trusted server.

Contributions: In this paper, we focus on Trick-Taking Games (TTGs), which are card games where each player plays one of his cards in turn, and where the player with the highest card wins the trick. For the sake of clarity, we focus our work on Spades, because it is the most played online TTG for real money, and its rules are simple. However, our protocol can be extended to other TTGs, such as Whist or Bridge.

We propose a scheme for Spades that has the following security properties:

- The game server is not trusted.
- The players are convinced that nobody cheats. It means that:
  1. *Theft-resistance*: a player cannot play a card that is not in his hand, nor can a player play cards from the hand of his partner .
  2. *Cheating-resistance*: a player cannot play a card that does not follow the rules of the game (in Spades, if a player has a card of the leading suit, he must play it).
- *Unpredictability*: the cards are dealt at random.
- *Hand-privacy*: the players do not know the hidden cards of the other players.
- *Game-privacy*: at each round, the protocol does not leak any information except for the played cards.

We propose a formal definition of a Spades scheme, then we give a formal definition of the security properties described above. We also design SecureSpades, a protocol based on the Decisional Diffie-Hellman (DDH) assumption, and zero-knowledge proofs. Finally, we prove the security of SecureSpades in the random oracle model.

Our protocol not only ensures all the security properties of the real card games, it also provides additional security features. In real card games, it is not possible to detect cheating exactly when the wrong card is played. In fact, our protocol also allows players to detect cheats that are undetectable with real cards, hence it can be used to create new TTGs, for instance a Spades variant where the game is stopped after 5 rounds. In this variant, if the players do not have to reveal the cards they did not play, then there is no way to prevent them from cheating. However, with our approach, such a game can be securely implemented.

Related Work: In 1982, Goldwasser and Micali introduced the *Mental Poker* problem [9]: it asks whether it is possible to play a fair game of poker without physical cards and without a trusted dealer, *i.e.,* by phone or over the Internet. Since then, several works have focused on this primitive, such as [1,13,15]. In [12], the author brings together references to scientific papers related to this problem.

Most of mental poker protocols are based on the following paradigm. The players encrypt the cards together and shuffle them, then ciphertexts are assigned to each player, and each player receives information from the other players in order to decrypt their own cards. At the end of the game, the players reveal their encryption keys, which reveals the hand of all the players. In trick-taking games, each time a player plays a card, he must prove that the card is in his hand and that he has no *high-priority* card that he should play instead of this card. To achieve this property, we model the deck in a different way: each card is associated to a commitment of the secret key of a player. The player plays a card by proving that the committed secret key matches one of its public keys. This allows the player to prove that he cannot play high-priority cards by proving that none of his public keys match possible high-priority cards.

David *et al.* [7] introduced protocols for secure multi-party computation with penalties and secure cash distribution, which can be used for online poker. Bentov *et al.* [2] give a poker protocol in a stronger security model, which is more efficient than [7]. More recently, David *et al.* [8] proposed *Royale*, a universally composable protocol for securely playing any card games with financial rewards/penalties enforcement.

All of these works focus on mental card game protocols with secure payment distributions, but they cannot prevent players from cheating by playing illegal cards. Indeed, these protocols allow the users to play cards digitally with the same security level as if they play with real cards. Our goal is not only to implement a secure trick-taking game, but also to increase its security, in comparison with its physical version.

Finally, an other line of research is to detect collusion frauds in online card games, as done for instance in [14]. Players may exchange information about their cards using some side channels. The goal of [14] is to detect such a collusion attack via the users' behavior. This work is complementary to ours, because these collusion detection processes can also be used with our protocol.

Outline: In Section 2, we describe the rules of *Spades*. In Section 3, we give an informal overview of our scheme. In Section 4, we present the cryptographic tools used in the paper. In Section 5, we model Spades schemes. In Section 6, we define the security properties. In Section 7, we describe SecureSpades before concluding in the last section.

Acknowledgement: We thank Wouter Lueks for his helpful comments and suggestions. We also thank the tarot players of *Le Checkpoint Café*.

2 Spades Rules

Spades was created in the United States in the 1930s. Since the mid-1990s it has become very popular thanks to its accessibility in online card gaming rooms on the Internet. This game uses a standard deck of 52 cards and allows between two and five players. The most famous version requires four players, which are splitted in two teams of two. As indicated by the name of the game, spades are always trump. We give the rules of Spades for the four players version:

1. All 52 cards are distributed one by one to each player, meaning each player has 13 cards at the beginning of the game.
2. There are 13 successive rounds. In the first round, the first player is chosen at random, and subsequently the player that won the previous round begins. Players then each play a card in turn.
3. At each round, the player who plays the highest card wins the trick (*i.e.,* he takes the four cards played this round, but he cannot replay these cards). The rank of the cards is the following, form highest to lowest: Ace, King, Queen, Jack, 10, 9, 8, 7, 6, 5, 4, 3, 2. Trumps are higher than cards of the suit of the first card of the round, which are higher than all other cards.
4. Each player has to follow the suit of the first card of the round. If a player has no card that follows the suit, then he can play any other cards.
5. The game is finished once all players have played all of their cards.

Before playing the cards, each player bids the number of tricks he expects to perform. The sum of all the propositions for all players should be different from the number of cards per player. At the end of the game, each player calculates his score according to his bid and the number of tricks he has won.

3 An overview of our protocol

We now give an informal overview of our Spades protocol. The idea is that the players must prove that each card they play follows the rule of the game. More precisely, the player first proves that he has the played card. If this card does not follow the suit, then he proves that none of his other cards are of the leading suit.

1. **Dealing cards**: We need to model the cards in such a way that these proofs are feasible. Each player i generates 13 pairs of public/private keys $(\mathsf{pk}_{i,j}, \mathsf{sk}_{i,j})$ (for $1 \leq j \leq 13$). To deal the cards, the players run a protocol that privately assigns each key to each card with the following steps: (i) each player generates commitments on his 13 secret keys, (ii) the players group all the $13 \cdot 4 = 52$ commitments together, (iii) each player shuffles and randomizes the commitments in turn , (iv) the players jointly associate each commitment to each card of the deck at random. The hand of a player is the set of the 13 cards that match the commitments of his secret keys. Figure 1 illustrates our dealing cards protocol, where $c(\mathsf{sk})$ denotes the commitment of a secret key sk, and $c'(\mathsf{sk})$ denotes the randomization of $c(\mathsf{sk})$. In this example, the 1st card of player 1 is A♣, his 2^{nd} card is 2♡, and his 13th card

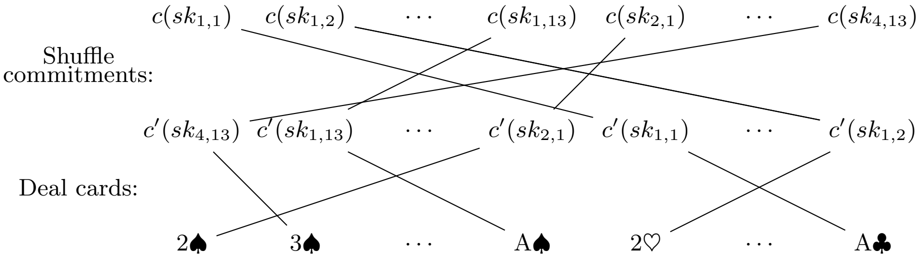

Fig. 1. Dealing cards in our Spades protocol.

is A♠. Note that the commitments and the public keys must be unlinkable for anyone who does not know the corresponding secret keys.

2. **Play a card**: To play a card, the player proves that this card matches the commitment of one of his secret keys. If the player does not follow the suit, then he proves that none of his other cards are of the leading suit. To do so, he proves that each commitment that matches a card of a non-leading suit commits one of his (not yet used) keys.

4 Cryptographic Tools

We present the cryptographic tools used throughout this paper.

Definition 1 (DDH [4]). *Let $\mathbb{G}$ be a prime order group. The DDH assumption states that given $(g, g^a, g^b, g^z) \in \mathbb{G}^4$, it is hard to decide whether $z = a \cdot b$ or not.*

A *n-party random generator* is a protocol that allows n users to generate a random number, even if $n-1$ users are dishonest.

Definition 2 (Multi-party random generator [3]). *A n-party $\mathcal{S}$-random generator $\mathsf{RG}_{P_1,\dots,P_n}$ is a protocol where n parties $(P_1, \dots, P_n)$ interact, and return $s \in \mathcal{S}$. Such a protocol is said to be secure when for any polynomial time distinguisher $\mathcal{D}$, any polynomial time adversary $\mathcal{A}$, there exists a negligible function ϵ such that: $|Pr[1 \leftarrow D(s) : s \overset{\$}{\leftarrow} \mathcal{S}] - Pr[1 \leftarrow D(s) : s \leftarrow \mathsf{RG}_{\mathcal{C},\mathcal{A}}(k)]| \leq \epsilon(k)$ where $s \overset{\$}{\leftarrow} \mathsf{RG}_{\mathcal{C},\mathcal{A}}$ denotes the output of $\mathcal{C}$ at the end of the protocol RG where $\mathcal{C}$ plays the role of a honest user, and $\mathcal{A}$ plays the role of the $n-1$ other users.*

Inspired by [3], we propose the following multi-party random generator protocol based on the random oracle model (ROM).

Definition 3. *Let $\mathcal{S}$ be a set and n be an integer, and let $H : \{0,1\}^* \to \{0,1\}^k$ and $H' : \{0,1\}^* \to \mathcal{S}$ be two hash functions simulated by random oracles. The protocol $\mathsf{RandGen}^{\mathcal{S}}_{P_1,\dots,P_n}(k)$ is a n-party $\mathcal{S}$-random generator defined as follows. Each player P_i (where $1 \leq i \leq n$) chooses $r_i \overset{\$}{\leftarrow} \{0,1\}^k$ at random, computes $H(r_i)$, and broadcasts it, then each player reveals r_i. Each player returns $H'(r_0|| \dots ||r_n)$.*

Lemma 1. *For any set $\mathcal{S}$ and any integer n, $\mathsf{RandGen}^{\mathcal{S}}_{P_1,\ldots,P_n}(k)$ is secure in the random oracle model.*

The proof of this lemma is given in Appendix. The idea is that dishonest parties cannot guess the r_i of the honest parties before revealing their commitments, hence they cannot predict $H(r_0||\ldots||r_n)$.

A (non-interactive) *Zero-Knowledge Proof of Knowledge* (ZKP) [10] for a binary relation $\mathcal{R}$ allows a prover knowing a witness w to convince a verifier that a statement s verifies $(s, w) \in \mathcal{R}$ without leaking any information. Throughout this paper, we use the Camenisch and Stadler notation [6], *i.e.,* $\mathsf{ZK}\{(w) : (w, s) \in \mathcal{R}\}$ denotes the proof of knowledge of w for the statement s and the relation $\mathcal{R}$. Such a proof is said to be *extractable* when given an algorithm that generates valid proofs with some probability, there exists an algorithm that returns the corresponding witness in a similar running time with at least the same probability. Such a proof is said to be *zero-knowledge* when there exists a polynomial time simulator that follows the same probability distribution as an honest prover.

5 Formal Definitions

We formalize Spades schemes and the corresponding security requirements. We model a 52 cards deck by a tuple $D = (\mathsf{id}_1, \ldots, \mathsf{id}_{52})$ such that $\forall\ i \in [\![1, 52]\!]$, $\mathsf{id}_i = (\mathsf{id}_i.\mathsf{suit}, \mathsf{id}_i.\mathsf{val}) \in \{\heartsuit, \spadesuit, \diamondsuit, \clubsuit\} \times \{1, \ldots, 10, \mathsf{J}, \mathsf{Q}, \mathsf{K}\}$ is called *a card*, where $\forall\ (i, j) \in [\![1, 52]\!]^2$ such that $i \neq j$, $\mathsf{id}_i \neq \mathsf{id}_j$. The set of all possible decks is denoted by Decks.

We first define Spades schemes, which are tuples that contain all the algorithms that are used by the players. KeyGen allows each player to generate its public/secret key. $\mathsf{GKeyGen}$ allows the players to generate a public game key. $\mathsf{DeckGen}$ is a protocol that generates a random deck. $\mathsf{GetHand}$ determines the hand of a given player from his secret key and the game key. Play allows a player to play a card, and to prove that it is a *legal* play. Verif allows the other players to check this proof. Finally, $\mathsf{GetSuit}$ returns the leading suit of the current round (in Spades, the suit of the first card played during this round).

Definition 4. *A* Spade *scheme is a tuple of eight algorithms* $W = (\mathsf{Init}, \mathsf{KeyGen}, \mathsf{GKeyGen}, \mathsf{DeckGen}, \mathsf{GetHand}, \mathsf{Play}, \mathsf{Verif}, \mathsf{GetSuit})$ *defined as follows:*

$\mathsf{Init}(k)$: *It returns a setup* setup*.*

$\mathsf{KeyGen}(\mathsf{setup})$: *It returns a key pair* $(\mathsf{pk}, \mathsf{sk})$.

$\mathsf{GKeyGen}$: *It is a* 4*-party protocol, where for all* $j \in [\![1, 4]\!]$ *the* j^{th} *party is denoted* P_j *and takes as input* $(\mathsf{sk}_j, \{\mathsf{pk}_i\}_{1\leq i\leq 4})$*. This protocol returns a game public key* PK*, or the bottom symbol* $\perp$*.*

$\mathsf{DeckGen}$: *It is a* 4*-party* Decks*-random generator.*

$\mathsf{GetHand}(\mathsf{sk}, \mathsf{pk}, \mathsf{PK}, D)$: *It returns a set of* 13 *different cards* H *called* a hand *(where* $D \in \mathsf{Decks}$*).*

$\mathsf{Play}(n, \mathsf{id}, \mathsf{sk}, \mathsf{pk}, \mathrm{st}, \mathsf{PK}, D)$: *It takes as input a player index* $n \in [\![1, 4]\!]$*, a card* id*, a pair of secret/public key, a global state* st *that stores the relevent information about the previous plays, the game public key* PK *and the deck* D*, and returns a proof* Π*, and the updated global state* st'*.*

$\mathsf{Verif}(n, \mathsf{id}, \mathit{\Pi}, \mathsf{pk}, \mathrm{st}, \mathrm{st}', \mathsf{PK}, D)$: *It takes as input a player index* $n \in [\![1, 4]\!]$*, a card identity* id*, a proof* $\mathit{\Pi}$ *generated by the algorithm* Verif*, the global state* st *and the updated global state* st'*, the game public key* PK *and the deck* D*, and returns a bit* b*. If* $b = 1$*, we say that* $\mathit{\Pi}$ *is* valid*.*

$\mathsf{GetSuit}(\mathrm{st})$: *It returns a suit* $\mathsf{suit} \in \{\heartsuit, \spadesuit, \diamondsuit, \clubsuit\}$ *from the current global state of the game* st*, where* suit *is the leading suit for the current turn.*

We then define the *Spades protocol*, which allows four players to play Spades using the algorithms of the Spades scheme. It is divided in four phases:

Initialisation phase: One player generates and broadcasts the public setup.

