The Landscape of Optimal Card-Based Protocols[1]

Alexander Koch[†,*]

[†]Competence Center for Applied Security Technology (KASTEL), Karlsruhe Institute of Technology (KIT)

Received: | *Revised:* | *Accepted:*

Abstract *In card-based cryptography – initiated by the "Five-Card Trick" of den den Boer (EUROCRYPT 1989) for securely computing the AND of two players' bits – one devises simple protocols for secure multiparty computation using a deck of playing cards of two symbols (♡ and ♣) with indistinguishable backs. These protocols can be run if no trusted computer is available, or in classroom settings to illustrate secure computations. We give two AND protocols which are card-minimal w.r.t. specific requirements, and show card-minimality of the COPY protocol (necessary for computing arbitrary circuits) of Mizuki and Sone (FAW 2009) and the AND protocol of Abe et al. (APKC 2018). By this, we completely determine the landscape of card-minimal protocols with respect to running time requirements (finite runtime or Las Vegas behavior with/without restarts) and practicality demands on the shuffling operations. For this, we systematize and extend techniques for lower bounds results in card-based cryptography.*

Keywords: Card-based protocols, Secure computation, Committed format, Cryptography without computers

2010 Mathematics Subject Classification: 97A20, 05A05, 05A18, 05E18, 94A60, 97Q60, 68Q17, 68R15

1 INTRODUCTION

One of the main achievements of modern cryptography is definitively the invention of secure multiparty computation (MPC), which has become quite practicable in recent years. While the world is full of opportunities where MPC would pose an elegant and more secure solution to a practical problem, it remains largely unknown outside of the cryptographic community. Hence, there is some impetus to us as a community to present our tools to the outside world, in particular to practitioners and decision makers.

Here, card-based cryptography, which implements MPC with just a few playing cards of two colors, ♡ and ♣, and indistinguishable backs, allows to clearly communicate the underlying ideas to non-experts and a general science-interested audience. This appeal has motivated researchers to come up with simple protocols for the basic building blocks, namely for computing the AND of two players' bits, and producing, from two cards encoding a bit, $n \in \mathbb{N}$ card pairs encoding the same bit, here called (n-)COPY protocols. Because of the physical nature of the cards, and because negating a bit is very easy, these protocols are sufficient and necessary to compute any circuit.

Oftentimes, e.g., in didactic contexts, only a very simple operation, such as a single AND is needed. An example is the dating problem, where two players would like to determine whether there is mutual interest (an AND on whether a person likes the other), without leaking anything about their answers, if the interest is one-sided only. For really using or explaining card-based cryptography it is hence good (and often sufficient) to know which the best (card-minimal) protocol to use are – depending of course on your needs.

For example, if you want to run an AND protocol that terminates after a certain number of steps, and want to use only natural shuffles, which permute the cards according to a permutation subgroup with a uniform distribution, called *uniform closed shuffles* in the following, then you need six cards, and the protocol by Mizuki and Sone [14] would be the natural choice. Let us give a short informal description of the protocol to understand the aims and functioning of typical card-based protocols. We start by giving both Alice and Bob two cards, one of each color (♡ and ♣). Then, the players secretly input their bits via the order of the two cards. Here, face-down cards in order ♡♣ encode a 1; the reverse order ♣♡ encodes a 0.[2] The protocol needs two additional helping cards encoding 0 that are put face-down next to the cards of the players. Hence, at the beginning, we are in one of the following configurations:

$$\underbrace{\heartsuit\clubsuit}_{a=1}\ \underbrace{\heartsuit\clubsuit}_{b=1}\ \underbrace{\clubsuit\heartsuit}_{0}\ /\ \underbrace{\heartsuit\clubsuit}_{a=1}\ \underbrace{\clubsuit\heartsuit}_{b=0}\ \underbrace{\clubsuit\heartsuit}_{0}\ /\ \underbrace{\clubsuit\heartsuit}_{a=0}\ \underbrace{\heartsuit\clubsuit}_{b=1}\ \underbrace{\clubsuit\heartsuit}_{0}\ /\ \underbrace{\clubsuit\heartsuit}_{a=0}\ \underbrace{\clubsuit\heartsuit}_{b=0}\ \underbrace{\clubsuit\heartsuit}_{0}$$

[1]The content of this paper appeared in the author's dissertation [6].

[**]Corresponding Author: alexander.koch@kit.edu

[2]If inputs *and outputs* are given in this format, the protocols are in *committed format*. This is the most interesting case considered in card-based cryptography, as this standard format ensures reusability of the outputs without having to learn them in the process. Other encodings on this *two-color deck* that come to mind, such as a single-card encoding ($\heartsuit \mathrel{\hat{=}} 1$, $\clubsuit \mathrel{\hat{=}} 0$) do *not* save cards, as the player still needs both card types anyway to have free decision on which card to input to the protocol, but such an encoding would make it impossible to copy or invert the bit with perfect security due to an impossibility result of [12]. See also [6, pp. 50–51] for an extensive discussion of different encoding choices.

Now, let us note that “a AND b”, denoted by $a \wedge b$ in the following, is equivalent to “if a then b else 0”. If we would not care about privacy, we could therefore turn over Alice’s cards and if they show $\heartsuit\clubsuit = 1$, then the result is encoded by Bob’s cards (at positions 3, 4), and otherwise by the two helping cards encoding a 0 (at positions 5, 6). But privacy requires us to mask Alice’s bit first.

Observe that $a \wedge b$ is *also* equivalent to “if $\neg a$ then 0 else b”, i.e., we inverted the bit of a and exchanged the two branches of the if-statement. We can do so also with the cards, by swapping Alice’s cards and exchanging Bob’s cards with the two helping cards, ending in a configuration as follows:

$$\underbrace{\clubsuit\,\heartsuit}_{\neg a=0}\ \underbrace{\clubsuit\,\heartsuit}_{0}\ \underbrace{\heartsuit\,\clubsuit}_{b=1}\ /\ \underbrace{\clubsuit\,\heartsuit}_{\neg a=0}\ \underbrace{\clubsuit\,\heartsuit}_{0}\ \underbrace{\clubsuit\,\heartsuit}_{b=0}\ /\ \underbrace{\heartsuit\,\clubsuit}_{\neg a=1}\ \underbrace{\clubsuit\,\heartsuit}_{0}\ \underbrace{\heartsuit\,\clubsuit}_{b=1}\ /\ \underbrace{\heartsuit\,\clubsuit}_{\neg a=1}\ \underbrace{\clubsuit\,\heartsuit}_{0}\ \underbrace{\clubsuit\,\heartsuit}_{b=0}$$

This operation is specified by the permutation $(1\ 2)(3\ 5)(4\ 6)$, mapping $1 \mapsto 2, 2 \mapsto 1, 3 \mapsto 5, 5 \mapsto 3, 4 \mapsto 6, 6 \mapsto 4$. Hence applying a shuffle on the cards, which performs the said permutation with probability $1/2$ and leaves the cards as is otherwise, effectively randomizes the order of the first two cards, which can therefore be turned over without leaking Alice’s secret bit. The result is then under the cards as specified above. Because the helping cards and Bob’s cards are indistinguishable, one cannot tell which of the two is at the output positions.

Interestingly, this shuffle operation can be performed by a *random bisection cut*, which puts the cards evenly in two groups and exchanges them with probability $1/2$, as specified in [14]. Formally, it is modeled as an operation specified by a permutation set $\Pi = \{\mathsf{id}, (1\ 2)(3\ 5)(4\ 6)\}$, where id is the identity permutation (doing nothing), and a probability distribution $\mathcal{F}$ on Π, such that a permutation from Π is drawn via $\mathcal{F}$ and obliviously applied to the cards.

The random bisection cut has two features which make it particularly easy to implement, namely that $\mathcal{F}$ is the *uniform distribution* on Π, and that Π is a permutation *group*. We call such shuffles *uniform* and *closed*, respectively. Both properties are usually wanted from a protocol, hence it is natural to ask for lower bounds on cards for AND not only in general, but also when protocols are restricted to *uniform closed*[3] shuffling operations. This has first been done in [4] for finite-runtime closed shuffle AND protocols.

Contribution. While finite-runtime shuffle-restricted AND protocols are already well understood w.r.t. their minimal number of cards, respective lower bounds in *shuffle-restricted n-COPY protocols* have so far been unexplored. This is also because n-COPY protocols are difficult to analyze: there are $(2n + 1)!$ permutations on $2n + 1$ cards, hence you really need to understand the underlying structure and cannot explore by just trying out some combinations and generalize from there. By giving a systematic approach to impossibility proofs, this work gets around these problems. We show that no committed format closed-shuffle protocol can go with less than $2n + 2$ cards, even if we allow Las Vegas behavior with restarts and non-uniform shuffles. This completely tightens all bounds w.r.t. the running time (finite-runtime, Las Vegas with/without restarts) and shuffle parameters. This is our main result.

Additionally, we derive tight lower bounds on the number of cards for AND protocols in refined settings. As a motivation for this note that for a demonstration of a single-shot committed-format AND in front of the class, using exactly four cards is nice, as you could go without any helper cards. In this case you probably care about simple shuffles, but not very much about the runtime behavior, e.g. it might be acceptable to have a small probability of restarting the protocol, including reinserting the inputs. In this setting it is interesting to know whether there is a four-card protocol. We show that the contrary is true, namely that you need five cards, even if restarts are allowed. This shows that the protocol of Abe et al. [1] is card-minimal for AND protocols restricted to uniform closed shuffles.

Beside answering these questions, we systematize and strengthen the toolset of impossibility proofs on card-based protocols. Apart from the more simple setting of [4, Sect. 8], this paper contains the first impossibility proof that takes into account restarts. Hence, we also show some useful properties for protocols in this setting. Note however that restarting protocols are impractical for larger composed protocols, as a failure would mean that you have to rerun the larger protocol. Nevertheless, we believe that our systematization also helps to prove lower bounds on protocols for more complex (e.g., three-input majority [16]) or even arbitrary n-ary boolean functions.

On the positive side, we tweak two AND protocols from the literature to have better properties with regard to their shuffles, completing the search for tight bounds also for AND protocols. Furthermore, we believe that there is also a theoretical motivation to get the table filled with tight bounds only.

Note that this paper solely focuses on the setting, where the deck consists of a multiset of the two symbols $\heartsuit$ and $\clubsuit$, which is called the *two-color deck setting*. This is by far the most common setting in card-based cryptography, not only due to its simplicity. Alternatively, few works [15, 11, 7, 8] have addressed the case of a *standard deck*, i.e. where all cards differ, as in the decks that you can usually buy in shops. In [7, 8], the above-mentioned new

[3]Not only are they much simpler to implement in a passively secure way, e.g. by the players taking turns in randomly choosing a permutation from the group and applying it with the other player not looking (where the closedness property of being a permutation group guarantees that the resulting permutation is in Π again, and the uniform probability of at least one individual permutation carries over to the total permutation applied in the shuffle), there is also an actively secure way to perform any uniform closed shuffle using additional helping cards, cf. [9]. Also, these shuffles are already complete in that they allow to compute any Boolean function.