Keys generation phase: After they have generated their public/private keys, the players run $\mathsf{GKeyGen}$ to generate the game key together.

Shuffle phase: The players choose a deck using $\mathsf{DeckGen}$, then they compute their own hand using $\mathsf{GetHand}$.

Game phase: Finally, they play in turn using the algorithms Play and Verif to prove the validity of the cards they play. If some verification fails, the player has to cancel only the last card he has played, and to simply play another card.

Definition 5. *Let* $W = (\mathsf{Init}, \mathsf{KeyGen}, \mathsf{GKeyGen}, \mathsf{DeckGen}, \mathsf{GetHand}, \mathsf{Play}, \mathsf{Verif}, \mathsf{GetSuit})$ *be a Spades scheme and* $k \in \mathbb{N}$ *be a security parameter. Let* Player_1*,* Player_2*,* Player_3*,* Player_4 *be four polynomial time algorithms. The* Spades protocol *instantiated by* W *and the setup* setup *between* $\mathsf{Player}_1, \mathsf{Player}_2$*,* Player_3 *and* Player_4 *is the following protocol:*

Initialisation phase: Player_1 *runs* $\mathit{setup} \leftarrow \mathsf{Init}(k)$ *and broadcasts* setup*.*

Keys generation phase: *The players set* $\mathrm{st} = \perp$*. Each player* Player_i *runs* $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathit{setup})$ *and broadcasts* pk_i*, then the players generate* PK *by running the protocol* $\mathsf{GKeyGen}$ *together.*

Shuffle phase: *The players generate a deck* $D \in \mathsf{Decks}$ *by running* $\mathsf{DeckGen}$ *together. For all* $i \in [\![1, 4]\!]$, Player_i *runs* $H_i \leftarrow \mathsf{GetHand}(\mathsf{sk}_i, \mathsf{pk}_i, \mathsf{PK}, D)$*.*

Game phase: *This phase is composed of* 52 *(sequential) steps (corresponding to the* 52 *cards played in a game). The players initialize the current player index* $p = 1$*. At each turn,* Player_p *designates the player who plays. Each step proceeds as follows:*

- Player_p *chooses* $\mathsf{id} \in H_p$*, then runs* $(\mathit{\Pi}, \mathrm{st}') \leftarrow \mathsf{Play}(p, \mathsf{id}, \mathsf{sk}_p, \mathsf{pk}_p, \mathrm{st}, \mathsf{PK}, D)$*.*
- *For all* $i \in [\![1, 4]\!] \setminus \{p\}$*,* Player_p *sends* $(\mathsf{id}, \mathit{\Pi}, \mathrm{st}')$ *to* Player_i*.*
- *Each* Player_i *then checks that* $\mathsf{Verif}(p, \mathsf{id}, \mathit{\Pi}, \mathsf{pk}_p, \mathrm{st}, \mathrm{st}', \mathsf{PK}, D) = 1$*, otherwise,* Player_i *sends* `error` *to* Player_p*, who repeats this step and plays a valid card.*
- *If* $\mathsf{Verif}(p, \mathsf{id}, \mathit{\Pi}, \mathsf{pk}_p, \mathrm{st}, \mathrm{st}', \mathsf{PK}, D) = 1$*, all players update the state* $\mathrm{st} := \mathrm{st}'$*, and update the index* p *that points the next player according to the rule of the game.*

6 Security Properties

We first define *Spades strategies*. In a card game, each player chooses what card he wants to play depending on his hand and the previously played cards of the other players. In order to formalize the security of our protocol, we need to model honest players who choose the cards they play themselves. A Spades strategy is an algorithm that decides which card to play using all known information by a given player. We define security experiments where the choices of each honest player is simulated by a Spades strategy. The idea is that a Spades scheme is secure if for any polynomial time adversary, the probability of winning the experiment is negligible, whatever the Spades strategies used by the experiment.

Definition 6. *A* Spades strategy *is a polynomial time algorithm* Strat *that takes as input a tuple of cards* played *(which represents all cards played at some point in a Spades game) and a set of cards* hand *(which represents all cards of a player at the same point), a first player index* p_**, a player index* p*, and that returns a card* $\mathsf{id} \in \mathsf{Hand}$ *which is valid according to the rules of Spades (i.e., that follows the suit of the first card of the current round).*

We define an experiment where a challenger simulates the Spades protocol to an adversary. We use this experiment to define Spades' security properties. The adversary first chooses the index of the player he wants to corrupt. The challenger generates the public/secret keys of the three other users, then the adversary sends his public key together with the index of an *accomplice*. The accomplice allows the experiment to capture the attacks where a dishonest player and his game partner collude. The adversary has access to the private key of all players. The adversary and the challenger then run the game key and the deck generation protocol, such that the adversary plays the role of the corrupted player and the accomplice. The challenger generates the hand of each player. Note that the challenger cannot use the hand generation algorithm for the corrupted player, because he does not know his secret key; however, the challenger can deduce this hand because it contains the 13 cards that are not in the hand of the three other users. Finally, the challenger and the adversary run the game phase, such that the adversary plays the role of the corrupted user and his accomplice.

Definition 7. *Let* $W = (\mathsf{Init}, \mathsf{KeyGen}, \mathsf{GKeyGen}, \mathsf{DeckGen}, \mathsf{GetHand}, \mathsf{Play}, \mathsf{Verif}, \mathsf{GetSuit})$ *be a Spades scheme,* $S = (\mathsf{Strat}_1, \mathsf{Strat}_2, \mathsf{Strat}_3, \mathsf{Strat}_4)$ *be a tuple of strategies, and* $k \in \mathbb{N}$ *be a security parameter. Let* $\mathcal{A}$ *and* $\mathcal{C}$ *be two polynomial time algorithms. The* Spades experiment $\mathsf{Exp}^{\mathsf{Spades}}_{W,S,\mathcal{A}}(k)$ *instantiated by* W *and* S *between the adversary* $\mathcal{A}$ *and the challenger* $\mathcal{C}$ *is defined as follows:*

Keys generation phase: $\mathcal{C}$ *runs* $\mathsf{setup} \leftarrow \mathsf{Init}(k)$*, sets* $\mathrm{st} = \perp$*, and sends the pair* $(\mathsf{setup}, \mathrm{st})$ *to* $\mathcal{A}$*, who returns a corrupted user index* $i_c \in [\![1, 4]\!]$*. For all* $i \in [\![1, 4]\!] \setminus \{i_c\}$*,* $\mathcal{C}$ *runs* $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{setup})$ *and sends* $(\mathsf{pk}_i, \mathsf{sk}_i)$ *to* $\mathcal{A}$*, who returns the public key* pk_{i_c} *and an accomplice index* i_a*.*

Game key generation phase: $\mathcal{C}$ *and* $\mathcal{A}$ *generate* PK *by running the algorithm* $\mathsf{GKeyGen}$ *together, such that* $\mathcal{A}$ *plays the role of the players* P_{i_c} *and* P_{i_a}*, and* $\mathcal{C}$ *plays the role of the other players. If* $\mathsf{PK} = \perp$*, then* $\mathcal{C}$ *aborts and returns* 0*.*

Shuffle phase: *$\mathcal{C}$ and $\mathcal{A}$ generate D by running the algorithm* $\mathsf{DeckGen}$ *together, such that $\mathcal{A}$ plays the role of the players P_{i_c} and P_{i_a}, and $\mathcal{C}$ plays the role of the two other players. $\mathcal{C}$ sets $p = 1$ and parses D as* $(\mathsf{id}_1, \dots, \mathsf{id}_{52})$*. For all $i \in \llbracket 1, 4 \rrbracket \setminus \{i_c\}$, $\mathcal{C}$ runs* $H_i \leftarrow \mathsf{GetHand}(\mathsf{sk}_i, \mathsf{pk}_i, \mathsf{PK}, D)$*, and sets* $H_{i_c} = \{\mathsf{id}_i\}_{1 \leq i \leq 52} \setminus (\cup_{i=1; i \neq i_c}^{4} H_i)$*.*

Game phase: *$\mathcal{C}$ initializes the current player index $p = 1$ and the corrupted play index $\gamma = 0$, and* $\mathsf{played} = \perp$*. For $i \in \llbracket 1, 52 \rrbracket$:*

If $p \neq i_c$ and $p \neq i_a$: *$\mathcal{C}$ runs* $\mathsf{id} \leftarrow \mathsf{Strat}_p(\mathsf{played}, H_p, p_*, p)$*, then $\mathcal{C}$ runs* $(\Pi, \mathrm{st}') \leftarrow \mathsf{Play}(p, \mathsf{id}, \mathsf{sk}_p, \mathsf{pk}_p, \mathrm{st}, \mathsf{PK}, D)$*. $\mathcal{C}$ sends* $(\mathsf{id}, \Pi, \mathrm{st}')$ *to $\mathcal{A}$ and updates* $\mathrm{st} := \mathrm{st}'$*.*

If $p = i_a$: *$\mathcal{C}$ receives* $(\mathsf{id}, \Pi, \mathrm{st}')$ *from $\mathcal{A}$. If* $\mathsf{Verif}(i_a, \mathsf{id}, \Pi, \mathsf{pk}_{i_a}, \mathrm{st}, \mathrm{st}', \mathsf{PK}, D) = 0$*, then $\mathcal{C}$ aborts and the experiment returns 0. Else, $\mathcal{C}$ updates* $\mathrm{st} := \mathrm{st}'$*.*

If $p = i_c$: *$\mathcal{C}$ increments $\gamma := \gamma + 1$, then receives* $(\mathsf{id}, \Pi, \mathrm{st}')$ *from $\mathcal{A}$ and sets* $(\mathsf{id}_{i_c,\gamma}, \Pi_{i_c,\gamma}) = (\mathsf{id}, \Pi)$*. $\mathcal{C}$ sets $\mathrm{st}_\gamma = \mathrm{st}$ and $\mathrm{st}'_\gamma = \mathrm{st}'$. $\mathcal{C}$ sets* $\mathsf{suit}_{i_c,\gamma} = \mathsf{GetSuit}(\mathrm{st})$*. If* $\mathsf{Verif}(i_c, \mathsf{id}_{i_c,\gamma}, \Pi_{i_c,\gamma}, \mathsf{pk}_{i_c}, \mathrm{st}_\gamma, \mathrm{st}'_\gamma, \mathsf{PK}, D) = 0$*, then $\mathcal{C}$ aborts and the experiment returns 0. Else, $\mathcal{C}$ updates* $\mathrm{st} := \mathrm{st}'$*.*

$\mathcal{C}$ then updates the index p that points to the next player according to the rule of Spades, parses played *as* $(\mathsf{pl}_1, \dots, \mathsf{pl}_n)$ *(where $n = |\mathsf{played}|$) and updates* $\mathsf{played} := (\mathsf{pl}_1, \dots, \mathsf{pl}_n, \mathsf{id})$*.*

Final phase: *The experiment returns 1.*

The first security property of a Spades scheme is the *theft-resistance*, which ensures that no adversary is able to play a card that is not in his hand, even if the card is in the hand of his accomplice. On the other words, two partners are not able to exchange their cards.

Definition 8. *A Spades scheme W is said to be* theft-resistant *if for any tuple of strategies* $S = (\mathsf{Strat}_1, \mathsf{Strat}_2, \mathsf{Strat}_3, \mathsf{Strat}_4)$ *and any polynomial time adversary $\mathcal{A}$ who plays the Spade experiment instantiated by W and S, the probability that there exists $\gamma \in \llbracket 1, 13 \rrbracket$ such that:*

- $\mathsf{Verif}(i_c, \mathsf{id}_{i_c,\gamma}, \Pi_{i_c,\gamma}, \mathsf{pk}_{i_c}, \mathrm{st}_\gamma, \mathrm{st}'_\gamma, \mathsf{PK}, D) = 1$*, i.e., the γ^{th} play of the adversary is accepted for the card* $\mathsf{id}_{i_c,\gamma}$*; and*
- $\forall\ \mathsf{id} \in H_{i_c}, \mathsf{id}_{i_c,\gamma} \neq \mathsf{id}$*, i.e., the card* $\mathsf{id}_{i_c,\gamma}$ *is not in the adversary hand;*

is negligible.

We then define the *cheating-resistance* property, which ensures that no adversary is able to play a card if he should play another valid one.

Definition 9. *A Spades scheme W is said to be* cheating-resistant *if for any tuple of strategies* $S = (\mathsf{Strat}_1, \mathsf{Strat}_2, \mathsf{Strat}_3, \mathsf{Strat}_4)$ *and any polynomial time adversary $\mathcal{A}$ who plays the Spade experiment instantiated by W and S, the probability that there exists $\gamma \in \llbracket 1, 13 \rrbracket$ such that:*

- $\mathsf{Verif}(i_c \mathsf{id}_{i_c,\gamma}, \Pi_{i_c,\gamma}, \mathsf{pk}_{i_c}, \mathrm{st}_\gamma, \mathrm{st}'_\gamma, \mathsf{PK}, D) = 1$*, i.e., the γ^{th} play of the adversary is accepted for the card* $\mathsf{id}_{i_c,\gamma}$*; and*
- $\mathsf{id}_{i_c,\gamma}.\mathsf{suit} \neq \mathsf{suit}_{i_c,\gamma}$ *and* $\mathsf{suit}_{i_c,\gamma} \neq \perp$ *i.e., the suit of the card* $\mathsf{id}_{i_c,\gamma}$ *is not the leading suit; and*

- $\exists\ \bar{\mathsf{id}} \in H_{i_c}$ *such that:* $\forall\ l \leq \gamma, \mathsf{id}_{i_c,l} \neq \bar{\mathsf{id}}$ *and* $\bar{\mathsf{id}}.\mathsf{suit} = \mathsf{suit}_{i_c,\gamma}$*. i.e., the adversary has a card of the leading suit in his hand that was not already played before the* γ^{th} *play;*

is negligible.

We define the *unpredictability*, which ensures that no adversary can influence the card dealing, *i.e.,* $\mathcal{A}$ cannot predict which card will be in which hand.

Definition 10. *A Spades scheme* W *is said to be* unpredictable *if for any tuple of strategies* $S = (\mathsf{Strat}_1, \mathsf{Strat}_2, \mathsf{Strat}_3, \mathsf{Strat}_4)$*, any polynomial time adversary* $\mathcal{A}$ *who plays the Spades experiment instantiated by* W *and* S*, for all* $i \in [\![1, 52]\!]$ *the probability that* $\mathsf{id}_i \in H_{i_c}$ *is negligibly close to* $1/4$*.*

We introduce a new experiment that is called the *hand Spades experiment*, where the challenger simulates the key generation phase of the Spades protocol (but not the game phase). In this experiment the adversary does not know the private keys of the other players and has no accomplice. This experiment will be used to model the attacks where an adversary tries to guess the cards of the other players, including his partner.

Definition 11. *Let* $W = (\mathsf{Init}, \mathsf{KeyGen}, \mathsf{GKeyGen}, \mathsf{DeckGen}, \mathsf{GetHand}, \mathsf{Play}, \mathsf{Verif}, \mathsf{GetSuit})$ *be a Spades scheme and* $k \in \mathbb{N}$ *be a security parameter. Let* $\mathcal{A}$ *and* $\mathcal{C}$ *be two polynomial time algorithms. The* hand Spades experiment $\mathit{Exp}^{\mathsf{HSpades}}_{W,\mathcal{A}}(k)$ *instantiated by* W *between the adversary* $\mathcal{A}$ *and the challenger* $\mathcal{C}$ *is defined by:*

Key generation phase: $\mathcal{C}$ *runs* $\mathit{setup} \leftarrow \mathsf{Init}(k)$*. It sets* $\mathrm{st} = \perp$*. It sends the pair* $(\mathit{setup}, \mathrm{st})$ *to* $\mathcal{A}$*, who returns* $i_c \in [\![1, 4]\!]$*. For all* $i \in [\![1, 4]\!] \backslash \{i_c\}$*,* $\mathcal{C}$ *runs* $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathit{setup})$ *and sends* pk_i *to* $\mathcal{A}$*, who returns* pk_{i_c}*.*

Game key generation phase: $\mathcal{C}$ *and* $\mathcal{A}$ *generate* PK *by running the algorithm* $\mathsf{GKeyGen}$ *together, such that* $\mathcal{A}$ *plays the role of* P_{i_c}*, and* $\mathcal{C}$ *plays the role of the three other players. If* $\mathsf{PK} = \perp$*, then* $\mathcal{C}$ *aborts and returns* 0*.*

Shuffle phase: $\mathcal{A}$ *sends a deck* $D \in \mathsf{Decks}$ *to* $\mathcal{C}$*.* $\mathcal{C}$ *parses* D *as* $(\mathsf{id}_1, \ldots, \mathsf{id}_{52})$*. For all* $i \in [\![1, 4]\!] \backslash \{i_c\}$*,* $\mathcal{C}$ *runs* $H_i \leftarrow \mathsf{GetHand}(\mathsf{sk}_i, \mathsf{pk}_i, \mathsf{PK}, D)$*, and sets* $H_{i_c} = \{\mathsf{id}_i\}_{1 \leq i \leq 52} \backslash (\cup^4_{i=1; i \neq i_c} H_i)$*.*

Challenge phase: $\mathcal{C}$ *picks* (θ_0, θ_1) *in* $([\![1, 4]\!] \backslash \{i_c\})^2$ *such that* $\theta_0 \neq \theta_1$*.* $\mathcal{C}$ *picks* $b \xleftarrow{\$} \{0, 1\}$ *and* $\bar{\mathsf{id}} \xleftarrow{\$} H_{\theta_b}$*, and sends* $(\bar{id}, \theta_0, \theta_1)$ *to* $\mathcal{A}$*, who returns* b_**.*

Final phase: *If* $b = b_*$*, then* $\mathcal{C}$ *returns* 1*, else it returns* 0*.*

We then define the *hand-privacy.* This property ensures that an adversary has no information about the hand of the other players before the game phase is run.

Definition 12. *A Spades scheme* W *is said to be* hand-private *if for any tuple of strategies* $S = (\mathsf{Strat}_1, \mathsf{Strat}_2, \mathsf{Strat}_3, \mathsf{Strat}_4)$ *and any polynomial time adversary* $\mathcal{A}$ *who plays the hand-Spades experiment instantiated by* W *and* S*, the probability that the experiment returns* 1 *is negligibly closed to* $1/2$*.*

The last property is the *game-privacy.* The idea is that, at each step of the game phase, the players learn nothing else than the cards that have been previously played. We show that, after the game key is generated, each player is able to simulate all the protocol interactions knowing the players' strategies. More formally, there exists a simulator that takes as input values known by the player such that the player cannot distinguish whether he plays the real game experiment or he interacts with the simulator.

Definition 13. *For any* $k \in \mathbb{N}$*, any Spades scheme* W*, any quadruplet of strategies* S*, any adversary* $\mathcal{D}$ *and any* $K = (\textsf{\textit{setup}}, \textsf{pk}_{i_c}, \{(\textsf{pk}_i, \textsf{sk}_i)\}_{1 \leq i \leq 4; i \neq i_c}, \textsf{PK})$*,* $Exp^{\textsf{Spades}}_{W,S,K,\mathcal{D}}(k)$ *denotes the same experiment as* $Exp^{\textsf{Spades}}_{W,S,\mathcal{D}}(k)$ *except:*

1. *The challenger and the adversary use the setup and the keys in* K *instead of generating fresh setup and keys during the experiment.*
2. *The challenger does not send* $\textsf{sk}_i$ *for all* $1 \leq i \leq 4$ *such that* $i \neq i_c$ *to* $\mathcal{A}$*, and* $\mathcal{A}$ *has no accomplice.*

A Spades scheme W *is said to be* game-private *if there exists a polynomial time simulator* $\textsf{Sim}$ *such that for any tuple of strategies* S *and any polynomial time* 5*-party algorithm* $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_4, \mathcal{D}_5)$*,* $|P_{\textsf{real}}(\mathcal{D}, k) - P_{\textsf{sim}}(\mathcal{D}, k)|$ *is negligible, where*

$$P_{\textsf{real}}(k) =$$

$$\textsf{\textit{Pr}}\left[1 \leftarrow \mathcal{D}_5(\textsf{vw}) : \begin{array}{l} \textsf{\textit{setup}} \leftarrow \textsf{Init}(k); i_c \leftarrow \mathcal{D}_1(\textsf{\textit{setup}}); \\ \forall i \in [\![1, 4]\!], \\ \textit{If } i \neq i_c, (\textsf{pk}_i, \textsf{sk}_i) \leftarrow \textsf{KeyGen}(\textsf{\textit{setup}}); \\ \textit{Else } \textsf{pk}_{i_c} \leftarrow \mathcal{D}_2(\textsf{\textit{setup}}, \{\textsf{pk}_i\}_{1 \leq j \leq i}, \textsf{vw}); \\ \textsf{PK} \leftarrow \textsf{GKeyGen}_{P_1, P_2, P_3, P_4} \textit{ where } P_{i_c} = \mathcal{D}_3; \\ K := (\textsf{\textit{setup}}, \textsf{pk}_{i_c}, \{(\textsf{pk}_i, \textsf{sk}_i)\}_{1 \leq i \leq 4; i \neq i_c}, \textsf{PK}); \\ \textit{If } \textsf{PK} = \perp, \textsf{vw}_r := \perp; \\ \textit{Else } b \leftarrow \textsf{\textit{Exp}}^{\textsf{Spades}}_{W,S,K,\mathcal{D}_4(\textsf{vw})}(k); \end{array} \right]$$

$$P_{\textsf{sim}}(k) =$$

$$\textsf{\textit{Pr}}\left[1 \leftarrow \mathcal{D}_5(\textsf{vw}) : \begin{array}{l} \textsf{\textit{setup}} \leftarrow \textsf{Init}(k); i_c \leftarrow \mathcal{D}_1(\textsf{\textit{setup}}); \\ \forall i \in [\![1, 4]\!], \\ \textit{If } i \neq i_c, (\textsf{pk}_i, \textsf{sk}_i) \leftarrow \textsf{KeyGen}(\textsf{\textit{setup}}); \\ \textit{Else } \textsf{pk}_{i_c} \leftarrow \mathcal{D}_2(\textsf{\textit{setup}}, \{\textsf{pk}_i\}_{1 \leq j \leq i}, \textsf{vw}); \\ \textsf{PK} \leftarrow \textsf{GKeyGen}_{P_1, P_2, P_3, P_4} \textit{ where } P_{i_c} = \mathcal{D}_3; \\ \textit{If } \textsf{PK} = \perp, \textsf{vw}_r := \perp; \\ \textit{Else } b \leftarrow \textsf{Sim}^{\textsf{Spades}}_{W,S,\mathcal{D}_4(\textsf{vw})}(k, \textsf{\textit{setup}}, i_c, \{\textsf{pk}_i\}_{1 \leq i \leq 4}, \textsf{PK}, \textsf{vw}); \end{array} \right]$$

and where $\textsf{vw}$ *denotes the view of* $\mathcal{D}$*, i.e., all the values sent and received by each algorithm of* $\mathcal{D}$ *during his interaction with the experiment.*

Note that if a scheme is both hand-private and private-game, then players have no information about the other players' hands except for all the cards they have already played.

7 Schemes

We first informally show how our protocol, SecureSpades, works, then we give its formal definition.

Keys generation. Each player i generates 13 key pairs $(\mathsf{pk}_{i,j}, \mathsf{sk}_{i,j})$ for $1 \leq j \leq 13$ such that $\mathsf{pk}_{i,j} = g^{\mathsf{sk}_{i,j}}$. The players then generate a game key PK together, which is made of 52 pairs (h_l, PK_l) such that $h_l{}^{\mathsf{sk}_{i,j}} = \mathsf{PK}_l$. The keys PK_l are shuffled, meaning each player does not know which PK_l corresponds to which $\mathsf{pk}_{i,j}$, except for his own public keys. To build PK, for all $l \in \llbracket 1, 52 \rrbracket$ the players set $h_{0,l} = g$ and $\mathsf{PK}_{0,l} = \mathsf{pk}_{i,j}$ such that (i, j) is in $\llbracket 1, 4 \rrbracket \times \llbracket 1, 13 \rrbracket$ and is different for each l. Note that it holds that $h_{0,l}{}^{\mathsf{sk}_{i,j}} = \mathsf{PK}_{0,l}$. The first player then randomizes and shuffles all pairs $(h_{0,l}, \mathsf{PK}_{0,l})$, *i.e.,* he chooses a random vector r and a random permutation δ and computes $h_{1,l} = (h_{0,\delta(i)})^{r_i}$ and $\mathsf{PK}_{1,l} = (\mathsf{PK}_{0,\delta(i)})^{r_i}$. The three other players randomize and shuffle the pairs $(h_{n,l}, \mathsf{PK}_{n,l})$ in order to obtain the pairs $(h_{n+1,l}, \mathsf{PK}_{n+1,l})$ for $1 \leq n \leq 3$ in turn in the same way, then they set $(h_l, \mathsf{PK}_l) = (h_{4,l}, \mathsf{PK}_{4,l})$ for all l. If the shuffles are correctly built, then it holds that for each l there exists a different pair (i, j) such that $h_l{}^{\mathsf{sk}_{i,j}} = \mathsf{PK}_l$. After each shuffle, the player proves that each $(h_{i,l}, \mathsf{PK}_{i,l})$ is a correct randomization of one $(h_{i-1,l'}, \mathsf{PK}_{i-1,l'})$ where $1 \leq l' \leq 52$. Each player then checks that each of his secret keys match one PK_l, otherwise he aborts the protocol. Since each player shuffles the keys using a secret permutation, they do not know which PK_l matches which $\mathsf{pk}_{i,j}$, except for their own public keys.

Hand generation. Players generate a random deck $D = (\mathsf{id}_1, \ldots, \mathsf{id}_{52})$ using the $\mathsf{RandGen}^{\mathsf{Deck}}$ protocol, then for all $1 \leq l \leq 52$, the key PK_l corresponds to the card id_l. The hand of the player i is the set of all cards id_l such that there exists $1 \leq j \leq 13$ such that $h_l{}^{\mathsf{sk}_{i,j}} = \mathsf{PK}_l$. Since the player does not know the keys $\mathsf{sk}_{i',j}$ for $i' \neq i$, he does not know the cards of the other players.

Play a card. To play the card id_l, the player i proves that the card id_l matches one of his key $\mathsf{pk}_{i,j}$ by showing that $h_l{}^{\mathsf{sk}_{i,j}} = \mathsf{PK}_l$. Note that since the player does not reveal $sk_{i,j}$, he can use the same set of public keys for different games. To prove that he cannot play any card of the leading suit, the player i sets L such that $l \in L$ if and only if id_l is not of the leading suit, then the player i proves in a zero-knowledge way that for all $\mathsf{pk}_{i,j}$ that correspond to cards that are not already played, there exists an (unrevealed) $l \in L$ such that $\log_{h_l}(\mathsf{PK}_l) = \log_g(\mathsf{pk}_{i,j})$. This implies that the player has no card of the leading suit, hence he is not cheating.