Table 1: Minimum number of cards required by committed format AND and n-COPY protocols, subject to the requirements specified in the first two columns. The second column specifies shuffle restrictions. See also Fig. 4.

Runtime	**Shuffle Restr.**	**#Cards**	**Protocol**	**Lower Bound**
AND PROTOCOLS:				
restart-free LV	closed	4	[10, Sect. 4]	– (trivial)
restart-free LV	uniform	4	Theorem 3 (this work) / [19]	– (trivial)
restarting LV	uniform closed	5	[1, Sect. 2]	Theorem 2 (this work, generalizes [4])
restart-free LV	uniform closed			
finite runtime	–	5	Theorem 4 (this work) / [19]	[10, Sect. 6]
finite runtime	uniform			
finite runtime	closed	6	[14, Sect. 3]	[4, Sect. 6]
finite runtime	uniform closed			
COPY PROTOCOLS:				
restarting LV	–	$2n + 1$	[17, Sect. 5]	[4, Sect. 8]
restart-free LV	uniform			
restarting LV	closed	$2n + 2$	[14, Sect. 5]	Theorem 1 (this work), [4, Sect. 9]
finite runtime	–			
finite runtime	uniform closed			

systematized proof technique for lower-bounds on the number of cards was adapted to this standard deck setting and successfully applied to show the impossibility of finite-runtime four-card standard-deck AND protocols.

Summary of Contribution. In this paper, we

- introduce a backward calculus on sets of states which allows to more systematically prove lower bounds on the number of cards in card-based protocols. For this, we provide some useful properties of the defined notions.
- give two new protocols which are optimal w.r.t. their parameters. More precisely, we specify a five-card finite-runtime and a four-card restart-free Las Vegas AND protocol which use only uniform (non-closed) shuffles. (Note that these are variations of the protocols from [10].)
- prove that five cards are necessary for AND protocols restricted to uniform closed shuffles, even with restarts. This bound is tight due to [1].
- prove that $2n + 2$ cards are necessary for producing n copies of a bit, when restricted to closed shuffles. This bound is tight due to [14].

Hence, in summary, this paper can be seen as the remaining puzzle piece for tight bounds on the number of cards for AND and COPY protocols w.r.t. all the interesting parameters: *running time* (finite runtime, restart-free Las Vegas, restarting Las Vegas), shuffle *uniformity* and *closedness*. For comparison, see Table 1.

Concurrent and Independent Related Work. Ruangwises and Itoh [18, 19] posted equivalent versions of the two uniform AND protocols from Appendix A. This constitutes concurrent and independent work, and was published as an earlier version of this paper on the IACR ePrint Archive, cf. [5].

Outline. In Section 2 we introduce the necessary notions for card-based cryptography, including protocol trees. As an important analysis tool, we define our "backward calculus" on state sets in Section 3. We use this to prove a lower bound of $2n + 2$ cards for n-COPY protocols with closed shuffles in Section 4. We prove that five cards are necessary for AND with uniform closed shuffles, even if we allow restarts, in Section 5. Appendix A gives two AND protocols, which are card-minimal with respect to a restriction to uniform shuffles and different runtime requirements.

Notation (Permutations). S_n denotes the *symmetric group* on $\{1, \dots, n\}$. For elements $x_1, \dots, x_k \in \{1, \dots, n\}$ the *cycle* $(x_1\ x_2\ \dots\ x_k)$ is the *cyclic* permutation π with $\pi(x_i) = x_{i+1}$ for $1 \leq i < k$, $\pi(x_k) = x_1$ and $\pi(x) = x$ for all x not occurring in the cycle. Every permutation can be written as a composition of pairwise disjoint cycles.

2 PRELIMINARIES

Card-based protocols operate on a *deck* of cards, which is specified by a multiset $\mathcal{D}$ of symbols from $\{\heartsuit, \clubsuit\}$. Multisets will be denoted by brackets, e.g. as $[\heartsuit, \heartsuit, \clubsuit, \clubsuit]$. As in the informal description of the six-card AND protocol, we have essentially four operations, namely we can *i)* look under cards, learning their hidden symbols, *ii)* deterministically permute the cards on the table, *iii)* shuffle the cards in some controlled way to introduce randomness and obscure which card is which, and *iv)* terminate with a list of card positions which specify the output of the protocol. For completeness, we consider one additional operation, namely *v)* a restart, which helps in cases where the protocol run hits some unwanted cases where we cannot evaluate to the end result any longer. Protocols using this operation are necessarily Las Vegas and prompt the user for the input bits again, which makes them impractical.

For introducing the computational model more formally, we make use of an equivalence of card-based protocols as defined in [12, 13] to state trees of [10] (also called KWH trees), due to Kastner et al. [4, Sections 3 and 4]. See the latter reference for a more thorough explanation. Using this, a protocol is given by a tree, where the nodes are the possible states of a protocol. The states are enriched in that they contain full information about which of the card sequences is possible at that time of the protocol run, with an annotation of the symbolic probability of the sequence in terms of the probabilities for the protocol inputs. More formally, for each possible input $i \in \{0,1\}^k$ to a k-ary Boolean function that is computed by the protocol, we have a variable X_i representing the symbolic probability that i was the input to the protocol. This leads to a formal definition:

Definition 1 (State, cf. [6]). *A* state *of a protocol on deck $\mathcal{D}$ computing a k-ary Boolean function is a map μ from the set of sequences on the deck $\mathcal{D}$ to the homogenous polynomials of degree* 1 *with variables in X_i for all inputs $i \in \{0,1\}^k$, and non-negative coefficients, such that the values of on all sequences "sum to one", i.e. $\sum_{\text{seq. } s \text{ on } \mathcal{D}} \mu(s) = \sum_{i\in\{0,1\}^k} X_i$. (Representing probability distributions, states are certain convex combinations.)*

As a reference, see the tree of the six-card AND protocol in Fig. 1. The state tree is directed, with annotations at the outgoing edges of the state, specifying the action that is performed next. These are defined next. Here, let μ be the state with the outgoing annotation:

i. (turn, T) branches the tree into states μ_v for each observation v possible by looking under the cards at positions $T \subseteq \{1,\dots,|\mathcal{D}|\}$. μ_v contains the sequences from μ which are compatible with observation v. For each sequence s compatible with v, we have $\mu_v(s) \coloneqq \mu(s)/\rho$, where $\rho \in (0,1]$ is the probability of observing v.
ii. (perm, π) permutes the sequences of the state using π, defined via $(\mathsf{shuffle}, \{\pi\})$, as described next.
iii. $(\mathsf{shuffle}, \Pi, \mathcal{F})$ leads to a state μ' given by $\mu'(s) \coloneqq \sum_{\pi\in\Pi} \mathcal{F}(\pi)\cdot\mu(\pi^{-1}(s))$, where s is a sequence from $\mathcal{D}$, Π a set of permutations from $S_{|\mathcal{D}|}$, and $\mathcal{F}$ is a probability distribution on Π. If $\mathcal{F}$ is omitted, it is assumed to be the uniform distribution on Π. We call a shuffle *uniform* in this case, and *closed* if Π is a subgroup of $S_{|\mathcal{D}|}$.
iv. $(\mathsf{result}, p_1,\dots,p_r)$ halts the protocol and specifies that the output is at card positions $p_1,\dots,p_r$.
v. $(\mathsf{restart})$ restarts the protocol. A protocol is *restarting* if it uses this operation, and *restart-free* otherwise.

We want to specify what it means for a protocol to compute a Boolean function. For this, we say that two (face-down) cards of symbols $\heartsuit$ and $\clubsuit$ *represent a commitment to* 1, if they are in order $\heartsuit\,\clubsuit$, and 0 otherwise. We extend this to sequences $x = (x_1,\dots,x_{2k})$ as follows: x *encodes* a bit string $b \in \{0,1\}^k$, if $x_{2t-1}\ x_{2t}$ represents $b[t]$ for all $1 \le t \le k$. A protocol then *computes a function* $f\colon \{0,1\}^k \to \{0,1\}^\ell$, $k,\ell \in \mathbb{N}$, if the deck $\mathcal{D}$ is a superset of $\max(k,\ell)\cdot[\heartsuit,\clubsuit]$, the start state (root of the tree) contains sequences encoding each $b \in \{0,1\}^k$ in the first $2k$ cards (the remaining cards are independent of b), and for any leaf node μ to which a result operation is assigned, the following holds: Let $p_1,\dots,p_r$ be the positions specified by the result operation. For all sequences s in μ and for all for all X_i occurring in $\mu(s)$, the subsequence $(s[p_1],\dots,s[p_r])$ encodes the value $f(i) \in \{0,1\}^\ell$ *(Correctness)*.

Definition 2. *The protocol is* secure *if at any turn operation, the probability for each observation v sums to a constant $\rho \in [0,1]$ (using $\sum_{i\in\{0,1\}^k} X_i = 1$). (This is equivalent to the usual definition of hiding both input and output from any observations you make during the protocol run, due to [4].) The security condition effectively states that for any turn operation in the tree, the state before the turn needs to be* turnable *at that position. For a state μ and an index $j \in \{1,\dots,|\mathcal{D}|\}$, this is defined as: for any $c \in \mathcal{D}$, there is a constant $\rho \in [0,1]$, such that* $\displaystyle\sum_{s \text{ with } s[j]=c} \mu(s) = \rho \sum_{i\in\{0,1\}^k} X_i$.

A protocol is *finite runtime* if the protocol tree is finite. A protocol is *Las Vegas*, if it is not finite runtime, but the expected length of any path in its tree is finite. As above, it is restart-free if it does not contain any restart operations, and restarting otherwise. We may draw edges to earlier states if the protocol tree is self-similar, as in Fig. 5.

For searching protocols, it is useful to note that deterministically permuting does not lead to an essentially new state, hence we want to identify states that are just a permuted version of each other. We call these (reduced) states *similar*. If one interprets a state as a matrix of symbols with annotated rows (as displayed in the state trees), states are similar, if there is a permutation on the columns mapping one state to the other.