Definition 14. *$\mathsf{SecureSpades}$ is a Spades scheme defined as follows:*

$\mathsf{Init}(k)$: *It generates a group $\mathbb{G}$ of prime order q, a generator $g \in \mathbb{G}$ and returns $(\mathbb{G}, p, g)$.*

$\mathsf{KeyGen}(\mathsf{setup})$: *For all $i \in \llbracket 1, 13 \rrbracket$, it picks $\mathsf{sk}_i \xleftarrow{\$} \mathbb{Z}_q^*$ and computes $\mathsf{pk}_i = g^{\mathsf{sk}_i}$. It returns $\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_{13})$ and $\mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_{13})$.*

$\mathsf{GKeyGen}$: *It is a 4-party protocol, where for all $i \in \llbracket 1, 4 \rrbracket$ the i^{th} party is denoted P_i, and takes as input $(\mathsf{sk}_i, \{\mathsf{pk}_j\}_{1 \leq j \leq 4})$. This protocol returns a game public key PK, or the bottom symbol $\perp$. If there exist (i_1, j_1) and (i_2, j_2) such that $(i_1, j_1) \neq (i_2, j_2)$ and $\mathsf{pk}_{i_1,j_1} = \mathsf{pk}_{i_2,j_2}$, then the players abort and return $\perp$.*

- *For all $i \in \llbracket 1, 4 \rrbracket$, each player parses pk_i as $(\mathsf{pk}_{i,1}, \ldots, \mathsf{pk}_{i,13})$. For all $j \in \llbracket 1, 13 \rrbracket$, each player sets $h_{0,(i-1)\cdot 13+j} = g$ and $\mathsf{PK}_{0,(i-1)\cdot 13+j} = \mathsf{pk}_{i,j}$.*

- Each player P_i (for $i \in [\![1,4]\!]$) does the following step in turn: P_i picks $r = (r_1, \ldots, r_{52}) \xleftarrow{\$} (\mathbb{Z}_q^*)^{52}$, and a permutation δ on the set $[\![1,52]\!]$. P_i computes $h_{i,l} = h_{i-1,\delta(l)}^{r_l}$ and $\mathsf{PK}_{i,l} = (\mathsf{PK}_{i-1,\delta(l)})^{r_l}$ for all $l \in [\![1,52]\!]$, then runs $\Pi_i = \mathsf{ZK}\left\{(r,\delta) : \bigwedge_{l=1}^{52}\left(h_{i,l} = h_{i-1,\delta(l)}^{r_l} \wedge \mathsf{PK}_{i,l} = \mathsf{PK}_{i-1,\delta(l)}^{r_l}\right)\right\}$. This proof ensures that each $(h_{i,l}, \mathsf{PK}_{i,l})$ is the randomization of one pair $(h_{i-1,l'}, \mathsf{PK}_{i-1,l'})$ for $l' \in [\![1,52]\!]$. P_i broadcasts $\{(h_{i,l}, \mathsf{PK}_{i,l})\}_{1 \leq l \leq 52}$ and Π_i, then each player verifies the proof Π_i. If the verification fails, then the player aborts and returns $\perp$.
- If there exists j such that for all l, $h_{4,l}^{\mathsf{sk}_{i,j}} \neq \mathsf{PK}_{4,l}$, then P_i aborts the protocol and returns $\perp$. For each $i \in [\![1,4]\!]$, P_i sets $\mathsf{PK}'_i = ((h_{4,1}, \mathsf{PK}_{4,1}), \ldots, (h_{4,52}, \mathsf{PK}_{4,52}))$ and broadcasts it. If there exists i_1 and i_2 such that $\mathsf{PK}'_{i_1} \neq \mathsf{PK}'_{i_2}$, then P_i aborts and returns $\perp$, else P_i returns $\mathsf{PK} = \mathsf{PK}'_i$.

DeckGen: It is the 4-party Deck-random generator $\mathsf{RandGen}^{\mathsf{Deck}}$ protocol.

GetHand$(\mathsf{sk}, \mathsf{pk}, \mathsf{PK}, D)$: It parses sk as $(\mathsf{sk}_1, \ldots, \mathsf{sk}_{13})$, PK as $((h_1, \mathsf{PK}_1), \ldots, (h_{52}, \mathsf{PK}_{52}))$ and D as $(\mathsf{id}_1, \ldots, \mathsf{id}_{52})$. It returns the set H such that $\mathsf{id}_i \in H$ iff there exists $j \in [\![1,13]\!]$ such that $\mathsf{PK}_i = h_i^{\mathsf{sk}_j}$.

Play$(n, \mathsf{id}, \mathsf{sk}, \mathsf{pk}, \mathrm{st}, \mathsf{PK}, D)$: It parses D as $(\mathsf{id}_1, \ldots, \mathsf{id}_{52})$, sk as $(\mathsf{sk}_1, \ldots, \mathsf{sk}_{13})$, pk as $(\mathsf{pk}_1, \ldots, \mathsf{pk}_{13}))$, PK as $((h_1, \mathsf{PK}_1), \ldots, (h_{52}, \mathsf{PK}_{52}))$, and st as $(\alpha, \mathsf{suit}, U_1, U_2, U_3, U_4)$. If $\mathrm{st} = \perp$ it sets four empty sets U_1, U_2, U_3 and U_4. Let $v \in [\![1,52]\!]$ be the integer such that $\mathsf{id} = \mathsf{id}_v$ (i.e., v is the index of the played card id) and $t \in [\![1,13]\!]$ be the integer such that $\log_g(\mathsf{pk}_t) = \log_{h_v}(\mathsf{PK}_v)$ (i.e., t is the index of the public key that corresponds to the played card id). It sets $U'_n = U_n \cup \{t\}$. Note that at each step of the game, the set U_n contains the indexes of all the public keys of the user n that have already been used to play a card. For all $i \in [\![1,4]\!] \setminus \{n\}$, it sets $U'_i = U_i$.

If $\alpha = 4$ or $\mathrm{st} = \perp$ then it sets $\alpha' = 1$ and $\mathsf{suit}' = \mathsf{id}.\mathsf{suit}$. Else it sets $\alpha' = \alpha + 1$ and $\mathsf{suit}' = \mathsf{suit}$. The index α states how many players have already played this round, so if $\alpha = 4$, players start a new round. Moreover, suit states which suit is the leading suit of the round, given by the first card played in the round. This algorithm sets $\mathrm{st}' = (\alpha', \mathsf{suit}', U'_1, U'_2, U'_3, U'_4)$. It generates $\Pi_0 = \mathsf{ZK}\{(\mathsf{sk}_t) : \mathsf{pk}_t = g^{\mathsf{sk}_t} \wedge \mathsf{PK}_v = h_v^{\mathsf{sk}_t}\}$, which proves that the played card id_v matches one of the secret keys of the player. Let $L \in [\![1,52]\!]$ be a set such that for all $l \in L$, $\mathsf{suit}' \neq \mathsf{id}_l.\mathsf{suit}$, i.e., L is the set of the indexes of the cards that are not of the leading suit this round. For all $j \in [\![1,13]\!]$

- If $\mathsf{suit}' = \mathsf{id}.\mathsf{suit}$, it sets $\Pi_j = \perp$ (if the card id is of the leading suit, then the player can play it in any case, so no additional proof is required).
- If $j \in U_n$, it sets $\Pi_j = \perp$ (We omit the keys that have already been used in the previous rounds).
- If $j \notin U_n$ it generates $\Pi_j = \mathsf{ZK}\left\{(\mathsf{sk}_j) : \bigvee_{l \in L}(\mathsf{pk}_j = g^{\mathsf{sk}_j} \wedge \mathsf{PK}_l = h_l^{\mathsf{sk}_j})\right\}$. This proof ensures that the card that corresponds to each public key pk_j is not of the leading suit, which proves that the player n cannot play a card of the leading suit.

Finally, it returns the proof $\Pi = (t, \Pi_0, \ldots, \Pi_{13})$, and the updated value st'.

$\mathsf{Verif}(n, \mathsf{id}, \Pi, \mathsf{pk}, \mathrm{st}, \mathrm{st}', \mathsf{PK}, D)$: *It parses* st *as* $(\alpha, \mathsf{suit}, U_1, U_2, U_3, U_4)$*,* st' *as* $(\alpha', \mathsf{suit}', U'_1, U'_2, U'_3, U'_4)$*,* pk *as* $(\mathsf{pk}_1, \ldots, \mathsf{pk}_{13})$*, the key* PK *as* $((h_1, \mathsf{PK}_1), \ldots, (h_{52}, \mathsf{PK}_{52}))$*,* D *as* $(\mathsf{id}_1, \ldots, \mathsf{id}_{52})$ *and* Π *as* $(t, \Pi_0, \ldots, \Pi_{13})$*. If* $\mathrm{st} = \perp$*, it sets four empty sets* U_1*,* U_2*,* U_3 *and* U_4*. Let* v *be the integer such that* $\mathsf{id}_v = \mathsf{id}$ *(i.e.,* v *is the index of the played card* id*). Let* $L \in \llbracket 1, 52 \rrbracket$ *be a set such that for all* $l \in L$*,* $\mathsf{suit}' \neq \mathsf{id}_l.\mathsf{suit}$*, i.e.,* L *is the set of the indexes of the cards that are not of the leading suit. This algorithm first verifies that the state* st *is correctly updated in* st' *according to the* Play *algorithm:*

- *If there exists* $i \in \llbracket 1, 4 \rrbracket \setminus \{n\}$ *such that* $U'_i \neq U_i$*, then it returns* 0*.*
- *If* $t \in U_n$ *or* $U_n \cup \{t\} \neq U'_n$*, then it returns* 0*.*
- *If* $\alpha = 4$ *or* $\mathrm{st} = \perp$*, and* $\alpha' \neq 1$ *or* $\mathsf{suit}' \neq \mathsf{id}.\mathsf{suit}$*, then it returns* 0*.*
- *If* $\alpha \neq 4$ *and* $\mathsf{suit} \neq \perp$*, and* $\alpha' \neq \alpha + 1$ *or* $\mathsf{suit}' \neq \mathsf{suit}$*, then it returns* 0*.*

This algorithm then verifies the zero-knowledge proofs in order to check that the player does not cheat by playing a card he has not, or by playing a card that is not of the leading suit even though he could play a card of the leading suit.

- *If* Π_0 *is not valid then it returns* 0*.*
- *If* $\mathsf{suit}' \neq \mathsf{id}.\mathsf{suit}$ *and there exists an integer* $j \in \llbracket 1, n \rrbracket$ *such that* $j \notin U_n$ *and* Π_j *is not valid then it returns* 0*.*

If none of the previous checks fails, then this algorithm returns 1*.*

$\mathsf{GetSuit}(\mathrm{st})$: *It parses* st *as* $(\alpha, \mathsf{suit}, U_1, U_2, U_3, U_4)$ *and returns* suit*.*

Instantiation. We show how to instantiate the two zero-knowledge proofs of knowledge used in our protocol. The first one is a zero-knowledge OR-proof of the equality of two discrete logarithms denoted $\mathsf{ZK}\left\{(w) : \bigvee_{i=1}^{n} {a_i}^w = c_i \wedge {b_i}^w = d_i\right\}$. An efficient instantiation of such ZKPs in the random oracle model is given in [5]. Our protocol also uses a proof of correctness of a randomization of a set of shuffled commitments. This proof is denoted $\mathsf{ZK}\{((r_1, \ldots, r_n), \delta) : \bigwedge_{i=1}^{n} c_i = a_{\delta(i)}^{r_i} \wedge d_i = b_{\delta(i)}^{r_i}\}$, and can be instantiated using the previous one, since it consists in proving the equality of two discrete logarithms for the statement $\{(a_i, b_i, c_j, d_j)\}_{1 \leq j \leq n}$ for each j in $\llbracket 1, 52 \rrbracket$.

Security. We prove the security of our scheme in Theorem 1, then we give the intuition of the proof. The full proof is given in Appendix.

Theorem 1. *If the two proofs of knowledge are sound, extractable and zero-knowledge, then SecureSpades is theft-resistant, cheating-resistant, hand-private, unpredictable, and game-private under the DDH assumption in the ROM.*

Theft-resistant. To play a card, the player i must prove that the discrete logarithm of one of his public keys $\mathsf{pk}_{i,j}$ is equal to the discrete logarithm of the key PK_l that corresponds to the card. If the card is not in his hand, then none of the the discrete logarithms of the public keys $\mathsf{pk}_{i,j}$ is equal to the discrete logarithm of the key PK_l. Hence, to play a card that is not in his hand, the player should forge a proof of a false statement, which is not possible, since the proof system is sound.

Cheating-resistant. To play a card that is not of the leading suit, the player i must prove that the discrete logarithm of each public key $\mathsf{pk}_{i,j}$ is equal to the discrete logarithm of one key PK_l that corresponds to a card that is not of the leading suit. Hence, assuming that the player has some cards of the leading suit, in order to play another card, he should forge a proof of a false statement. This is not possible, since the proof system is sound.

Unpredictable. Since the deck D is chosen at random thanks to the protocol $\mathsf{RandGen}$, players have no way of guessing which card matches which public key during the keys generation phase.

Hand-private. Each player shuffles the keys PK_l using a secret permutation when he runs the $\mathsf{GKeyRound}$ algorithm. Moreover, the zero-knowledge proofs ensure that for each PK_l there exists a key $\mathsf{pk}_{i,j}$ such that $\log_{h_l}(\mathsf{PK}_l) = \log_g(\mathsf{pk}_{i,j})$. Guessing the hand of a player i is equivalent to guessing pairs (j, l) such that the key PK_l has the same discrete logarithm in basis h_l as the key $\mathsf{pk}_{i,j}$, which is equivalent to guessing whether PK_l is the Diffie-Hellman of h_l and $\mathsf{pk}_{i,j}$.

Game-private. During the game, the players use nothing other than zero-knowledge proofs, which leak nothing about the secret values of the players.