Reduced State Trees. As in [4, Def. 3] we define reduced states, where states are not annotated by their symbolic probabilities, but by the output that is prescribed to it, given the ways it can arise from the input sequences (made precise below). This is to simplify impossibility results, as reducing information allows us to make sense of which types of states are reachable without caring about the exact probabilities, and it actually makes the search space of

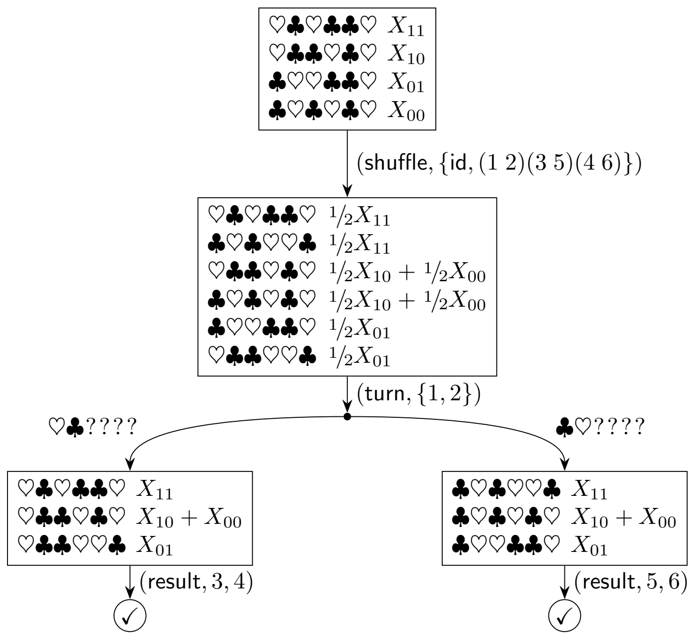


Figure 1: The six-card AND protocol from [14].

(reachable) states finite. Reduced trees capture only a weak form of security, namely *possibilistic security*, where each output in the image of the function needs to be still possible (but not that the outputs obey a certain probability distribution). Showing that a protocol is impossible in this weak sense already implies its general impossibility.

Here, to obtain a reduced state tree, we project all the symbolic probabilities of the sequences in a state tree to a *type* (representing the possible future output associated with the sequence in a correct protocol, see below), which can be any $o \in \{0,1\}^\ell$ and also $\perp$ if no clear output can be generated. For this, let $\mathcal{P}$ be a protocol computing a function $f\colon \{0,1\}^k \to \{0,1\}^\ell$ and μ be a state in the state tree. For any sequence s with $\mu(s)$ being a polynomial with positive coefficients for the variables $X_{b_1}, \ldots, X_{b_i}$ $(i \geq 1)$, set $\hat{\mu}(s) := o \in \{0,1\}^\ell$ if $o = f(b_1) = f(b_2) = \ldots = f(b_i)$ in the resulting *reduced state* $\hat{\mu}$. If there are b, b' with positive coefficients for variables $X_b, X_{b'}$ which do not evaluate to the same output under f, i.e. $f(b) \neq f(b')$, set $\hat{\mu}(s) := \perp$. We call sequences in $\hat{\mu}$ according to their type *o-sequences* or *⊥-sequences*. Alternatively, we say they are *of type o* or *of type* ⊥, respectively. As no ambiguities can arise, the 1^ℓ- and 0^ℓ-sequences of ℓ-COPY protocols are additionally called 1- and 0-sequences, respectively.

Note that *turnability of a position i in a reduced state* $\hat{\mu}$ w.r.t. possibilistic security implies the following: For each $c \in \mathcal{D}$ occurring in column i, among sequences with $s[i] = c$, there is an r-sequence for all $r \in \{0,1\}^\ell$ in the image of the function computed by the protocol, or a ⊥-sequence. This is a necessary criterion for all outputs still being possible, and hence capturing possibilistic (output) security.

Semi-reduced States and their Partitions. In this paper, we introduce a new representation, which is an intermediate between states and reduced states, but captures the *distinctness* of sequence probabilities that is not contained in (fully) reduced states. We make use of this additional information in impossibility proofs for protocols which are restricted to uniform closed shuffles. As an intermediary step, we need to introduce partitions of a state. We refer the reader to Fig. 2 for an example and illustration of what our definitions in this section aim at.

♡♡♣♣	$1/2X_{11}$
♡♣♡♣	$1/2X_{11}$
♣♡♡♣	$1/2X_{10} + 1/2X_{01}$
♡♣♣♡	$1/2X_{10} + 1/2X_{01}$
♣♡♣♡	$1/2X_{00}$
♣♣♡♡	$1/2X_{00}$

♡♡♣♣	1_1
♡♣♡♣	1_1
♣♡♡♣	0_1
♡♣♣♡	0_1
♣♡♣♡	0_2
♣♣♡♡	0_2

♡♡♣♣	1
♡♣♡♣	1
♣♡♡♣	0
♡♣♣♡	0
♣♡♣♡	0
♣♣♡♡	0

Figure 2: A state μ from Fig. 5 given in three forms. *Left:* the full (non-reduced) form, *middle:* its semi-reduced form, where the 0-sequences are divided into two parts, and *right:* its reduced form, which no longer carries the information about these distinct probabilities. The state is turnable at positions 2 and 3.

A *partition* $P(\mu)$ of a state μ arises by putting all sequences of the state into the same set, which evaluate to the same symbolic probability. (More formally, it arises from the equivalence relation of having the same $\mu(\cdot)$ value).

Moreover, a *type partition* $T(\mu)$ of μ is defined similarly, putting all sequences of the same type (i.e. the associated output value or a $\bot$-symbol) into a set.

Example 1. *The state μ of Fig. 2 (left), has partition $P(\mu) = \{\{\heartsuit\heartsuit\clubsuit\clubsuit, \heartsuit\clubsuit\heartsuit\clubsuit\}, \{\clubsuit\heartsuit\heartsuit\clubsuit, \heartsuit\clubsuit\clubsuit\heartsuit\}, \{\clubsuit\heartsuit\clubsuit\heartsuit, \clubsuit\clubsuit\heartsuit\heartsuit\}\}$ (which is also encoded in its semi-reduced representation as exactly the sequences with the same annotation end up in a common set of the partition), and a type partition $T(\mu) = \{\{\heartsuit\heartsuit\clubsuit\clubsuit, \heartsuit\clubsuit\heartsuit\clubsuit\}, \{\clubsuit\heartsuit\heartsuit\clubsuit, \heartsuit\clubsuit\clubsuit\heartsuit, \clubsuit\heartsuit\clubsuit\heartsuit, \clubsuit\clubsuit\heartsuit\heartsuit\}\}$.*

We aim to represent the partition $P(\mu)$ of a state in its semi-reduced form. For this, we introduce as many copies of types as there are partitions with sequences of this type, and add a subscript from $\mathbb{N}$ to distinguish these, cf. Fig. 2 (middle). (If there is only one partition of a type, we nevertheless give it the subscript 1). The concrete order of these subscripts will be irrelevant for our purposes. This *semi-reduced form of a state* can be generated by appropriately projecting the sequence probabilities to these *indexed type* symbols.

2.1 PROPERTIES OF CLOSED SHUFFLES

The set Π of a *closed* shuffle on deck $\mathcal{D}$ is a subgroup of $S_{|\mathcal{D}|}$. Hence, we would like to exploit the algebraic structure that arises when permutations of Π act on the card sequences by reordering them according to their permutation specification. Formally, given a permutation $\pi \in \Pi$ of a permutation group Π, and a sequence x from the set X of sequences on $\mathcal{D}$, we denote by $\pi(x)$ the natural group action of Π on X, where we apply permutation π on x. A formal definition and short discussion is given in the beginning of Appendix B.

Let G be a group acting on a set X. Then we define the *orbit* of an $x \in X$ as $G(x) := \{g(x) \colon g \in G\}$. The orbits form a partition of X, called the *orbit partition* of X through G. The *stabilizer subgroup* $\mathsf{Stab}_x(G)$ of a group G w.r.t. an $x \in X$, is defined as $\mathsf{Stab}_x(G) = \{g \in G : g(x) = x\}$. For an analysis of a “backward shuffle” (to be introduced in Section 3), we make use of the state partitions as defined above. To state our criterion in a formal way, we define the (natural) ordering on partitions as follows: Given two partitions P, P' on some set M, we say that P *is finer* than P', or equivalently, that P' *is coarser* than P, if every set in P is subset of a set in P'. For example, for any state μ, its partition $P(\mu)$ is finer than its type partition $T(\mu)$.

As an example, we have that in Example 1 μ’s partition $P(\mu)$ is finer than $T(\mu)$, as $\{\heartsuit\heartsuit\clubsuit\clubsuit, \heartsuit\clubsuit\heartsuit\clubsuit\}$ is contained in both sets, and both $\{\clubsuit\heartsuit\heartsuit\clubsuit, \heartsuit\clubsuit\clubsuit\heartsuit\}$ and $\{\clubsuit\heartsuit\clubsuit\heartsuit, \clubsuit\clubsuit\heartsuit\heartsuit\}$ are subsets of the larger four-element set of $T(\mu)$. Using these, let us rephrase relevant results from [4, Sect. 5], in our new vocabulary:

Lemma 1 ([4, Lemma 2 and 3])**.** *Let μ be transformed into μ' via some shuffle with group* Π. *Then the* orbit partition *of* Π *is (non-strictly) finer than the* type partition $T(\mu')$ *of μ'. Moreover, if the shuffle is uniform, the orbit partition of* Π *is (non-strictly) finer than the* partition $P(\mu')$ *of μ'.*

Proof. Let μ be transformed into μ' via some shuffle with group Π. Assume to the contrary that there are two sequences in the same orbit of Π, but with distinct type (where type includes $\emptyset$ for a sequence that is not present in the state, i.e. with probability 0, and $\bot$). This is a contradiction to [4, Lemma 2]. In the same way, if the shuffle is uniform, there cannot be two sequences in the same orbit of Π with distinct probabilities due to [4, Lemma 3]. □

Note that the case that it is *strictly* finer occurs, if the sequences of two distinct orbits of the shuffle subgroup happen to have the same symbolic probability or type, respectively, after the shuffle.

3 A BACKWARD CALCULUS FOR CARD-BASED PROTOCOLS

Our main technical contribution is to define a systematic way to determine all states that can reach the final states (states that allow a correct result operation) of a protocol by an allowed set of actions. This happens in an iterative process, which, if it terminates, allows to check if the start state is contained in that set. If it is not contained, no final state is reachable from the start state, showing the impossibility of the aimed-for protocol. For this, we define:

- $\mathsf{shuf}_*^{-1}(\mathcal{G})$ for a set of states $\mathcal{G}$ and $* \in \{\emptyset, \mathrm{u}, \mathrm{c}, \mathrm{uc}\}$ which we omit if $* = \emptyset$. This is the *set of states that are transformed into a state in $\mathcal{G}$ by a shuffle* that is *i)* general if $* = \emptyset$, *ii)* closed if $* = \mathrm{c}$, *iii)* uniform if $* = \mathrm{u}$ and *iv)* uniform closed if $* = \mathrm{uc}$. The trivial shuffle is allowed, i.e. $\mathcal{G} \subseteq \mathsf{shuf}_*^{-1}(\mathcal{G})$.[4]
- $\mathsf{turn}_\star^{-1}(\mathcal{G})$ for a set of states $\mathcal{G}$ and $\star \in \{\emptyset, \mathrm{r}, \mathrm{f}\}$[5] which we omit if $\star = \emptyset$. This is the set of states from $\mathcal{G}$ or states that have a turnable position i such that

  if $\star = \emptyset$: a state from $\mathcal{G}$ is on one of the branches of a turn at i (as immediate child).

  if $\star = \mathrm{r}$: a state from $\mathcal{G}$ is on one of the branches of a turn at i (as immediate child) and all other branches are $\bot$-free, i.e. the other children states do not contain a sequence of type $\bot$.

  if $\star = \mathrm{f}$: all immediate successor states from a turn at i are in $\mathcal{G}$.

[4]We are not restricted to these choices. If one aims to show impossibility of a protocol using only *random cuts*, one can readily extend these.
[5]where f stands for finite runtime, r for restart-free Las Vegas, and $\emptyset$ for Las Vegas without restrictions, i.e. it may use restarts.

Using this, we can derive a very general high-level strategy for proving lower bounds on the number of cards for general functionalities. For this, let $\mathcal{G}$ be the set of final states of the respective protocol and note that they do not include any states that contain a $\bot$-sequence. Let $* \in \{\emptyset, \mathrm{u}, \mathrm{c}, \mathrm{uc}\}$ and $\star \in \{\emptyset, \mathrm{r}, \mathrm{f}\}$. Define by $\mathsf{cl}_{\star,*}(\mathcal{G})$ *the closure* of $\mathsf{turn}_\star^{-1}(\cdot)$ and $\mathsf{shuf}_*^{-1}(\cdot)$ operations on $\mathcal{G}$, i.e. the set of all states that are reachable from $\mathcal{G}$ via a finite sequence of $\mathsf{turn}_\star^{-1}(\cdot)$ and $\mathsf{shuf}_*^{-1}(\cdot)$ operations. If the start state is not contained in $\mathsf{cl}_{\star,*}(\mathcal{G})$, then no protocol exists for the running time/shuffle restrictions $\star$ and $*$, as specified above.

Note that we can define this backward calculus for both detail levels of state trees, namely for (semi-)reduced states and for normal states. However, in this paper we *only* need it for *(semi-)reduced states*, and assume this to be the case throughout the document. If the start state is found in the set derived from the calculus, one may take a closer look and see at which state and by which order of operations it has been added, possibly leading to a protocol, if one switches to the non-reduced version of the calculus. If the process constitutes a search for finite-runtime protocols, the protocol can be directly derived from the steps. Otherwise, we can at least guarantee a restarting Las Vegas protocol, as we can recover at least one protocol path leading to a final state. All branches which leave this path can be replaced with a restart operation, resulting in a protocol.

Remark 1. $\mathsf{shuf}_*^{-1}(\cdot)$, $\mathsf{turn}_\star^{-1}(\cdot)$ *and* $\mathsf{cl}_{\star,*}(\cdot)$ *are (inclusion-)monotone functions. This is because both,* $\mathsf{shuf}_*^{-1}(\cdot)$ *and* $\mathsf{turn}_\star^{-1}(\cdot)$*, are defined in a way that* $\mathcal{G} \subseteq \mathsf{shuf}_*^{-1}(\mathcal{G})$ *and* $\mathcal{G} \subseteq \mathsf{turn}_\star^{-1}(\mathcal{G})$ *for any set* $\mathcal{G}$*, and from their definition it is immediate that the arising set of states is directly defined by states reachable from* $\mathcal{G}$ *in a certain way. Hence, enlarging* $\mathcal{G}$ *can only increase the set of states reachable from* $\mathcal{G}$ *(in the specified way).*

In impossibility proofs we can use the monotonicity to deliberately but carefully enlarge the sets of states, helping to simplify proofs. In the following we derive two lemmas which make calculating $\mathsf{turn}_\star^{-1}(\cdot)$ and $\mathsf{shuf}_*^{-1}(\cdot)$ easier. They are based on the simple fact that states not reachable by a turn or shuffle need not be considered when determining the respective backward calculus state sets.

Lemma 2. *Let* $\mathcal{G}$ *be a set of states, and let* $\mathsf{cc}(\mathcal{G})$ *be the subset of* $\mathcal{G}$ *with states that have a constant column. Then,* $\mathsf{turn}_\star^{-1}(\mathcal{G}) = \mathcal{G} \cup \mathsf{turn}_\star^{-1}(\mathsf{cc}(\mathcal{G}))$ *for* $\star \in \{\emptyset, \mathrm{r}, \mathrm{f}\}$.

Proof. Monotony of $\mathsf{turn}_\star^{-1}(\cdot)$, $\mathsf{cc}(\mathcal{G}) \subseteq \mathcal{G}$, and $\mathcal{G} \subseteq \mathsf{turn}_\star^{-1}(\mathcal{G})$ directly implies $\mathcal{G} \cup \mathsf{turn}_\star^{-1}(\mathsf{cc}(\mathcal{G})) \subseteq \mathsf{turn}_\star^{-1}(\mathcal{G})$. For the other direction, observe that any state μ in $\mathsf{turn}_\star^{-1}(\mathcal{G})$ (not already in $\mathcal{G}$) arises from a state $\mu' \in \mathcal{G}$ which is at one of the branches when turning a position in μ, by definition. For this we need $\mu' \in \mathsf{cc}(\mathcal{G})$. □

We derive a similar statement for reverse shuffle steps, which will be useful in the proof of Theorem 2. In the following, a state is *generateable via uniform closed shuffles*, if it has a partition that is non-strictly coarser than that of a permissible orbit partition of a subgroup Π, and *generateable via closed shuffles*, if it has a *type* partition that is non-strictly coarser than that of a permissible orbit partition of a subgroup Π.

Lemma 3. *Let* $\mathcal{G}$ *be a set of states, and let* $\mathsf{gen}_\mathrm{c}(\mathcal{G})$ *and* $\mathsf{gen}_\mathrm{uc}(\mathcal{G})$ *be the subset of* $\mathcal{G}$ *of all states that are generateable via a closed and uniform closed shuffles, respectively. Then,* $\mathsf{shuf}_*^{-1}(\mathcal{G}) = \mathcal{G} \cup \mathsf{shuf}_*^{-1}(\mathsf{gen}_*(\mathcal{G}))$, for $* \in \{\mathrm{c}, \mathrm{uc}\}$.

Proof. For $\mathcal{G} \cup \mathsf{shuf}_*^{-1}(\mathsf{gen}_*(\mathcal{G})) \subseteq \mathsf{shuf}_*^{-1}(\mathcal{G})$ observe that $\mathsf{shuf}_*^{-1}(\cdot)$ is monotone, $\mathsf{gen}_*(\mathcal{G}) \subseteq \mathcal{G}$, and $\mathcal{G} \in \mathsf{shuf}_*^{-1}(\mathcal{G})$. The other direction follows, as any state μ in $\mathsf{shuf}_*^{-1}(\mathcal{G})$ must either be in $\mathcal{G}$, or it arises from a state $\mu' \in \mathcal{G}$ such that there is a shuffle of type $*$ that produces μ' from μ. Let Π be the permutation group of such a shuffle. We obtain generateablility of μ' via a shuffle of type $*$ with group Π directly from Lemma 1. □

4 IMPOSSIBILITY OF CLOSED-SHUFFLE COPY WITH $2N + 1$ CARDS

This section contains our main result. We apply the techniques from the previous section to show a strong impossibility result for closed-shuffle COPY protocols with $2n + 1$ cards, where $n \geq 2$. This implies that in this setting, the protocol of [14] using $2n + 2$ cards is optimal.

Theorem 1. *There is no (possibilistically secure)* 2*-color* $(2n + 1)$*-card n-COPY protocol using only closed shuffles.*

Proof. We proceed by using the backwards calculus technique as developed in Section 3, i.e., we would like to show that the start state is not contained in the closure $\mathsf{cl}_\mathrm{c}(\mathcal{G}_0)$, where $\mathcal{G}_0$ is the set of final states in $(2n + 1)$-card n-COPY protocols. For this, we apply $\mathsf{turn}^{-1}(\cdot)$ and $\mathsf{shuf}_\mathrm{c}^{-1}(\cdot)$ operations to the growing set of states, starting from $\mathcal{G}_0$, until this process becomes stationary. Our analysis will show that this already happens after one reverse turn and one reverse shuffle step. We assume w.l.o.g. that the helping card is $\heartsuit$, yielding the deck $\mathcal{D} = [(n + 1) \cdot \heartsuit, n \cdot \clubsuit]$. Let $\mathcal{G}_0$ be the set of *final states for n-COPY* and note that these look in reduced form (up to similarity of states) like this:

$$\begin{array}{ll} \heartsuit(\clubsuit\heartsuit)^n & 0 \\ \heartsuit(\heartsuit\clubsuit)^n & 1. \end{array}$$

The start state of an n-COPY protocol on deck $\mathcal{D}$ looks w.l.o.g. (up to similarity of states) as follows:

$$\begin{array}{ll} \clubsuit\heartsuit(\heartsuit\clubsuit)^{n-1}\heartsuit & 0 \\ \heartsuit\clubsuit(\heartsuit\clubsuit)^{n-1}\heartsuit & 1 \end{array}$$

As in [4, Theorem 3], we know that final states are not reachable by a shuffle, because there is only one 0-sequence and only one 1-sequence, and we cannot take any proper subset of these. Hence, let us look at the set $\mathcal{G}_1 := \mathsf{turn}^{-1}(\mathcal{G}_0)$, i.e. the states that are turnable and have a state from $\mathcal{G}_0$ at one of its branches.

The newly added states, i.e. the states from $\mathcal{G}_1 \setminus \mathcal{G}_0$, look (up to similarity) as follows:

$$\begin{array}{rl} f'_0\colon \heartsuit(\clubsuit\heartsuit)^n & 0 \\ f'_1\colon \heartsuit(\heartsuit\clubsuit)^n & 1 \\ \clubsuit\ldots\ldots & ? \\ \vdots\quad & ? \\ \clubsuit\ldots\ldots & ?, \end{array}$$

where ? can be of type 0, 1, $\perp$, and we have at least one sequence starting with $\clubsuit$ (if it is a $\perp$-sequence; otherwise we have at least one 0- and one 1-sequence, due to turnability). Let us call them *prefinal*. As they have at least three sequences, the start state is not among the prefinal states. Because they do not have a constant column, a new reverse turn does not yield any additional states, i.e., $\mathcal{G}_1 = \mathsf{turn}^{-1}(\mathcal{G}_1)$. Let us denote the 0-sequence in the first row by f'_0, and the 1-sequence in the second row by f'_1.

For the main step of the proof, let $\mathcal{G}_2 := \mathsf{shuf}_\mathsf{c}^{-1}(\mathcal{G}_1) = \mathsf{shuf}_\mathsf{c}^{-1}(\mathcal{G}_1 \setminus \mathcal{G}_0)$. These are the states that reach a prefinal state via a closed shuffle. Note that if f'_0 and f'_1 is the only 0- and 1-sequence, resp. (in a state $\mu' \in \mathcal{G}_1$), all other sequences need to be of type $\perp$ and any subset of the state (i.e. a $\mu \in \mathcal{G}_2$, as in a reverse shuffle) need to contain at least f'_0, f'_1. These states are again prefinal or final.