Other TTGs. Our Spades security model and scheme can be generalized to several trick-taking games. It works for any number of cards, of players, and for any team configuration. Moreover, it can be generalized to any game where players must play some kinds of cards according to a priority order, as long as the players can establish the set of all the cards that should be played (when it is possible) instead of the played one. This includes (but is not restricted to) all variants of Spades, Whist, Bridge, Belotte, Napoleon or Boston. Moreover, physical cards limit trick-taking games to games where players reveal all their cards, because if they do not, cheating could not be detected, even later. Our protocol allows the creation of new fair TTGs where players do not play all the cards of their hand.

8 Conclusion

In this paper, we have designed a secure protocol for trick-taking games. We used Spades, a famous online gambling card game, to illustrate our approach. Until now, such games required a trusted sever that ensures that players are not cheating. Our protocol allows the players to manage the game and detect cheating by themselves, without leaking any information about the hidden cards. Hence, a player cannot play a card that he does not have or that does not follow the rule of the game. Our construction is based on the discrete logarithm assumption and zero knowledge proofs. We proposed a security model and prove the security of our protocol.

In the future, we would like to implement a prototype, in order to evaluate the practical efficiency of our solution. Moreover, we would like to add secure payment distributions mechanism to our protocol. Another perspective is to try to generalize this approach to other games.

References

1. A. Barnett and N. P. Smart. Mental poker revisited. In *Cryptography and Coding, 9th IMA International Conference, Cirencester, UK, December 16-18, 2003, Proceedings*, volume 2898, pages 370–383. Springer, 2003.
2. I. Bentov, R. Kumaresan, and A. Miller. Instantaneous decentralized poker. In T. Takagi and T. Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 410–440, Cham, 2017. Springer International Publishing.
3. M. Blum. Coin flipping by telephone a protocol for solving impossible problems. *SIGACT News*, 15(1):23–27, Jan. 1983.
4. D. Boneh. The decision Diffie-Hellman problem. In *Third Algorithmic Number Theory Symposium (ANTS)*, LNCS. Springer, 1998. Invited paper.
5. X. Bultel and P. Lafourcade. Unlinkable and strongly accountable sanitizable signatures from verifiable ring signatures. In *CANS 2017*. LNCS. Springer, 2017.
6. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Advances in Cryptology — CRYPTO '97*, pages 410–424, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.
7. B. David, R. Dowsley, and M. Larangeira. Kaleidoscope: An efficient poker protocol with payment distribution and penalty enforcement. In *21st International Conference, FC 2018*, 2018.
8. B. David, R. Dowsley, and M. Larangeira. ROYALE: A framework for universally composable card games with financial rewards and penalties enforcement. *IACR Cryptology ePrint Archive*, 2018:157, 2018.
9. S. Goldwasser and S. Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, pages 365–377, New York, NY, USA, 1982. ACM.
10. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, Vol. 18(1), 1989.
11. V. Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. `https://eprint.iacr.org/2004/332`.
12. H. Stamer. Bibliography on mental poker. `https://www.nongnu.org/libtmcg/MentalPoker.pdf`.
13. T.-j. Wei. Secure and practical constant round mental poker. In *Information Sciences*, volume 273, pages 352–386, 07 2014.
14. J. Yan. Collusion detection in online bridge. In M. Fox and D. Poole, editors, *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*. AAAI Press, 2010.
15. W. Zhao, V. Varadharajan, and Y. Mu. A secure mental poker protocol over the internet. In C. Johnson, P. Montague, and C. Steketee, editors, *ACSW frontiers 2003*, Conferences in research and practice in information technology, pages 105–109, Australia, 2003. Australian Computer Society.

A Cryptographic Background

In this section, we give more details about the definitions of the DDH assumption and the zero-knowledge proofs.

Definition 15 (DDH [4]). *Let $\mathbb{G}$ be a multiplicative group of prime order q and $g \in \mathbb{G}$ be a generator. Given an instance (g^x, g^y, h_b) for unknown $(x, y) \xleftarrow{\$} (\mathbb{Z}_q^*)^2$ and $b \xleftarrow{\$} \{0,1\}$ such that $h_0 \xleftarrow{\$} \mathbb{G}$ and $h_1 = g^{x \cdot y}$, the* Decisional Diffie-Hellman *(DDH)* problem *is to guess b. The DDH* assumption *states that there exists no polynomial time algorithm that solves the DDH problem with a non-negligible advantage.*

A *Zero-Knowledge Proof of knowledge* (ZKP) [10] allows a prover knowing a witness w to convince a verifier that a statement s is in a given language without leaking any information. We recall the definition of a non-interactive zero-knowledge proof.

Definition 16 (NIZKP). *Let $\mathcal{R}$ be a binary relation and let $\mathcal{L}$ be a language such that $s \in \mathcal{L} \Leftrightarrow (\exists w, (s, w) \in \mathcal{R})$. A* non-interactive ZKP *(NIZKP) for the language $\mathcal{L}$ is a couple of algorithms* $(\mathsf{ZK}, \mathsf{Ver})$ *such that:*

$\mathsf{ZK}\{w : (s, w) \in \mathcal{R}\}$**.** *This algorithm outputs a proof π.*
$\mathsf{Ver}(s, \pi)$**.** *This algorithm outputs a bit b.*

A NIZKP proof has the following properties:

Soundness. *There is no polynomial time adversary $\mathcal{A}$ such that $\mathcal{A}(\mathcal{L})$ outputs (s, π) such that* $\mathsf{Ver}(s, \pi) = 1$ *and $s \notin \mathcal{L}$ with non-negligible probability.*

Extractability. *For all $s \in \mathcal{L}$, there exists an algorithm $\mathcal{E}$ (called an* extractor*) such that for any algorithm $\mathcal{A}$, the extractor $\mathcal{E}^{\mathcal{A}}(s)$ returns w such that $(s, w) \in \mathcal{R}$ with a similar running time than $\mathcal{A}$, and with at least the same probability as the probability that $\mathcal{A}$ returns π such that* $\mathsf{Ver}(s, \pi) = 1$.

Zero-knowledge. *A proof π* leaks no information, *i.e., there exists a polynomial time algorithm Sim (called the* simulator*) such that* $\mathsf{ZK}\{w : (s, w) \in \mathcal{R}\}$ *and* $\mathsf{Sim}(s)$ *follow the same probability distribution.*

B Proofs

B.1 Proof of Lemma 1

Proof. We recall Lemma 1: for any set $\mathcal{S}$ and any integer n, $\mathsf{RandGen}^{\mathcal{S}}_{P_1,\ldots,P_n}(k)$ is secure in the random oracle model. To prove this lemma, we use the game-based methodology [11]. We use the following sequence of games.

Game G_0: It is the real security game of the n-parties $\mathcal{S}$-random generator protocols.

Game G_1: It is defined as G_0 except that if $\mathcal{A}$ asks a query that contains the commitment of $\mathcal{C}$ (denoted $s_{\mathcal{C}}$) to the random oracle before than $\mathcal{C}$ reveals it, then $\mathcal{C}$ picks $s \xleftarrow{\$} \mathcal{S}$, returns it, and aborts. The probability that $\mathcal{C}$ aborts is the probability that $\mathcal{A}$ guesses $s_{\mathcal{C}}$ in q tries, where q is the number of calls to the random oracle, hence:

$$|\mathsf{Pr}[1 \leftarrow D(s) : s \xleftarrow{\$} G_0(k)] - \mathsf{Pr}[1 \leftarrow D(s) : s \xleftarrow{\$} G_1(k)]| \leq q/(2^k - q)$$

Game G_2: It is defined as G_1 except that if $\mathcal{A}$ asks x and x' to the random oracle such that $H(x) = H(x')$, then $\mathcal{C}$ picks $s \xleftarrow{\$} \mathcal{S}$, returns it, and aborts. The probability that $\mathcal{C}$ aborts is lower than the probability that $\mathcal{A}$ guesses x' such that $H(x) = H(x')$ given x in q tries, where q is the number of calls to the random oracle, hence:

$$|\mathsf{Pr}[1 \leftarrow D(s) : s \xleftarrow{\$} G_1(k)] - \mathsf{Pr}[1 \leftarrow D(s) : s \xleftarrow{\$} G_2(k)]| \leq q/2^k$$

Finally, $\mathcal{A}$ does not know any information about $s_{\mathcal{C}}$ before it chooses his commitments, and $\mathcal{A}$ cannot open wrongly one of its commitments by finding a collusion, hence it cannot guesses $r_0|| \dots ||r_n$ before the commitment of $\mathcal{C}$ is open, and it cannot change its own commitments. Since the value that $\mathcal{C}$ output is randomly generated by the random oracle, then the outputs of $\mathcal{C}$ is indistinguishable from a truly random generator. We deduce that:

$$\mathsf{Pr}[1 \leftarrow D(s) : s \xleftarrow{\$} G_2(k)] = \mathsf{Pr}[1 \leftarrow D(s) : s \xleftarrow{\$} \mathsf{RandGen}_{\mathcal{C},\mathcal{A}}(k)]$$

From previous results, we deduce:

$$|\mathsf{Pr}[1 \leftarrow D(s) : s \xleftarrow{\$} \mathcal{S}] - \mathsf{Pr}[1 \leftarrow D(s) : s \xleftarrow{\$} \mathsf{RandGen}_{\mathcal{C},\mathcal{A}}(k)]| \leq q/(2^k - q) + q/2^k$$

which concludes the proof. □

B.2 Proof of Theorem 1

The proof of Theorem 1 follows from the theorems of this section.

Theorem 2. *If SecureSpades is instantiated by two proofs of knowledge that are sound and zero-knowledge, then SecureSpades is theft-resistant.*

Proof. We claim that:

$$\mathsf{Pr}[\mathcal{A} \text{ wins}] \leq \epsilon_{\mathsf{sound}}(k)$$

We prove this claim by observing that if the adversary breaks the theft-resistance of $\mathsf{SecureSpades}$, then, at the end of the Spades experiment, $\exists\ \gamma \in [\![1, 13]\!]$ such that:

$$1 = \mathsf{Verif}(i_c, \mathsf{id}_{i_c,\gamma}, \mathit{\Pi}_{i_c,\gamma}, \mathsf{pk}_{i_c}, \mathrm{st}_\gamma, \mathrm{st}'_\gamma, \mathsf{PK}, D) \quad (1)$$

$$\forall\ \mathsf{id} \in H_{i_c}, \mathsf{id}_{i_c,\gamma} \neq \mathsf{id} \quad (2)$$

With non-negligible probability. Let $v \in [\![1, 52]\!]$ be the integer such that $\mathsf{id} = \mathsf{id}_v$. We parse $\mathit{\Pi}_{i_c,\gamma}$ as $(t, \mathit{\Pi}_0, \dots, \mathit{\Pi}_{13})$. Equation 1 implies:

$$1 = \mathsf{Ver}(\{(g, h_v, \mathsf{pk}_{i_c,t}, \mathsf{PK}_v)\}, \mathit{\Pi}_0).$$

We recall that if the experiment does not abort during the key generation phase, then all the keys $\mathsf{pk}_{i,j}$ are different, because if there exist $(i_1, i_2) \in [\![1, 4]\!]^2$ and $(j_1, j_2) \in [\![1, 13]\!]^2$ such that $(i_1, j_1) \neq (i_2, j_2)$ and $\mathsf{pk}_{i_1,j_1} = \mathsf{pk}_{i_2,j_2}$ then the algorithm $\mathsf{GKeyGen}$ returns $\perp$. Hence, Equation 2 implies that $\exists i \in [\![1, 4]\!] \backslash \{i_c\}$ and

$j \in [\![1, 13]\!]$ such that $\log_g(\mathsf{pk}_{i,j}) = \log_{h_v}(\mathsf{PK}_v)$, which implies that $\log_g(\mathsf{pk}_{i_c,t}) \neq \log_{h_v}(\mathsf{PK}_v)$. We deduce that if $\mathcal{A}$ wins, then during the experiment it generates a proof Π_0 such that $1 = \mathsf{Ver}(\{(g, h_v, \mathsf{pk}_{i_c,t}, \mathsf{PK}_v)\}, \Pi_0)$ and $\log_g(\mathsf{pk}_{i_c,t}) \neq \log_{h_v}(\mathsf{PK}_v)$, which is a proof of a false statement, which concludes the proof. □

Theorem 3. *If* SecureSpades *is instantiated by two proofs of knowledge that are sound and zero-knowledge, then* SecureSpades *is cheating-resistant.*

Proof. We claim that:

$$\mathsf{Pr}[\mathcal{A} \text{ wins}] \leq 2 \cdot \epsilon_{\mathsf{sound}}(k).$$

We prove this claim by observing that if the adversary breaks the cheating-resistance of SecureSpades, then $\exists\ \gamma \in [\![1, 13]\!]$ such that:

1. $1 = \mathsf{Verif}(i_c, \mathsf{id}_{i_c,\gamma}, \Pi_{i_c,\gamma}, \mathsf{pk}_{i_c}, \mathrm{st}_\gamma, \mathrm{st}'_\gamma, \mathsf{PK}, D)$,
2. $\mathsf{id}_{i_c,\gamma}.\mathsf{suit} \neq \mathsf{suit}_{i_c,\gamma}$ and $\mathsf{suit}_{i_c,\gamma} \neq \perp$
3. $\exists\ \bar{\mathsf{id}} \in H_{i_c}$ such that:
   (a) $\forall\ l \leq \gamma, \mathsf{id}_{i_c,l} \neq \bar{\mathsf{id}}$
   (b) $\bar{\mathsf{id}}.\mathsf{suit} = \mathsf{suit}_{i_c,\gamma}$

with non-negligible probability. We distinguish two cases:

– The adversary forges a proof of a false statement during the GKeyGen protocol. In this case, to win the experiment, the adversary $\mathcal{A}$ must produce a valid proof of a false statement, hence $\mathsf{Pr}[\mathcal{A} \text{ wins}|\text{case } 1] \leq \epsilon_{\mathsf{sound}}(k)$.
– The adversary does not forge a proof of a false statement during the GKeyGen protocol. In this case, for all $v \in [\![1, 52]\!]$, there exist $i \in [\![1, 4]\!]$ and $j \in [\![1, 13]\!]$ such that $\log_g(\mathsf{pk}_{i,j}) = \log_{h_v}(\mathsf{PK}_v)$. In the following, we show that in this case, $\mathsf{Pr}[\mathcal{A} \text{ wins}|\text{case } 2] \leq \epsilon_{\mathsf{sound}}(k)$. The two cases imply that $\mathsf{Pr}[\mathcal{A} \text{ wins}] \leq 2 \cdot \epsilon_{\mathsf{sound}}(k)$.

Assume that $\mathcal{A}$ wins the experiment. We recall that if the experiment does not abort during the key generation phase, then all the keys $\mathsf{pk}_{i,j}$ are different. Let $L \subset [\![1, 52]\!]$ be the set such that for all $l \in L$, $\mathsf{suit}_{i_c,\gamma} \neq \mathsf{id}_l.\mathsf{suit}$. We parse $\Pi_{i_c,\gamma}$ as $(t, \Pi_0, \ldots, \Pi_{13})$. Items 1, 2 and 3 imply: $\forall j \in [\![1, 13]\!]$ such that $\Pi_j \neq \perp$, $1 = \mathsf{Ver}(\{(g, h_l, \mathsf{pk}_{i_c,j}, \mathsf{PK}_l)\}_{l \in L}, \Pi_j)$. Let v be such that $\bar{\mathsf{id}} = \mathsf{id}_v$. We remark that $v \notin L$ from (3a). Moreover, since $\mathsf{id}_v \in H_{i_c}$, there exists j such that $\log_g(\mathsf{pk}_{i_c,j}) = \log_{h_v}(\mathsf{PK}_v)$. We deduce that $\forall l \in L, \log_g(\mathsf{pk}_{i_c,j}) \neq \log_{h_l}(\mathsf{PK}_l)$, which implies that Π_j is a valid proof of a false statement, hence $\mathcal{A}$ breaks the soundness of the proof. This concludes the proof. □

Theorem 4. *If* SecureSpades *is instantiated by two proofs of knowledge that are sound and zero-knowledge and by a secure multi-party random generator, then* SecureSpades *is unpredictable.*

Proof. We define G_0 as the real unpredictability security experiment, and G_1 as the same game as G_0 except that the deck D is chosen at random in Deck by the challenger $\mathcal{C}$. Let $\epsilon_{\mathsf{RG}}(k)$ be the advantage on the security of RandGen. We have:

$$|\mathsf{Pr}[1 \leftarrow G_0(k)] - \mathsf{Pr}[1 \leftarrow G_1(k)]| \leq \epsilon_{\mathsf{RG}}(k).$$

Let $\mathcal{A}$ be an adversary that plays the game G_1. Assume that the adversary does not forge a proof of a false statement during the $\mathsf{GKeyGen}$ protocol. In this case, if $b = 1$ and $\mathsf{PK} \neq \perp$, then for all $v \in [\![1, 52]\!]$, there exist $i \in [\![1, 4]\!]$ and $j \in [\![1, 13]\!]$ such that $\log_g(\mathsf{pk}_{i,j}) = \log_{h_v}(\mathsf{PK}_v)$. Since $D = (\mathsf{id}_1, \ldots, \mathsf{id}_{52})$ is chosen at random, for all $i \in [\![1, 52]\!]$ the probability that $\mathsf{id}_i \in H_{i_c}$ is $1/4$. Hence to have a non-null advantage, the adversary $\mathcal{A}$ must produce a valid proof of a false statement, hence the advantage of $\mathcal{A}$ is lower than $\epsilon_{\mathsf{sound}}(k)$:

$$|\mathsf{Pr}[1 \leftarrow G_1(k)] - 1/4| \leq \epsilon_{\mathsf{sound}}(k).$$

Finally:

$$|\mathsf{Pr}[1 \leftarrow G_0(k)] - 1/4| \leq \epsilon_{\mathsf{RG}}(k) + \epsilon_{\mathsf{sound}}(k)$$

which is negligible. This concludes the proof. □

Definition 17 (n-DDH). *Let $\mathbb{G}$ be a multiplicative group of prime order q and $g \in \mathbb{G}$ be a generator. Given an instance $\left\{(g^{a_i}, g^{b_i}, h_{i,b})\right\}_{1 \leq i \leq n}$ such that for all $i \in [\![1, n]\!]$, $(a_i, b_i) \xleftarrow{\$} (\mathbb{Z}_q^*)^2$ and $b \xleftarrow{\$} \{0, 1\}$ such that $h_{i,0} \xleftarrow{\$} \mathbb{G}$ and $h_{i,1} = g^{a_i \cdot b_i}$, the n-*Decisional Diffie-Hellman *(n-DDH)* problem *is to guess b. The n-DDH* assumption *states that there exists no polynomial time algorithm that solves the n-DDH problem with a non-negligible advantage.*

Lemma 2. *For any $n \in \mathbb{N}$, n-DDH holds under the DDH assumption.*

Proof. We use an hybrid argument. Consider the following problem:

(j, n)-DDH problem : Let $\mathbb{G}$ be a multiplicative group of prime order q and $g \in \mathbb{G}$ be a generator. Let $j \in \mathbb{N}$ be such that $0 \leq j \leq n$. Given an instance $\left\{(g^{a_i}, g^{b_i}, h_{i,b})\right\}_{1 \leq i \leq n}$ such that for all $i \in [\![1, n]\!]$, $(a_i, b_i) \xleftarrow{\$} (\mathbb{Z}_q^*)^2$ and $b \xleftarrow{\$} \{0, 1\}$ such that:

- if $i \leq j$, $h_{i,0} \xleftarrow{\$} \mathbb{G}$ and $h_{i,1} = g^{a_i \cdot b_i}$
- else, $h_{i,1} \xleftarrow{\$} \mathbb{G}$ and $h_{i,0} = g^{a_i \cdot b_i}$

the (j, n)-*Decisional Diffie-Hellman* ((j, n)-DDH) *problem* is to guess b.

Let $\mathsf{Adv}^{(j,n)\text{-}\mathsf{DDH}}(k)$ (resp. $\mathsf{Adv}^{n\text{-}\mathsf{DDH}}(k)$, $\mathsf{Adv}^{\mathsf{DDH}}(k)$) be the advantage of the best algorithm that solves the (j, n)-DDH (resp. n-DDH, DDH) problem. Let (j, n) be a couple of positive integers such that $0 \leq j \leq n - 1$. For any adversary that solves the (j, n)-DDH problem with advantage $\mathsf{Adv}_{\mathcal{A}}^{(j,n)\text{-}\mathsf{DDH}}(k)$, we build the algorithm $\mathcal{B}$ that tries to solve the DDH problem.

algorithm $\mathcal{B}(g_1, g_2, h)$: This algorithm picks $b' \xleftarrow{\$} \{0, 1\}$, then for all $i \in [\![1, n]\!] \backslash \{j + 1\}$ it picks $(a_i, b_i) \xleftarrow{\$} (\mathbb{Z}_q^*)^2$, sets $g_{i,1} = g^{a_i}$, $g_{i,2} = g^{b_i}$:

- if $i \leq j$, it picks $h_{i,0} \xleftarrow{\$} \mathbb{G}$ and sets $h_{i,1} = g^{a_i \cdot b_i}$
- else, $h_{i,1} \xleftarrow{\$} \mathbb{G}$ and $h_{i,0} = g^{a_i \cdot b_i}$.

It sets $g_{j+1,1} = g_1$, $g_{j+1,2} = g_2$, and $h_{j+1,1} = h_{j+1,0} = h$. It runs $b_* \xleftarrow{\$} \mathcal{A}(\{(g_{i,1}, g_{i,2}, h_{i,b})\}_{1 \leq i \leq n})$ and returns b_*.

We then deduce that:

$$\mathsf{Adv}_{\mathcal{B}}^{\mathsf{DDH}}(k) = \left| \mathsf{Adv}_{\mathcal{A}}^{(j,n)\text{-}\mathsf{DDH}}(k) - \mathsf{Adv}_{\mathcal{A}}^{(j+1,n)\text{-}\mathsf{DDH}}(k) \right| .$$

Hence,

$$\mathsf{Adv}^{\mathsf{DDH}}(k) \geq \left| \mathsf{Adv}^{(j,n)\text{-}\mathsf{DDH}}(k) - \mathsf{Adv}^{(j+1,n)\text{-}\mathsf{DDH}}(k) \right|$$

which implies

$$n \cdot \mathsf{Adv}^{\mathsf{DDH}}(k) \geq \mathsf{Adv}^{n\text{-}\mathsf{DDH}}(k).$$

This concludes the proof. □

Theorem 5. *$\mathsf{SecureSpades}$ is hand-private under the 52-DDH assumption.*

Proof. Assume that there exists an adversary $\mathcal{A}$ that breaks the hand-privacy of $\mathsf{SecureSpades}$ with advantage $\lambda(k)$. We show how to build an adversary $\mathcal{B}$ that solves the 52-DDH problem in $(\mathbb{G}, q, g)$ with a non-negligible advantage.

Construction of $\mathcal{B}\left(\left\{(\hat{h}_l, \hat{\mathsf{pk}}_l, \hat{\mathsf{PK}}_l)\right\}_{1 \leq l \leq 52}\right)$**:**

Key generation phase: This algorithm sets st $=\perp$. It sends $((\mathbb{G}, q, g), \mathrm{st})$ to $\mathcal{A}$. For all $i \in \llbracket 1, 4 \rrbracket \setminus \{i_c\}$ and $j \in \llbracket 1, 13 \rrbracket$, this algorithm sets $\mathsf{pk}_{i,j} = \hat{\mathsf{pk}}_{(i-1)\cdot 13+j}$ and $\mathsf{pk}_i = (\mathsf{pk}_{i,1}, \ldots, \mathsf{pk}_{i,13})$, then it sends pk_i to $\mathcal{A}$. Finally $\mathcal{A}$ returns $\mathsf{pk}_{i_c} = (\mathsf{pk}_{i_c,1}, \ldots, \mathsf{pk}_{i_c,13})$.

Game key generation phase: If there exist (i_1, j_1) and (i_2, j_2) such that $(i_1, j_1) \neq (i_2, j_2)$ and $\mathsf{pk}_{i_1,j_1} = \mathsf{pk}_{i_2,j_2}$, then $\mathcal{C}$ aborts the experiment. For all $j \in \llbracket 1, 13 \rrbracket$, $\mathcal{B}$ sets $h_{0,(i-1)\cdot 13+j} = g$ and $\mathsf{PK}_{0,(i-1)\cdot 13+j} = \mathsf{pk}_{i,j}$. For all $i \in \llbracket 1, 4 \rrbracket$. We distinguish two cases:

Case 1 ($i_c = 4$)**:** For all $i \in \llbracket 1, 4 \rrbracket$:

- If $i = 1$, then $\mathcal{B}$ picks a vector $r_1 = (r_{1,1}, \ldots, r_{1,52}) \xleftarrow{\$} (\mathbb{Z}_q^*)^{52}$, and a permutation δ_1 on the set $\llbracket 1, 52 \rrbracket$. For all $l \in \llbracket 1, 52 \rrbracket$:
  - If $\mathsf{PK}_{0,l} = \hat{\mathsf{pk}}_l$, then $\mathcal{B}$ sets $\tilde{h}_{1,l} = \hat{h}_l$ and $\tilde{PK}_{1,l} = \hat{\mathsf{PK}}_l$.
  - Else it sets $\tilde{h}_{1,l} = h_{0,l}^{r_{1,l}}$ and $\tilde{\mathsf{PK}}_{1,l} = \mathsf{PK}_{0,l}^{r_{1,l}}$.

  Finally $\mathcal{C}$ sets $h_{1,l} = \tilde{h}_{1,\delta_1(l)}$ and $\mathsf{PK}_{1,l} = \tilde{PK}_{1,\delta_1(l)}$, and generates the proof $\mathit{\Pi}_1$ by running the simulator of the proof of knowledge of randomization of a set of commitments. $\mathcal{B}$ sends $\{(h_{1,l}, \mathsf{PK}_{1,l})\}_{1 \leq l \leq 52}$ and $\mathit{\Pi}_1$ to $\mathcal{D}_3$
- If $i \neq 1$ and $i \neq 4$, then $\mathcal{B}$ processes as in the real protocol.
- If $i = 4$, it receives $\{(h_{4,l}, \mathsf{PK}_{4,l})\}_{1 \leq l \leq 52}$ and $\mathit{\Pi}_4$ from $\mathcal{D}_3$, then it uses the knowledge extractor of the proof of knowledge of randomization of a set of commitments (which is extractable by hypothesis) to extract the function δ_4. If this extraction fails, then it aborts the experiment and returns a random bit. If the proof $\mathit{\Pi}_4$ is not valid then $\mathcal{B}$ aborts the experiment and returns a random bit.

$\mathcal{B}$ computes $\mathsf{PK}^*_{i_c,l} = \left((\mathsf{PK}_{i_c,l})^{\prod_{j=2}^{i_c} \frac{1}{r_{j,\delta_{j+1}(\ldots\delta_{i_c}(l)\ldots)}}} \right)$.

If $\{\hat{\mathsf{PK}}_{(i-1)\cdot 13+j}\}_{1\leq i\leq 4;1\leq j\leq 13;i\neq i_c} \not\subset \{\mathsf{PK}^*_{i_c,l}\}_{1\leq l\leq 52}$, then $\mathcal{B}$ aborts the experiment and returns a random bit, because the adversary does not correctly shuffle the commitments of the honest players.
Case 2 $(i_c \neq 4)$**:** For all $i \in [\![1, 4]\!]$:

- If $i \neq i_c$ and $i \neq i_c + 1$, then $\mathcal{B}$ processes as in the real protocol.
- If $i = i_c$, it receives $\{(h_{i,l}, \mathsf{PK}_{i,l})\}_{1\leq l\leq 52}$ and Π_i from $\mathcal{D}_3$, then it uses the knowledge extractor of the proof of knowledge of randomization of a set of commitments (which is extractable by hypothesis) to extract the function δ_4 and the vector r_i. If this extraction fails, it aborts the experiment and returns a random bit. If the proof Π_4 is not valid then $\mathcal{B}$ aborts the experiment and returns a random bit.
- If $i = i_c + 1$, then $\mathcal{B}$ picks a vector $r_i = (r_{i,1}, \ldots, r_{i,52}) \xleftarrow{\$} (\mathbb{Z}_q^*)^{52}$, and a permutation δ_i on the set $[\![1, 52]\!]$. For all $l \in [\![1, 52]\!]$, it computes $\mathsf{PK}^*_{i-1,l} = \left((\mathsf{PK}_{i-1,l})^{\prod_{j=1}^{i-1} \frac{1}{r_{j,\delta_{j+1}(\ldots\delta_{i-1}(l)\ldots)}}} \right)$ then:
  - If there exists $l' \in [\![1, 52]\!]$ such that $\mathsf{PK}^*_{i-1,l} = \hat{\mathsf{pk}}_{l'}$, then $\mathcal{B}$ sets $\tilde{h}_{i,l} = \hat{h}_{l'}$ and $\tilde{PK}_{i,l} = \hat{\mathsf{PK}}_{l'}$.
  - Else it sets $\tilde{h}_{i,l} = h^{r_{i,l}}_{i-1,l}$ and $\tilde{\mathsf{PK}}_{i,l} = \mathsf{PK}^{r_{i,l}}_{i-1,l}$.