Hence, we assume w.l.o.g. that there are at least two 0-sequences (i.e., the type partition of type 0 has more than one sequence), and assume that the shuffle step creates one of these. More formally, let Π be the shuffle subgroup from a state $\mu \in \mathcal{G}_2$ to a prefinal state $\mu' \in \mathcal{G}_1 \setminus \mathcal{G}_0$ as specified above, such that μ' contains at least one additional 0-sequence, which we can also w.l.o.g. assume to be f'_0. Let $f_0 \in \mu$, such that there is a $\tilde{\pi} \in \Pi$ with $\tilde{\pi}(f_0) = f'_0$, and let us consider the state $\tilde{\mu} := \tilde{\pi}(\mu)$ that is similar to μ but contains f'_0. Due to the closedness of the shuffle, it is retained that $\tilde{\mu}$ shuffles to μ' via Π. We can deduce that $\tilde{\mu}$ looks as follows:

$$\begin{array}{ll} \heartsuit(\clubsuit\heartsuit)^n & 0 \\ \clubsuit\ldots\ldots & ? \\ \vdots & ? \\ \clubsuit\ldots\ldots & ?, \end{array}$$

where none of the other rows contains a $\heartsuit$ in the first position. Assume the contrary, namely that there is a sequence $s \neq f'_0$, with a $\heartsuit$ in the first position. If it would be of type 0 or $\perp$, then this does not shuffle into μ', as $\mathsf{id} \in \Pi$ and there is no other 0-sequence in μ'. Otherwise, an s of type 1 would mean that either $\tilde{\mu}$ is identical to μ' (in the case that $s = f'_1$), or similarly to before this does not shuffle into μ' via Π. Hence, the form is as specified above.

Note that another reverse shuffle does not yield any new states. This is because the states we described did not assume anything about the concrete orbit partition, and taking another subset of the sequences does not help.

Hence, it remains to show that we cannot reach the states in $\mathcal{G}_2 \setminus \mathcal{G}_1$ via a turn, i.e. that $\mathcal{G}_2 = \mathsf{turn}^{-1}(\mathcal{G}_2)$. We proceed by showing that the newly added states from $\mathcal{G}_2 \setminus \mathcal{G}_1$ do not have a constant column. For this, let us assume the contrary, namely there is a constant column, at position $p \neq 1$. Let us distinguish two cases by the parity of p:

Case 1: $p = 2\ell + 2$ **is even.** We get a constant $\clubsuit$ column, and the state $\tilde{\mu}$ looks like this (with at least 2 sequences):

$$\begin{array}{rl} f'_0\colon \heartsuit(\clubsuit\heartsuit)^{\ell}\clubsuit\heartsuit(\clubsuit\heartsuit)^{n-\ell-1} & 0 \\ f_1\colon \clubsuit\ldots\ldots\clubsuit\ldots\ldots\ldots\ldots\ldots & 1 \\ \vdots\qquad & ? \\ \clubsuit\ldots\ldots\clubsuit\ldots\ldots\ldots\ldots\ldots & ? \end{array}$$

Let $f_1 \in \tilde{\mu}$ be a sequence that is mapped to $f'_1 = \heartsuit(\heartsuit\clubsuit)^n \in \mu'$ by some $\pi \in \Pi$. As indicated, because of the constant column, f_1 has at least two $\clubsuit$s which are in positions, where there is a $\heartsuit$ in f'_1, namely positions 1 and p. Let us look at π^{-1}, which is also in Π due to the closedness of the shuffle and which maps f'_1 to f_1.

For the general proof idea, we show that π^{-1} needs to have certain features, which imply that it maps *the* 0-*sequence* f'_0 to a sequence of type 0 with a $\heartsuit$ at positions 1 *and* p, leading to the contradiction, as the only 0-sequence with a $\heartsuit$ in the first position has to be f'_0, which has a $\clubsuit$ at even positions.

For this, note that to get $\clubsuit$s into positions 1 and p via π^{-1}, there need to be odd positions $u, v \neq 1$ (which contain $\clubsuit$ in $f_1' = \heartsuit(\heartsuit\clubsuit)^n$), such that $\pi^{-1}(u) = p$, and $\pi^{-1}(v) = 1$. Let us look at the sequence $s' = \pi^{-1}(f_0')$. f_0' has a $\heartsuit$ at all the odd positions, hence, s' does have a $\heartsuit$ in positions 1 and p, leading to the contradiction as discussed above.

Case 2: $p = 2\ell + 3$ **is odd.** In this case we get a constant $\heartsuit$ column, and the state looks like this:

$$\begin{array}{rll} f_0'\colon & \heartsuit(\clubsuit\heartsuit)^{\ell}\clubsuit\heartsuit(\clubsuit\heartsuit)^{n-\ell-1} & 0 \\ f_1\colon & \clubsuit\ldots\ldots\heartsuit\ldots\ldots\ldots\ldots & 1 \\ & \qquad\qquad\vdots & ? \\ & \clubsuit\ldots\ldots\heartsuit\ldots\ldots\ldots\ldots & ? \end{array}$$

Our argumentation is as above, but more involved, as f_0' has a $\heartsuit$ also in the first position. As before let $f_1 \in \tilde{\mu}$ be a sequence that is mapped to $f_1' = \heartsuit(\heartsuit\clubsuit)^n \in \mu'$ by some $\pi \in \Pi$. Here, f_1 has a $\clubsuit$ at the first position which needs to be a $\heartsuit$ in f_1', and a $\heartsuit$ at the odd position p, which should become a $\clubsuit$ through π. Again, $\pi^{-1} \in \Pi$ and maps f_1' to f_1.

To fulfill this, we need a position u which is even or 1, such that $\pi^{-1}(u) = p$, and an odd position $v \neq 1$, such that $\pi^{-1}(v) = 1$. Here again, let us analyze $s' = \pi^{-1}(f_0')$. s' is a 0-sequence with a $\heartsuit$ in the first position. If u would be even, then s' has a $\clubsuit$ in position p, already leading to the same contradiction as above. Hence, $u = 1$. Then, for prefinality of μ', $s' = f_0'$ (otherwise, we would have two 0-sequences, starting with $\heartsuit$), and $\pi, \pi^{-1} \in \mathsf{Stab}_{f_0'}(S_{2n+1})$.

In this case, we can deduce that $f_1 = \clubsuit(\heartsuit\clubsuit)^{\ell}\heartsuit\heartsuit(\heartsuit\clubsuit)^{n-\ell-1}$, i.e. that the state $\tilde{\mu}$ looks more closely as follows:

$$\begin{array}{rll} f_0'\colon & \heartsuit(\clubsuit\heartsuit)^{\ell}\clubsuit\heartsuit(\clubsuit\heartsuit)^{n-\ell-1} & 0 \\ f_1\colon & \clubsuit(\heartsuit\clubsuit)^{\ell}\heartsuit\heartsuit(\heartsuit\clubsuit)^{n-\ell-1} & 1 \\ & \clubsuit\ldots\ldots\heartsuit\ldots\ldots\ldots\ldots & ? \\ & \qquad\qquad\vdots & ? \\ & \clubsuit\ldots\ldots\heartsuit\ldots\ldots\ldots\ldots & ?, \end{array}$$

where we have at least three sequences, as otherwise we would have a final state.

By [4, Lemma 1], because $\pi^{-1}(1) = p$ as described above, after shuffling with Π, we obtain the same number of $\heartsuit$, and $\clubsuit$ in columns 1 and p in the resulting prefinal state μ'. Because the prefinal state has exactly 2 $\heartsuit$s in the first column, the same holds for column p in the prefinal states reachable from $\tilde{\mu}$ via Π. The same has to hold for $\tilde{\mu}$, i.e. in column p only two $\heartsuit$ are allowed. (To see this, note that Π contains id and, hence, the sequences in $\tilde{\mu}$ form a subset of the sequences of μ'). But by assumption this is the constant column which may only contain $\heartsuit$s. Hence, there would be at most two sequences in $\tilde{\mu}$. As discussed before, this would then be already final.

As the start state has a constant column, contrary to the states in $\mathcal{G}_2 \setminus \mathcal{G}_0$, it is not contained in the closure $\mathsf{cl}_c(\mathcal{G}_0)$. Hence, no final state is reachable from the start state. □

5 A UNIFORM CLOSED AND PROTOCOL REQUIRES FIVE CARDS

In this section we generalize the impossibility result regarding four-card AND protocols restricted to uniform closed shuffles of [4, Theorem 2] to restarting protocols. This, together with the impossibility result of Section 4 and the four-card restart-free AND protocol using only uniform (non-closed) shuffles, shows that *restarts are unnecessary* for realizing AND and COPY with a minimal number of cards. This shows that allowing restarts is relatively powerless, except possibly for protocols that directly compute more complex Boolean functions.

The proof demonstrates the developed backward calculus and how to deal with $\perp$-sequences in impossibility arguments. Before we start, let us note some general facts about orbit partitions on the four-card deck:

Proposition 1. *Let $\mathcal{D} = [\heartsuit, \heartsuit, \clubsuit, \clubsuit]$, X the set of all sequences on $\mathcal{D}$ and Π a non-trivial subgroup of S_4. Then,*

1. *the size of the stabilizer on an $x \in X$ is at most* 4.
2. *if there is an orbit of size* 1*, then there are exactly two orbits of size* 1*. In this case, the corresponding sequences have distance* 4.
3. *any orbit partition of X via Π has set sizes i)* $(1, 1, 2, 2)$*, ii)* $(1, 1, 4)$*, iii)* $(3, 3)$*, iv)* $(4, 2)$ *or v)* (6)*. Note that i) corresponds to a partition of maximal fineness. (In total, this upper-bounds the number of orbits to* 4*.)*

Proof. The proof proceeds by a close analysis and is given in Appendix B. □

Using this, and the techniques from Section 3 we can now prove the following theorem.

Theorem 2. *There is no (restarting) committed format four-card AND protocol using only uniform closed shuffles.*

Proof. As before, we start by iteratively expanding the set of good states, starting from the final states, and in each step adding all the states that are able to reach the good-states-so-far by a shuffle or turn, to the set. We show that this process terminates, and the resulting set does not include the start state, showing that no final state is reachable at all.

Note that, contrary to the proof of [4, Theorem 2], we need to take $\perp$-sequences into account, which complicates the proof. Moreover, it is important to make some use of both the uniformity and the closedness of the shuffles, hence to look at the more concrete probabilities of the states, as there are four-card protocols in the closed-shuffle setting ([10, Sect. 4]), and in the uniform-shuffle setting (Theorem 3). We want to avoid working with very concrete probabilities, for reasons of simplicity, hence we make only use of the type-annotated partition of the states, which already carries a lot of information, as introduced in the form the semi-reduced variants of states in Section 2.

Moreover, for $\perp$-sequences we may distinguish two versions, namely those, which are assigned a constant probability, by $\perp^{c}$, and those which are assigned a non-constant (but result-mixed) probability, by $\perp^{nc}$. For example, $1/3(X_{00} + X_{01} + X_{10} + X_{11})$ is of type $\perp^{c}$, where as $1/3 X_{00} + 2/3 X_{11}$ would be of type $\perp^{nc}$. We write $\perp$ if we do not differentiate the types. Note that as our deck is $[\heartsuit, \heartsuit, \clubsuit, \clubsuit]$, we can form at most 6 sequences shown in Fig. 3a.

Figure 3:

$$\begin{array}{cccc}
\heartsuit & \heartsuit & \clubsuit & \clubsuit \\
\heartsuit & \clubsuit & \heartsuit & \clubsuit \\
\clubsuit & \heartsuit & \heartsuit & \clubsuit \\
\heartsuit & \clubsuit & \clubsuit & \heartsuit \\
\clubsuit & \heartsuit & \clubsuit & \heartsuit \\
\clubsuit & \clubsuit & \heartsuit & \heartsuit
\end{array}$$

(a) All sequences with four cards.

$$\begin{array}{ccccc}
\heartsuit & \clubsuit & \heartsuit & \clubsuit & 1 \\
\heartsuit & \clubsuit & \clubsuit & \heartsuit & 1 \\
\clubsuit & \heartsuit & \heartsuit & \clubsuit & 0 \\
\clubsuit & \heartsuit & \clubsuit & \heartsuit & 0
\end{array}$$

(b) A reduced, sequence-maximal final state with four cards, where the first two positions encode the result.

Final and Prefinal States. The state of Fig. 3b is (up to similarity) the largest final state. Up to similarity, we obtain all final states by choosing a subset of the sequences of this state with the restriction of having at least one 1- and one 0-sequence. The set of final states is denoted by $\mathcal{G}_0$. It is easy to see that the start state is not already final.

As $\mathcal{G}_0$ does not contain any states with a $\perp$-sequence, but each state has at least one 1- and one 0-sequence, any state which shuffles into $\mathcal{G}_0$ contains a subset of the sequences of a state in $\mathcal{G}_0$, with the same restriction of having at least one 1- and one 0-sequence, as we cannot create 0-/1-sequences by a shuffle out of nothing. Hence, $\mathsf{shuf}^{-1}(\mathcal{G}_0) = \mathcal{G}_0$. Therefore, the prefinal states are the non-final states that reach the set of final states by a turn. W.l.o.g. they look as follows:

$$\begin{array}{ccccc}
\heartsuit & \clubsuit & \clubsuit & \heartsuit & 1 \\
\clubsuit & \heartsuit & \clubsuit & \heartsuit & 0 \\
\hline
\clubsuit & \heartsuit & \heartsuit & \clubsuit & ? \\
\heartsuit & \clubsuit & \heartsuit & \clubsuit & ? \\
\clubsuit & \clubsuit & \heartsuit & \heartsuit & ?
\end{array}$$

where the sequences marked by ? can be any of $1, 0, \perp$ or empty, as long as there is at least one sequence present (otherwise, it would already be final). These need to be turnable at the third or fourth column, of course. This general form of the state is as described because the only final states with a constant column are the states with exactly one 1-sequence and one 0-sequence (above the rule), and they can be combined with any state with a constant position (of the other symbol) at the same index, leading to at least one and at most three additional sequences (below the rule).

Note that the start state is not a prefinal state. For this you would need to choose two of the sequences below the rule to be 0 and the other absent. But then the state is no longer turnable, because for this, there would need to be at least one additional 1- or $\perp$-sequence below the rule. For notation, set $\mathcal{G}_1 := \mathsf{turn}^{-1}(\mathcal{G}_0)$.

Second Enlargement, by reverse-turn. We enlarge the set of good states ($\mathcal{G}_1$) by the states that can reach them with a turn operation. Denote these by $\mathcal{G}_2 := \mathsf{turn}^{-1}(\mathcal{G}_1)$. These look as follows:

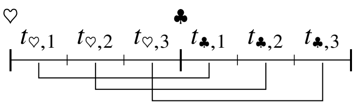
$$\begin{array}{ccccc}
\heartsuit & \clubsuit & \clubsuit & \heartsuit & 1 \\
\clubsuit & \heartsuit & \clubsuit & \heartsuit & 0 \\
\hline
\clubsuit & \clubsuit & \heartsuit & \heartsuit & \perp^{c} \\
\hline
\heartsuit & \clubsuit & \heartsuit & \clubsuit & ? \\
\clubsuit & \heartsuit & \heartsuit & \clubsuit & ? \\
\heartsuit & \heartsuit & \clubsuit & \clubsuit & ?
\end{array}$$

Here again the sequences marked by ? can be any of $1, 0, \perp$ or empty, as long as there is at least one of the sequences present (otherwise, it would already be prefinal). These need to be turnable at the fourth column, of course. (The first three sequences constitute a state with a constant column from $\mathcal{G}_1 \setminus \mathcal{G}_0$, turnable at the third position, if looked at in isolation.) As the newly-added states have a $\perp$-sequence, they cannot be the start state. Moreover, an additional reverse-turn on $\mathcal{G}_2$ is futile, as states from $\mathcal{G}_2 \setminus \mathcal{G}_1$ all have at least four sequences, and hence no constant column.

Third Enlargement, by reverse-shuffle. The difficult part is to consider the states which will lead to any of the currently-good states ($\mathcal{G}_2$) via a uniform closed shuffle. Denote them by $\mathcal{G}_3 := \mathsf{shuf}_{\mathrm{uc}}^{-1}(\mathcal{G}_2)$. As id is present in any shuffle, these states are subsets of their resulting states, with the additional possibility of splitting two (or more) $\perp$-sequences into a 1- and a 0-sequence (and possibly more 1-, 0-, or $\perp$-sequences) beforehand. (As in the forward-process two sequences of different types can mix into a $\perp$-sequence.)

As we saw before, the final states cannot be reached by a shuffle. Hence, by Lemma 3, we only need to consider the generateable states from $\mathcal{G}_2 \setminus \mathcal{G}_0$. By Proposition 1, the orbit sizes of a finest partition that can be generated are $1, 1, 2, 2$ and there are at most 2 orbits of size 1. Also, generateable states from $\mathcal{G}_2 \setminus \mathcal{G}_0$ have at least four sequences.[6]

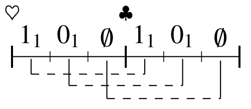
We will use the turn-split representation of a state, as given in more detail in Appendix C:

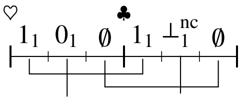


♡
♣
$t_{\heartsuit,1}$ $t_{\heartsuit,2}$ $t_{\heartsuit,3}$ $t_{\clubsuit,1}$ $t_{\clubsuit,2}$ $t_{\clubsuit,3}$

where $t_{\heartsuit,1}, \dots, t_{\heartsuit,3}$, $t_{\clubsuit,1}, \dots, t_{\clubsuit,3}$ are types of sequences denoted by $s_{\heartsuit,1}, \dots, s_{\heartsuit,3}$, $s_{\clubsuit,1}, \dots, s_{\clubsuit,3}$, the first three belonging to the $\heartsuit$-branch of a possible turn (the state was assumed to be turnable), and the last three at the $\clubsuit$-branch of a turn. The connecting brackets between the sequences (from which we abstract by this drawing) represent a possible orbit decomposition of a shuffle, namely connected sequences are in the same orbit. A single line drawn downward will mean that the sequence is in an orbit of size 1. By Proposition 1, we know that we cannot have both sequences in an orbit of size 1 *on the same side of the turn*, as they would not have distance 4 then.

We think it is instructive to take a closer look at the possible generateable states from $\mathcal{G}_2 \setminus \mathcal{G}_0$. Note that all of these are turnable. Let us do a case distinction on the number of sequences in a state and start with the minimum of *four sequences*. Using our abstract notation observe that we are in one of the following situations:

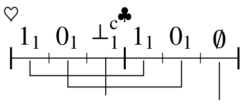




♡
♣
1_1 0_1 $\emptyset$ 1_1 $\perp_1^{\mathrm{nc}}$ $\emptyset$

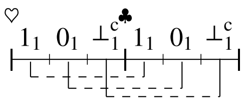
(The dashed lines on the left indicate that we can arbitrarily choose one of the orbits into two, as by Proposition 1 having exactly three orbits of size 2 is impossible in our setting). This is because turnability requires $\mu(s_{\heartsuit,1}) + \mu(s_{\heartsuit,2}) = p$ and $\mu(s_{\clubsuit,1}) + \mu(s_{\clubsuit,2}) = 1 - p$, for $p \in [0, 1]$, where $s_{\heartsuit,1}, s_{\heartsuit,2}$ are sequences in one part of the turn branch, and $s_{\clubsuit,1}, s_{\clubsuit,2}$ in the other. Moreover, we can only choose one of the two orbit combinations (where on the right the 0 was w.l.o.g. be chosen to be in the orbit with only one element). Hence, the orbit partition implies that $\mu(s_{\heartsuit,1}) = \mu(s_{\clubsuit,1})$, *or* $\mu(s_{\heartsuit,2}) = \mu(s_{\clubsuit,2})$, let us assume w.l.o.g. the first. Then we can deduce that if $p = 1/2$, we are in the situation to the left, and if $p < 1/2$ in the situation to the right.

In the case of *five sequences*, the only turnable and generateable states look as follows:

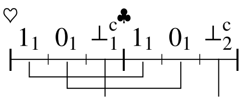


This is, similarly to the proof in [4], because we have to choose the single empty sequence to be in an orbit of size one, forcing the pairs of 1- and 0-sequences each into an orbit of size two. By turnability we have that $\mu(s_{\heartsuit,1}) + \mu(s_{\heartsuit,2}) + \mu(s_{\heartsuit,3}) = p$ and $\mu(s_{\clubsuit,1}) + \mu(s_{\clubsuit,2}) = 1 - p$ for some $p \in [0, 1]$. The orbit decomposition implies $\mu(s_{\heartsuit,1}) = \mu(s_{\clubsuit,1})$ and $\mu(s_{\heartsuit,2}) = \mu(s_{\clubsuit,2})$ which, if plugged into the second equation implies that $\mu(s_{\heartsuit,3})$ is a constant, because $\mu(s_{\heartsuit,1}) + \mu(s_{\heartsuit,2}) = 1 - p$ and $\mu(s_{\heartsuit,1}) + \mu(s_{\heartsuit,2}) + \mu(s_{\heartsuit,3}) = 1 - p + \mu(s_{\heartsuit,3}) = p \Leftrightarrow \mu(s_{\heartsuit,3}) = 2p - 1$.