  Finally $\mathcal{C}$ sets $h_{i,l} = \tilde{h}_{i,\delta_i(l)}$ and $\mathsf{PK}_{i,l} = \tilde{PK}_{i,\delta_i(l)}$, and generates the proof Π_i by running the simulator of the proof of knowledge of randomization of a set of commitments. $\mathcal{B}$ sends $\{(h_{i,l}, \mathsf{PK}_{i,l})\}_{1\leq l\leq 52}$ and Π_i to $\mathcal{D}_3$

$\mathcal{B}$ computes $\mathsf{PK}^*_{i_c,l} = \left((\mathsf{PK}_{i_c,l})^{\prod_{j=1}^{i_c} \frac{1}{r_{j,\delta_{j+1}(\ldots\delta_{i_c}(l)\ldots)}}} \right)$.

If $\{\mathsf{pk}_{i,j}\}_{1\leq i\leq 4;1\leq j\leq 13;i\neq i_c} \not\subset \{\mathsf{PK}^*_{i_c,l}\}_{1\leq l\leq 52}$, then $\mathcal{B}$ aborts the experiment and returns a random bit, because the adversary does not correctly shuffle the commitments of the honest players.

Finally, in all cases $\mathcal{B}$ sets $\delta = \delta_4 \circ \delta_3 \circ \delta_2 \circ \delta_1$, and for all $l \in [\![1, 52]\!]$ it sets $h_l = h_{4,l}$, $\mathsf{PK}_l = \mathsf{PK}_{4,l}$, and $\mathsf{PK} = ((h_1, \mathsf{PK}_1), \ldots, (h_{52}, \mathsf{PK}_{52}))$.

Shuffle phase: $\mathcal{A}$ sends a deck $D \in \mathsf{Decks}$ to $\mathcal{B}$, which parse D as $(\mathsf{id}_1, \ldots, \mathsf{id}_{52})$. For all $i \in [\![1, 4]\!] \setminus \{i_c\}$, $\mathcal{B}$ computes $H_i = (\mathsf{id}_{\delta((i-1)\cdot 13+1)}, \cdots, \mathsf{id}_{\delta((i-1)\cdot 13+13)})$.

Challenge phase: $\mathcal{B}$ picks (θ_0, θ_1) in $(i \in [\![1, 4]\!] \setminus \{i_c\})^2$ such that $\theta_0 \neq \theta_1$. $\mathcal{B}$ picks $b' \xleftarrow{\$} \{0, 1\}$ and $\bar{\mathsf{id}} \in H_{\theta_{b'}}$. It sends $(\bar{id}, \theta_0, \theta_1)$ to $\mathcal{A}$, which returns b'_*.

Final phase: If $b'_* = b'$, then $\mathcal{B}$ returns 1, else it returns 0.

We distinguish two cases :

- The adversary forges a proof of a false statement during the $\mathsf{GKeyGen}$ protocol. In this case, if the adversary $\mathcal{A}$ does not produce a valid proof for the false statement, $\mathsf{PK} = \perp$, then the experiment aborts, hence the advantage of $\mathcal{B}$ is lower than $\epsilon_{\mathsf{sound}}(k)$.

- The adversary does not forge a proof of a false statement during the $\mathsf{GKeyGen}$ protocol. In this case, if $b = 1$ and $\mathsf{PK} \neq \perp$, then for all $v \in [\![1, 52]\!]$, there exist $i \in [\![1, 4]\!]$ and $j \in [\![1, 13]\!]$ such that $\log_g(\mathsf{pk}_{i,j}) = \log_{h_v}(\mathsf{PK}_v)$. In the following, we show that in this case, the advantage of $\mathcal{B}$ is lower than $2 \cdot \epsilon_{\text{52-DDH}}(k)$. If $b = 1$, then the experiment is perfectly simulated, else, there is no discrete logarithms equality between the keys $\mathsf{pk}_{i,j}$ and the keys PK_v for $i \in [\![1, 4]\!] \setminus \{i_c\}$, $j \in [\![1, 13]\!]$ and $v \in [\![1, 52]\!]$. In this case the best strategy for $\mathcal{A}$ is to guess b at random. We observe that:

$$\begin{aligned} \mathsf{Pr}[\mathcal{B} \text{ wins}] &= \frac{1}{2} \cdot \mathsf{Pr}[\mathcal{A} \text{ wins}|b = 0] + \frac{1}{2} \cdot \mathsf{Pr}[\mathcal{A} \text{ wins}|b = 1] \\ &= \frac{1}{2} \cdot \left(\frac{1}{2} + \mathsf{Pr}[\mathcal{A} \text{ wins}|b = 1] \right). \end{aligned}$$

  Hence:

$$\left| \frac{1}{2} - \mathsf{Pr}[\mathcal{B} \text{ wins}] \right| = \frac{1}{2} \cdot \left| \frac{1}{2} - \mathsf{Pr}[\mathcal{A} \text{ wins}|b = 1] \right| = \frac{\lambda(k)}{2}$$

  which is non-negligible.

The two cases imply that the advantage of $\mathcal{B}$ is lower than $2 \cdot \epsilon_{\text{52-DDH}}(k) + \epsilon_{\text{sound}}(k)$, which concludes the proof. □

Definition 18 (n-ShDDH). *Let $\mathbb{G}$ be a multiplicative group of prime order q and $g \in \mathbb{G}$ be a generator. Given an instance $\left\{(g^{a_i}, g^{b_i}, h_{i,b})\right\}_{1 \leq i \leq n}$ such that for all $i \in [\![1, n]\!]$, $(a_i, b_i) \xleftarrow{\$} (\mathbb{Z}_q^*)^2$, $b \xleftarrow{\$} \{0, 1\}$, $h_i = g^{a_i \cdot b_i}$, δ a permutation on the set $[\![1, n]\!]$, $h_{i,0} = h_{\delta(i)}$ and $h_{i,1} = h_i$, the* n-Shuffled Decisional Diffie-Hellman *(n-ShDDH)* problem *is to guess b. The n-ShDDH* assumption *states that there exists no polynomial time algorithm that solves the n-ShDDH problem with a non-negligible advantage.*

Proof. For any integer n, assume that there exists an algorithm $\mathcal{A}$ that solves the n-ShDDH assumption with a non-negligible advantage $\lambda(k)$. We show how to build an algorithm $\mathcal{B}$ that solves the the n-DDH assumption with a non-negligible advantage.

$\mathcal{B}(\{(g_{i,1}, g_{i,2}, h_i)\}_{1 \leq i \leq n})$**:** $\mathcal{B}$ picks $b' \xleftarrow{\$} \{0, 1\}$ and δ a permutation on the set $[\![1, n]\!]$ at random.

- If $b' = 1$, it runs $b'_* \leftarrow \mathcal{A}(\{(g_{i,1}, g_{i,2}, h_i)\}_{1 \leq i \leq n})$.
- If $b' = 0$, it runs $b'_* \leftarrow \mathcal{A}(\left\{(g_{i,1}, g_{i,2}, h_{\delta(i)})\right\}_{1 \leq i \leq n})$.

Finally, $\mathcal{B}$ if $b' = b'_*$ then $\mathcal{B}$ returns 1, else it returns 0.

We observe that:

$$\begin{aligned} \mathsf{Pr}[\mathcal{B} \text{ wins}] &= \frac{1}{2} \cdot \mathsf{Pr}[\mathcal{A} \text{ wins}|b = 0] + \frac{1}{2} \cdot \mathsf{Pr}[\mathcal{A} \text{ wins}|b = 1] \\ &= \frac{1}{2} \cdot \left(\frac{1}{2} + \mathsf{Pr}[\mathcal{A} \text{ wins}|b = 1] \right). \end{aligned}$$

Hence:

$$\left|\frac{1}{2} - \mathsf{Pr}[\mathcal{B} \text{ wins}]\right| = \frac{1}{2} \cdot \left|\frac{1}{2} - \mathsf{Pr}[\mathcal{A} \text{ wins}|b = 1]\right| = \frac{\lambda(k)}{2}$$

which is non-negligible. This concludes the proof. □

Theorem 6. *If* SecureSpades *is instantiated by two proofs of knowledge that are sound, extractable and zero-knowledge, then* SecureSpades *is private under the* DDH *assumption.*

Proof. We build the following simulator:

Construction of $\mathsf{Sim}^{\mathsf{Spades}}_{\mathsf{SecureSpade},S,\mathcal{A}(\mathsf{vw})}(k, \mathit{setup}, s, \{\mathsf{pk}_i\}_{1\leq i\leq 4}, \mathsf{PK}, \mathsf{vw})$*:*

Key generation phase: The simulator parses:
- For all $i \in \{1, 2, 3, 4\}$, pk_i as $(\mathsf{pk}_{i,1}, \ldots, \mathsf{pk}_{i,13})$
- It deduces sk_{i_c} from vw and parses it as $(\mathsf{sk}_{i_c,1}, \ldots, \mathsf{sk}_{i_c,13})$ such that for all $j \in [\![1, 13]\!]$, $g^{\mathsf{sk}_{i_c,j}} = \mathsf{pk}_{i_c,j}$. The simulator does not abort at this point even if it cannot find sk_{i_c} correctly.

It sets $\mathrm{st} =\perp$. It sends $(\mathsf{setup}, \mathrm{st})$ to $\mathcal{A}$, then for all $i \in \{1, 2, 3, 4\}$, it sends pk_i to $\mathcal{A}$.

Deck key generation phase: The simulator parses PK as $((h_1, \mathsf{PK}_1), \ldots, (h_{52}, \mathsf{PK}_{52}))$. If $\mathsf{PK} =\perp$ then it aborts the experiment and returns 0, else it sends PK to $\mathcal{A}$.

Shuffle phase: $\mathcal{A}$ sends a deck $D \in \mathsf{Decks}$ and a first player index p_* to the simulator, which sets $p = p_*$ and parses D as $(\mathsf{id}_1, \ldots, \mathsf{id}_{52})$. It computes $H_{i_c} \leftarrow \mathsf{GetHand}(\mathsf{sk}_{i_c}, \mathsf{pk}_{i_c}, \mathsf{PK}, D)$. For all $i \in \{1, 2, 3, 4\} \setminus \{i_c\}$, the simulator picks H_i at random such that $|H_i| = 13$ and $H_i \subseteq \{\mathsf{id}_l\}_{1\leq l\leq 52} \setminus (H_{i_c} \cup (\cup_{j=1,j\neq i_c}^{i-1} H_j)$.

Game phase: The simulator sets $\gamma = 0$ and $\mathsf{played} =\perp$. For all $i \in [\![1, 52]\!]$:

If $p \neq i_c$: The simulator runs $\mathsf{id} \leftarrow \mathsf{Strat}_p(\mathsf{played}, H_p, p_*, p)$, it parses st as $(\alpha, \mathsf{suit}, U_1, U_2, U_3, U_4)$ and U_p as $(u_1, \ldots, u_{13})$, then it processes as the algorithm $(\mathit{\Pi}, \mathrm{st}') \leftarrow \mathsf{Play}(p, \mathsf{id}, \mathsf{sk}_p, \mathsf{pk}_p, \mathrm{st}, \mathsf{PK}, D)$ except that:
- It picks t at random in the set $\{t \in [\![1, 13]\!] : u_t = 0\}$
- We recall that there exists a polynomial time simulator Sim that perfectly simulates the proof algorithm of the proof of knowledge of discrete logarithms equality because it is *zero-knowledge*. It computes $\mathit{\Pi}_0$ using the simulator Sim as follows: $\mathit{\Pi}_0 \leftarrow \mathsf{Sim}(\{(g, h_v, \mathsf{pk}_t, \mathsf{PK}_v)\})$.
- It computes $\mathit{\Pi}_j$ for all $j \in [\![1, 13]\!]$ using the simulator Sim as follows: let $L \in [\![1, 52]\!]$ be a set such that for all $l \in L$, $\mathsf{suit}' \neq \mathsf{id}_l.\mathsf{suit}$. For all $j \in [\![1, 13]\!]$
  - If $\mathsf{suit}' = \mathsf{id}.\mathsf{suit}$, it sets $\mathit{\Pi}_j =\perp$.
  - If $u_j = 1$, it sets $\mathit{\Pi}_j =\perp$.
  - If $u_j = 0$ it generates $\mathit{\Pi}_j \leftarrow \mathsf{Sim}(\{(g, h_l, \mathsf{pk}_j, \mathsf{PK}_l)\}_{l\in L})$.

It sends $(\mathsf{id}, \Pi, \mathrm{st}')$ to $\mathcal{A}$ and updates $\mathrm{st} := \mathrm{st}'$. Finally, it updates the index p that points the next player according to the rule of Spades. It then parses played as $(\mathsf{pl}_1, \dots, \mathsf{pl}_n)$ (where $n = |\mathsf{played}|$) and updates $\mathsf{played} := (\mathsf{pl}_1, \dots, \mathsf{pl}_n, \mathsf{id})$.

If $p = i_c$**:** The simulator processes as in $\mathsf{Exp}^{\mathsf{Spades}}_{\mathsf{SecureSpade},S,K,\mathcal{A}}(k)$. Note that the simulation may fail if it does not know sk_{i_c}.