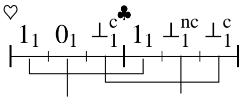
For states with *six sequences*, we are in one of the following configurations:



♡
♣
1_1 0_1 $\perp_1^{\mathrm{c}}$ 1_1 0_1 $\perp_1^{\mathrm{c}}$



♡
♣
1_1 0_1 $\perp_1^{\mathrm{c}}$ 1_1 0_1 $\perp_2^{\mathrm{c}}$



♡
♣
1_1 0_1 $\perp_1^{\mathrm{c}}$ 1_1 $\perp_1^{\mathrm{nc}}$ $\perp_1^{\mathrm{c}}$

[6] If it would have only three sequences, then states that reach such a state via a non-trivial shuffle need to have at least one sequence less. But if it has exactly two sequences (one 1-sequence, one 0-sequence), it is already final.

Note that a resulting state needs a $\perp^c$-sequence, and if a state has two or three 0-sequences and only one 1-sequence, it cannot result in a state with sequences of the types $\perp$, 1 and 0 by a uniform closed shuffle. This is because for $\perp$ you need at least one 1- and one 0-sequence which are mixed together, and one additional 1- and 0-sequence in different orbits which are not mixed together, so that you need at least two 1-sequences.

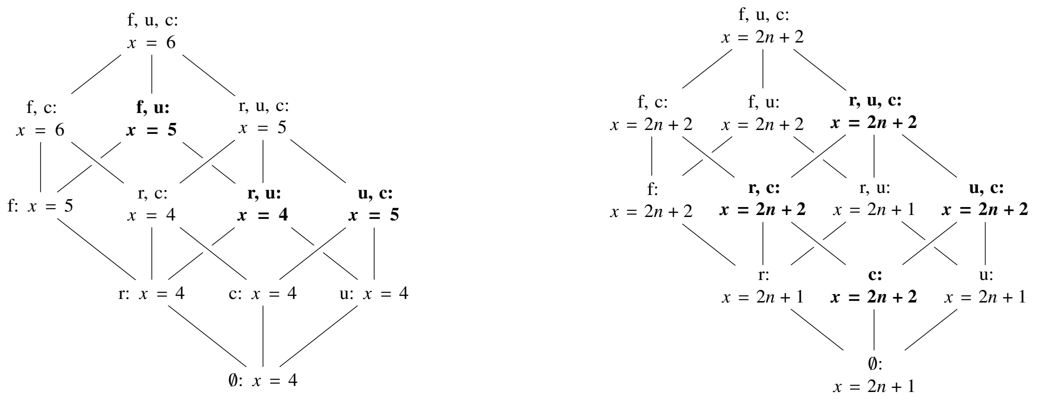
By the claim, it follows that the start state – which has three 0-sequences and one 1-sequence – would only be added to the set, if it shuffles into a good state *without a $\perp$-sequence*. For this to happen, the start state has to be a proper subset of the resulting state (cf. Lemma 5 of [4]). Given that we are in one of the scenarios as argued above, this cannot happen, as none of the states has three 0-sequences.

A Fourth Enlargement is Futile. Now let us observe that a reverse-turn does not lead to any new states. For this, note that we need to only consider states that have been added in the last enlargement. For a reverse-turn, we can only reach those which have a constant column, and hence have at most three sequences. Note that by the same argument as before, states with two 0-sequences and one 1-sequence or the other way round, *with a constant column* have not been added in the last step, as the only $\perp$-free configuration of the above list cannot have a constant column. (We did not add any new states with a constant column to $\mathcal{G}_2$.) Hence $\mathcal{G}_3 = \mathsf{turn}^{-1}(\mathcal{G}_3)$.

We want to show $\mathcal{G}_3 = \mathsf{shuf}_{\mathrm{uc}}^{-1}(\mathcal{G}_3)$. For this note that no arising state has more than two sequences of the same type (except for the three $\perp$-sequences that may arise in states with 6 sequences). The additional power of successive $\mathsf{shuf}_{\mathrm{uc}}^{-1}(\cdot)$-steps comes only into play, if there are more than two sequences of the same type, as we are only possibly restricted by the shuffle when taking a subset of the sequences. □

6 CONCLUSION

For the two central building blocks in composite card-based protocols, AND and COPY, we complete the picture on the necessary number of cards in committed format protocols with respect to all combinations of practicality requirements. With this work, all bounds are tight, as shown in Fig. 4 and Table 1. Hence, this paper provides a reference showing the best protocol for a used setting, and one can compare whether to trade fewer cards with different characteristics of the runtime or shuffles.



(a) AND protocols

(b) COPY protocols

Figure 4: The number of cards necessary and sufficient for committed format protocols for AND and COPY, given as a Hasse diagram. The nodes/corners specify the different requirement combinations: f is finite-runtime, r is restart-free LV, c and u are restrictions to closed and uniform shuffles, respectively. A line between two nodes in the lattice specifies that the configuration above has more restrictions than below. New results are in bold.

Using our developed technique to impossibility results, we can interpret other results on lower bounds from the literature in our framework: In [10, Sect. 6], the set of good states is a superset of $\mathsf{cl}_{\mathrm{f}}(\mathcal{G}_0)$, where $\mathcal{G}_0$ is as in Theorem 2. More exactly, $\mathsf{cl}_{\mathrm{f}}(\mathcal{G}_0) = \mathsf{turn}_{\mathrm{f}}^{-1}(\mathcal{G}_0) =: \mathcal{G}_1$, i.e. the process stops after one step and $\mathsf{shuf}^{-1}(\mathcal{G}_1) = \mathsf{turn}_{\mathrm{f}}^{-1}(\mathcal{G}_1) = \mathcal{G}_1$. The same holds for the proof of [4, Sect. 7], i.e. $\mathsf{cl}_{\mathrm{r,uc}}(\mathcal{G}_0) = \mathsf{turn}_{\mathrm{r}}^{-1}(\mathcal{G}_0) =: \mathcal{G}_1'$ and the proof stops after one step as $\mathsf{shuf}_{\mathrm{uc}}^{-1}(\mathcal{G}_1') = \mathsf{turn}_{\mathrm{r}}^{-1}(\mathcal{G}_1') = \mathcal{G}_1'$. Regarding the lower bounds for COPY in [4, Sects. 8, 9], the process stops even earlier. Both impossibility proofs show that the set of final states constitutes already the backward calculus closure $\mathsf{cl}_{\mathrm{f}}(\cdot)$ and $\mathsf{cl}(\cdot)$, respectively. This comparison explains the relative complexity of our new proofs. More specifically, our proof in Section 4 shows that $\mathsf{cl}_{\mathrm{c}}(\mathcal{G}) = \mathsf{shuf}_{\mathrm{c}}^{-1}(\mathsf{turn}^{-1}(\mathcal{G}))$ for the set of n-COPY final states $\mathcal{G}$ on $2n + 1$ cards. Likewise, the proof in Section 5 shows that $\mathsf{cl}_{\mathrm{uc}}(\mathcal{G}) = \mathsf{shuf}_{\mathrm{uc}}^{-1}(\mathsf{turn}^{-1}(\mathsf{turn}^{-1}(\mathcal{G})))$ for 4-card AND final states $\mathcal{G}$.

Acknowledgments. I would like to thank Stefan Walzer for invaluable discussions, and Akin Ünal and anonymous reviewers for their helpful comments. The ideas were partly developed during a research stay at NTT Secure Platform Laboratories in Tokyo, Japan, kindly made possible by Masayuki Abe and Ryo Nishimaki.

REFERENCES

[1] Yuta Abe, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. "Five-Card AND Protocol in Committed Format Using Only Practical Shuffles". In: *APKC@AsiaCCS 2018*. Ed. by Keita Emura, Jae Hong Seo, and Yohei Watanabe. ACM, 2018, pp. 3–8. DOI: 10.1145/3197507.3197510.

[2] Bert den Boer. "More Efficient Match-Making and Satisfiability: The Five Card Trick". In: *EUROCRYPT '89*. Ed. by Jean-Jacques Quisquater and Joos Vandewalle. LNCS 434. Springer, 1989, pp. 208–217. DOI: 10.1007/3-540-46885-4_23.

[3] John D. Dixon and Brian Mortimer. *Permutation groups*. Graduate texts in mathematics; 163. New York: Springer, 1996.

[4] Julia Kastner, Alexander Koch, Stefan Walzer, Daiki Miyahara, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. "The Minimum Number of Cards in Practical Card-based Protocols". In: *ASIACRYPT 2017*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. LNCS 10626. Springer, 2017, pp. 126–155. DOI: 10.1007/978-3-319-70700-6_5.

[5] Alexander Koch. *The Landscape of Optimal Card-based Protocols*. Oct. 9, 2018. Cryptology ePrint Archive, Report 2018/951.

[6] Alexander Koch. "Cryptographic Protocols from Physical Assumptions". PhD thesis. Karlsruhe: Karlsruhe Institute of Technology (KIT), 2019. DOI: 10.5445/IR/1000097756.

[7] Alexander Koch, Michael Schrempp, and Michael Kirsten. "Card-Based Cryptography Meets Formal Verification". In: *ASIACRYPT 2019,* Proceedings, Part I. Ed. by Steven D. Galbraith and Shiho Moriai. LNCS. Springer, Nov. 25, 2019, pp. 488–517. DOI: 10.1007/978-3-030-34578-5_18.

[8] Alexander Koch, Michael Schrempp, and Michael Kirsten. "Card-based Cryptography Meets Formal Verification". In: *New Gener. Comput.* 39 (Special Issue on Card-Based Cryptography 2021), pp. 115–158. DOI: 10.1007/s00354-020-00120-0.

[9] Alexander Koch and Stefan Walzer. "Foundations for Actively Secure Card-based Cryptography". In: *Fun with Algorithms, FUN 2021*. Ed. by Martin Farach-Colton, Giuseppe Prencipe, and Ryuhei Uehara. LIPIcs 157. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020, 17:1–17:23. DOI: 10.4230/LIPIcs.FUN.2021.17.

[10] Alexander Koch, Stefan Walzer, and Kevin Härtel. "Card-based Cryptographic Protocols Using a Minimal Number of Cards". In: *ASIACRYPT 2015*. Ed. by Tetsu Iwata and Jung Hee Cheon. LNCS 9452. Springer, 2015, pp. 783–807. DOI: 10.1007/978-3-662-48797-6_32.

[11] Takaaki Mizuki. "Efficient and Secure Multiparty Computations Using a Standard Deck of Playing Cards". In: *CANS 2016*. Ed. by Sara Foresti and Giuseppe Persiano. LNCS 10052. 2016, pp. 484–499. DOI: 10.1007/978-3-319-48965-0_29.