Assume that there exists $\mathcal{D}$ such that $|P_{\mathsf{real}}(\mathcal{D}, k) - P_{\mathsf{sim}}(\mathcal{D}, k)| = \lambda(k)$ where λ is non-negligible. We show how to build an algorithm $\mathcal{B}$ that solves the 39-ShDDH with non negligible advantage.

Construction of $\mathcal{B}\left(\left\{(\hat{h}_{l'}, \hat{\mathsf{pk}}_{l'}, \hat{\mathsf{PK}}_{l'})\right\}_{1 \le l' \le 39}\right)$*:*

Key generation phase: This algorithm sets $\mathrm{st} = \perp$. It sends $((\mathbb{G}, q, g), \mathrm{st})$ to $\mathcal{D}_1$ which returns i_c. For all $i \in [\![1, 4]\!] \setminus \{i_c\}$ and $j \in [\![1, 13]\!]$, this algorithm sets:

- if $i < i_c$, $\mathsf{pk}_{i,j} = \hat{\mathsf{pk}}_{(i-1)\cdot 13+j}$,
- if $i > i_c$, $\mathsf{pk}_{i,j} = \hat{\mathsf{pk}}_{(i-2)\cdot 13+j}$.

It sets $\mathsf{pk}_i = (\mathsf{pk}_{i,1}, \dots, \mathsf{pk}_{i,13})$, then it sends pk_i to $\mathcal{D}_2$, which returns $\mathsf{pk}_{i_c} = (\mathsf{pk}_{i_c,1}, \dots, \mathsf{pk}_{i_c,13})$.

Game key generation phase: If there exist (i_1, j_1) and (i_2, j_2) such that $(i_1, j_1) \neq (i_2, j_2)$ and $\mathsf{pk}_{i_1,j_1} = \mathsf{pk}_{i_2,j_2}$, then $\mathcal{C}$ aborts the experiment. For all $j \in [\![1, 13]\!]$, $\mathcal{B}$ sets $h_{0,(i-1)\cdot 13+j} = g$ and $\mathsf{PK}_{0,(i-1)\cdot 13+j} = \mathsf{pk}_{i,j}$. For all $i \in [\![1, 4]\!]$. We distinguish two cases:

Case 1 $(i_c = 4)$**:** For all $i \in [\![1, 4]\!]$:

- If $i = 1$, then $\mathcal{B}$ picks a vector $r_1 = (r_{1,1}, \dots, r_{1,52}) \xleftarrow{\$} (\mathbb{Z}_q^*)^{52}$, and a permutation δ_1 on the set $[\![1, 52]\!]$. For all $l \in [\![1, 52]\!]$:
  - If there exists $l' \in [\![1, 39]\!]$ such that $\mathsf{PK}_{0,l} = \hat{\mathsf{pk}}_{l'}$, then $\mathcal{B}$ sets $\tilde{h}_{1,l} = \hat{h}_{l'}$ and $\tilde{PK}_{1,l} = \hat{\mathsf{PK}}_{l'}$.
  - Else it sets $\tilde{h}_{1,l} = h_{0,l}^{r_{1,l}}$ and $\tilde{\mathsf{PK}}_{1,l} = \mathsf{PK}_{0,l}^{r_{1,l}}$.

  Finally $\mathcal{C}$ sets $h_{1,l} = \tilde{h}_{1,\delta_1(l)}$ and $\mathsf{PK}_{1,l} = \tilde{PK}_{1,\delta_1(l)}$, and generates the proof Π_1 by running the simulator of the proof of knowledge of randomization of a set of commitments. $\mathcal{B}$ sends $\{(h_{1,l}, \mathsf{PK}_{1,l})\}_{1 \le l \le 52}$ and Π_1 to $\mathcal{D}_3$.
- If $i \neq 1$ and $i \neq 4$, then $\mathcal{B}$ processes as in the real protocol.
- If $i = 4$, it receives $\{(h_{4,l}, \mathsf{PK}_{4,l})\}_{1 \le l \le 52}$ and Π_4 from $\mathcal{D}_3$, then it uses the knowledge extractor of the proof of knowledge of randomization of a set of commitments (which is extractable by hypothesis) to extract the function δ_4. If this extraction fails, it aborts the experiment and returns a random bit. If the proof Π_4 is not valid, then $\mathcal{B}$ aborts the experiment and returns a random bit.

$\mathcal{B}$ computes $\mathsf{PK}^*_{i_c,l} = \left((\mathsf{PK}_{i_c,l})^{\prod_{j=2}^{i_c} \frac{1}{r_{j,\delta_{j+1}(\dots\delta_{i_c}(l)\dots)}}}\right)$.

If $\{\hat{\mathsf{PK}}_{l'}\}_{1 \le l' \le 39} \not\subset \{\mathsf{PK}^*_{i_c,l}\}_{1 \le l \le 52}$, then $\mathcal{B}$ aborts the experiment and returns a random bit, because the adversary does not correctly shuffle the commitments of the honest players.

Case 2 ($i_c \neq 4$)**:** For all $i \in \llbracket 1, 4 \rrbracket$:

- If $i \neq i_c$ and $i \neq i_c + 1$, then $\mathcal{B}$ processes as in the real protocol.
- If $i = i_c$, it receives $\{(h_{i,l}, \mathsf{PK}_{i,l})\}_{1 \leq l \leq 52}$ and Π_i from $\mathcal{D}_3$, then it uses the knowledge extractor of the proof of knowledge of randomization of a set of commitments (which is extractable by hypothesis) to extract the function δ_4 and the vector r_i. If this extraction fails, it aborts the experiment and returns a random bit. If the proof Π_4 is not valid then $\mathcal{B}$ aborts the experiment and returns a random bit.
- If $i = i_c + 1$, then $\mathcal{B}$ picks a vector $r_i = (r_{i,1}, \ldots, r_{i,52}) \xleftarrow{\$} (\mathbb{Z}_q^*)^{52}$, and a permutation δ_i on the set $\llbracket 1, 52 \rrbracket$. For all $l \in \llbracket 1, 52 \rrbracket$, it computes $\mathsf{PK}^*_{i-1,l} = \left((\mathsf{PK}_{i-1,l})^{\prod_{j=1}^{i-1} \frac{1}{r_{j,\delta_{j+1}(\ldots\delta_{i-1}(l)\ldots)}}} \right)$ then:
  - If there exists $l' \in \llbracket 1, 39 \rrbracket$ such that $\mathsf{PK}^*_{i-1,l} = \hat{\mathsf{pk}}_{l'}$, then $\mathcal{B}$ sets $\tilde{h}_{i,l} = \hat{h}_{l'}$ and $\tilde{PK}_{i,l} = \hat{\mathsf{PK}}_{l'}$.
  - Else it sets $\tilde{h}_{i,l} = h^{r_{i,l}}_{i-1,l}$ and $\tilde{\mathsf{PK}}_{i,l} = \mathsf{PK}^{r_{i,l}}_{i-1,l}$.

  Finally $\mathcal{C}$ sets $h_{i,l} = \tilde{h}_{i,\delta_i(l)}$ and $\mathsf{PK}_{i,l} = \tilde{PK}_{i,\delta_i(l)}$, and generates the proof Π_i by running the simulator of the proof of knowledge of randomization of a set of commitments. $\mathcal{B}$ sends $\{(h_{i,l}, \mathsf{PK}_{i,l})\}_{1 \leq l \leq 52}$ and Π_i to $\mathcal{D}_3$.

$\mathcal{B}$ computes $\mathsf{PK}^*_{i_c,l} = \left((\mathsf{PK}_{i_c,l})^{\prod_{j=1}^{i_c} \frac{1}{r_{j,\delta_{j+1}(\ldots\delta_{i_c}(l)\ldots)}}} \right)$.

If $\{\hat{\mathsf{pk}}_{l'}\}_{1 \leq l' \leq 39} \not\subset \{\mathsf{PK}^*_{i_c,l}\}_{1 \leq l \leq 52}$, then $\mathcal{B}$ aborts the experiment and returns a random bit, because the adversary does not correctly shuffle the commitments of the honest players.

Finally, in all cases $\mathcal{B}$ sets $\delta = \delta_4 \circ \delta_3 \circ \delta_2 \circ \delta_1$, and for all $l \in \llbracket 1, 52 \rrbracket$ it sets $h_l = h_{4,l}$, $\mathsf{PK}_l = \mathsf{PK}_{4,l}$, and $\mathsf{PK} = ((h_1, \mathsf{PK}_1), \ldots, (h_{52}, \mathsf{PK}_{52}))$.

Shuffle phase: $\mathcal{D}_4$ sends y, then $\mathcal{B}$ and $\mathcal{D}_4$ generate D by running the protocol $\mathsf{DeckGen}_{P_1,P_2,P_3,P_4}$ together, such that $\mathcal{D}$ plays the role of P_{i_c}. $\mathcal{B}$ sets $p = p_*$ and parses D as $(\mathsf{id}_1, \ldots, \mathsf{id}_{52})$. For all $i \in \{1, 2, 3, 4\}$, $\mathcal{B}$ computes $H_i = (\mathsf{id}_{\delta((i-1)\cdot 13+1)}, \cdots, \mathsf{id}_{\delta((i-1)\cdot 13+13)})$.

Game phase: $\mathcal{B}$ sets $\gamma = 0$ and $\mathsf{played} = \perp$. For $i \in \llbracket 1, 52 \rrbracket$:

If $p \neq i_c$**:** $\mathcal{B}$ runs $\mathsf{id} \leftarrow \mathsf{Strat}_p(\mathsf{played}, H_p, p_*, p)$, then it processes as the algorithm $(\Pi, \mathrm{st}') \leftarrow \mathsf{Play}(p, \mathsf{id}, \mathsf{sk}_p, \mathsf{pk}_p, \mathrm{st}, \mathsf{PK}, D)$ except that:

- We recall that there exists v such that $\mathsf{id} = \mathsf{id}_v$. It chooses $t = \delta^{-1}(v) - (p-1) \cdot 13$. Hence, $v = \delta((p-1) \cdot 13 + t)$.
- For all $j \in \llbracket 1, 13 \rrbracket$, it computes Π_j as in the simulator $\mathsf{Sim}^{\mathsf{Spades}}_{\mathsf{SecureSpade},S,\mathcal{A}(\mathsf{vw})}(k, \mathsf{setup}, s, \{\mathsf{pk}_i\}_{1 \leq i \leq 4}, \mathsf{PK}, \mathsf{vw})$.

It sends $(\mathsf{id}, \Pi, \mathrm{st}')$ to $\mathcal{D}_4$ and updates $\mathrm{st} := \mathrm{st}'$. Finally, it updates the index p that points the next player according to the rule of Spades. It then parses played as $(\mathsf{pl}_1, \ldots, \mathsf{pl}_n)$ (where $n = |\mathsf{played}|$) and updates $\mathsf{played} := (\mathsf{pl}_1, \ldots, \mathsf{pl}_n, \mathsf{id})$.

If $p = i_c$: $\mathcal{B}$ processes as in $\mathsf{Exp}^{\mathsf{Spades}}_{\mathsf{SecureSpade},S,K,\mathcal{D}}(k)$.

Final phase The simulated experiment returns 1.

Finally, $\mathcal{B}$ runs $b_* \leftarrow \mathcal{D}_5(\mathsf{vw})$, where vw denotes all the values send and received by $\mathcal{D}$ during its interaction with the simulated experiment, then $\mathcal{B}$ returns b_*.

Analysis: We distinguish two cases :

- The adversary forges a proof of a false statement during the $\mathsf{GKeyGen}$ protocol. In this case, if $\mathcal{D}$ does not produce a valid proof for the false statement, then $\mathsf{PK} = \perp$ so the experiment aborts, hence the advantage of $\mathcal{B}$ is lower than $\epsilon_{\mathsf{sound}}(k)$.
- The adversary does not forge a proof of a false statement during the $\mathsf{GKeyGen}$ protocol. In this case, if $b = 1$ and $\mathsf{PK} \neq \perp$, then for all $v \in [\![1, 52]\!]$, there exist $i \in [\![1, 4]\!]$ and $j \in [\![1, 13]\!]$ such that $\log_g(\mathsf{pk}_{i,j}) = \log_{h_v}(\mathsf{PK}_v)$. In the following, we show that in this case, the advantage of $\mathcal{B}$ is lower than $2 \cdot \epsilon_{39\text{-}\mathrm{Sh}\mathsf{DDH}}(k)$. If $b = 1$, then the experiment is perfectly simulated, else, the simulator $\mathsf{Sim}^{\mathsf{Spades}}_{\mathsf{SecureSpade},S,\mathcal{A}(\mathsf{vw})}(k, \mathsf{setup}, \{\mathsf{pk}_i\}_{2 \leq i \leq n}, \mathsf{PK}, \mathsf{vw})$ is perfectly simulated. We observe that:

$$\begin{aligned}\mathsf{Pr}[\mathcal{B} \text{ wins}] &= \frac{1}{2}\left(\mathsf{Pr}[1 \leftarrow \mathcal{D}_5(\mathsf{vw})|b = 1] + \mathsf{Pr}[0 \leftarrow \mathcal{D}_5(\mathsf{vw})|b = 0]\right) \\ &= \frac{1}{2}\left(\mathsf{Pr}[1 \leftarrow \mathcal{D}_5(\mathsf{vw})|b = 1] - \mathsf{Pr}[1 \leftarrow \mathcal{D}_5(\mathsf{vw})|b = 0] + 1\right).\end{aligned}$$

Finally:

$$\left|\mathsf{Pr}[\mathcal{B} \text{ wins}] - \frac{1}{2}\right| = \frac{1}{2} \cdot |P_{\mathsf{real}}(\mathcal{D}, k) - P_{\mathsf{sim}}(\mathcal{D}, k)| = \frac{\lambda(k)}{2}.$$

The two cases imply that the advantage of $\mathcal{B}$ is lower than $2 \cdot \epsilon_{39\text{-}\mathrm{Sh}\mathsf{DDH}}(k) + \epsilon_{\mathsf{sound}}(k)$, which concludes the proof. □