[12] Takaaki Mizuki and Hiroki Shizuya. "A formalization of card-based cryptographic protocols via abstract machine". In: *International Journal of Information Security* 13.1 (2014), pp. 15–23. DOI: 10.1007/s10207-013-0219-4.

[13] Takaaki Mizuki and Hiroki Shizuya. "Computational Model of Card-Based Cryptographic Protocols and Its Applications". In: *IEICE Transactions* 100-A.1 (2017), pp. 3–11. URL: https://search.ieice.org/bin/summary.php?id=e100-a_1_3.

[14] Takaaki Mizuki and Hideaki Sone. "Six-Card Secure AND and Four-Card Secure XOR". In: *FAW 2009*. Ed. by Xiaotie Deng, John E. Hopcroft, and Jinyun Xue. LNCS 5598. Springer, 2009, pp. 358–369. DOI: 10.1007/978-3-642-02270-8_36.

[15] Valtteri Niemi and Ari Renvall. "Solitaire Zero-knowledge". In: *Fundam. Inform.* 38.1-2 (1999), pp. 181–188. DOI: 10.3233/FI-1999-381214.

[16] Takuya Nishida, Takaaki Mizuki, and Hideaki Sone. "Securely Computing the Three-Input Majority Function with Eight Cards". In: *TPNC 2013*. Ed. by Adrian Horia Dediu, Carlos Martín-Vide, Bianca Truthe, and Miguel A. Vega-Rodríguez. LNCS 8273. Springer, 2013, pp. 193–204. DOI: 10.1007/978-3-642-45008-2_16.

[17] Akihiro Nishimura, Takuya Nishida, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. "Card-based protocols using unequal division shuffles". In: *Soft Comput.* 22.2 (2018), pp. 361–371. DOI: 10.1007/s00500-017-2858-2.

[18] Suthee Ruangwises and Toshiya Itoh. "AND Protocols Using Only Uniform Shuffles". In: *ArXiv e-prints* (Oct. 2, 2018). ID: 1810.00769 [cs.CR].

[19] Suthee Ruangwises and Toshiya Itoh. “AND Protocols Using only Uniform Shuffles”. In: *Computer Science – Theory and Applications*. Proceedings: CSR 2019. Ed. by René van Bevern and Gregory Kucherov. LNCS. Springer, 2019, pp. 349–358. DOI: 10.1007/978-3-030-19955-5_30.

A CARD-MINIMAL AND PROTOCOLS WITH UNIFORM SHUFFLES

As stated in Section 1, these protocols were given concurrently and independently by Ruangwises and Itoh [19].

Theorem 3. *There is a secure four-card restart-free Las Vegas protocols for* AND *using only uniform shuffles.*

The protocol is given in Fig. 5. It is a slight modification of the protocol of [10], where the non-uniform but closed shuffle was replaced by a uniform but non-closed shuffle with the same effect.

Using this modification, we can tweak the idea further to also get five-card finite-runtime protocol with uniform shuffles. This is because the five-card protocol from [10] is an adaptation of its four-card protocol, where at one point of the protocol the additional card is used, via a non-uniform non-closed shuffle to break out of the loop. Hence, we have the following.

Theorem 4. *There is a secure five-card finite-runtime protocol for* AND *using only uniform shuffles.*

Figure 6 shows the five-card protocol. It is a slight modification of the protocol of [10], where the non-uniform (non-closed) shuffle was replaced by a uniform (non-closed) shuffle on the left branch, and the same replacement as in Fig. 5 was performed on the right branch.

B GROUP ACTIONS AND A FULL PROOF OF PROPOSITION 1

Let us prepend this section with a brief formal definition of group actions and the orbit-stabilizer theorem, which are assumed to be known in the rest of the paper.

Definition 3 (Group action, e.g. [3, Sect. 1.3])**.** *Let X be a nonempty set, G a group, and $\varphi\colon G \times X \to X$ a function given as $g(x) := \varphi(g, x)$ for $g \in G, x \in X$. G* acts *on X, or G is a* group action *on X if $\mathsf{id}(x) = x$ for all $x \in X$, where id denotes the neutral element in G, and $(g \circ h)(x) = g(h(x))$ for all $x \in X$ and all $g, h \in G$.*

The following lemma is a well-known fact about orbit partitions. A proof is given, e.g., in [3, Thm. 1.4A].

Lemma 4 (Orbit-Stabilizer theorem)**.** *Let G be a group, X a nonempty set and let G act on X. Then, $|G(x)| = |G|/|\mathsf{Stab}_x(G)|$ for all $x \in X$.*

We can now restate the proposition and give present its formal proof.

Proposition 1. *Let $\mathcal{D} = [\heartsuit, \heartsuit, \clubsuit, \clubsuit]$, X the set of all sequences on $\mathcal{D}$ and Π a non-trivial subgroup of S_4. Then,*

1. *the size of the stabilizer on an $x \in X$ is at most 4.*
2. *if there is an orbit of size 1, then there are exactly two orbits of size 1. In this case, the corresponding sequences have distance 4.*
3. *any orbit partition of X via Π has set sizes i) $(1, 1, 2, 2)$, ii) $(1, 1, 4)$, iii) $(3, 3)$, iv) $(4, 2)$ or v) (6). Note that i) corresponds to a partition of maximal fineness. (In total, this upper-bounds the number of orbits to 4.)*

Proof.

1. Let us first show that $|\mathsf{Stab}_s(S_4)| = 4$, the statement then follows from $\mathsf{Stab}_s(\Pi) \subseteq \mathsf{Stab}_s(S_4)$. For this, note that for any sequence on $\mathcal{D}$ there are exactly four permutations that leave the sequence fixed, namely id, the swap of the two $\heartsuit$, the swap of the two $\clubsuit$, and the combination of both.
2. For this, assume there is an orbit of size 1, inhabited by a sequence $s \in X$. This implies that Π is a subset of the stabilizer $\mathsf{Stab}_s(S_4)$ of s. Determine the unique sequence $\overline{s}$ of maximum distance 4 by exchanging $\heartsuit$ and $\clubsuit$, and note that it has the same stabilizer, i.e. $\mathsf{Stab}_s(S_4) = \mathsf{Stab}_{\overline{s}}(S_4)$. We can infer that $\overline{s}$ also has an orbit of size 1.

   By this argument, we can deduce that the number of orbits of size 1 is even. Assume there are two distinct sequences s_1, s_2, such that $s_2 \neq \overline{s_1}$. Hence, $s_1, s_2, \overline{s_1}, \overline{s_2}$ are four distinct sequences and s_1, s_2 have distance 2. Now observe that the stabilizers of s_1 and s_2 have trivial intersection, i.e. $\mathsf{Stab}_{s_1}(S_4) \cap \mathsf{Stab}_{s_2}(S_4) = \{\mathsf{id}\}$. As above, note that Π is a subset of *both* stabilizers, leading to $\Pi \subseteq \{\mathsf{id}\}$, which contradicts the assumption that Π was non-trivial. This restricts the number of orbits of size 1 to 2.

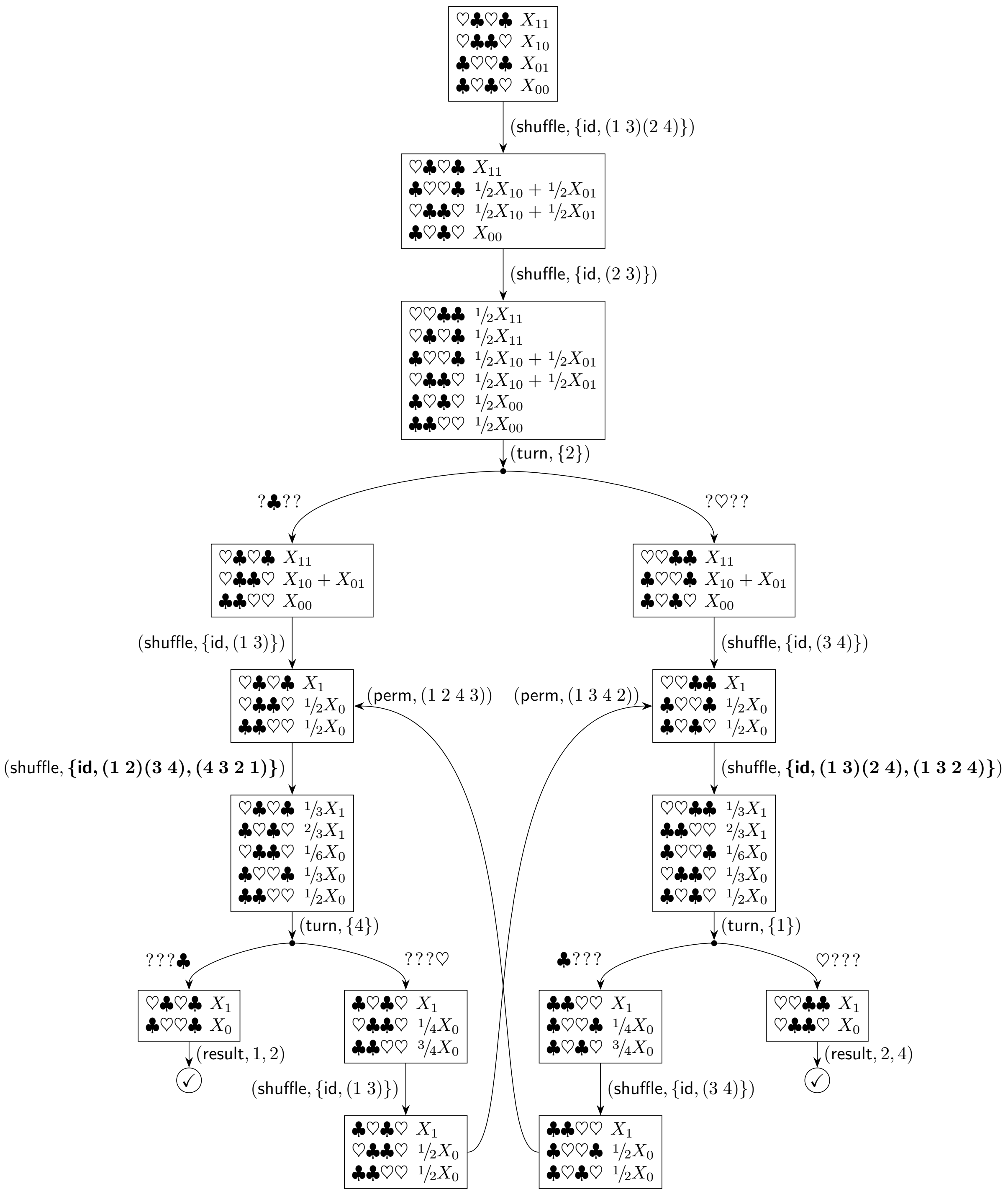


Figure 5: A modification of the four-card AND protocol from [10], where the shuffle after the second state of the protocol after branching, has been replaced by a uniform non-closed shuffle operation. Note that each turned card is implicitly turned back again.

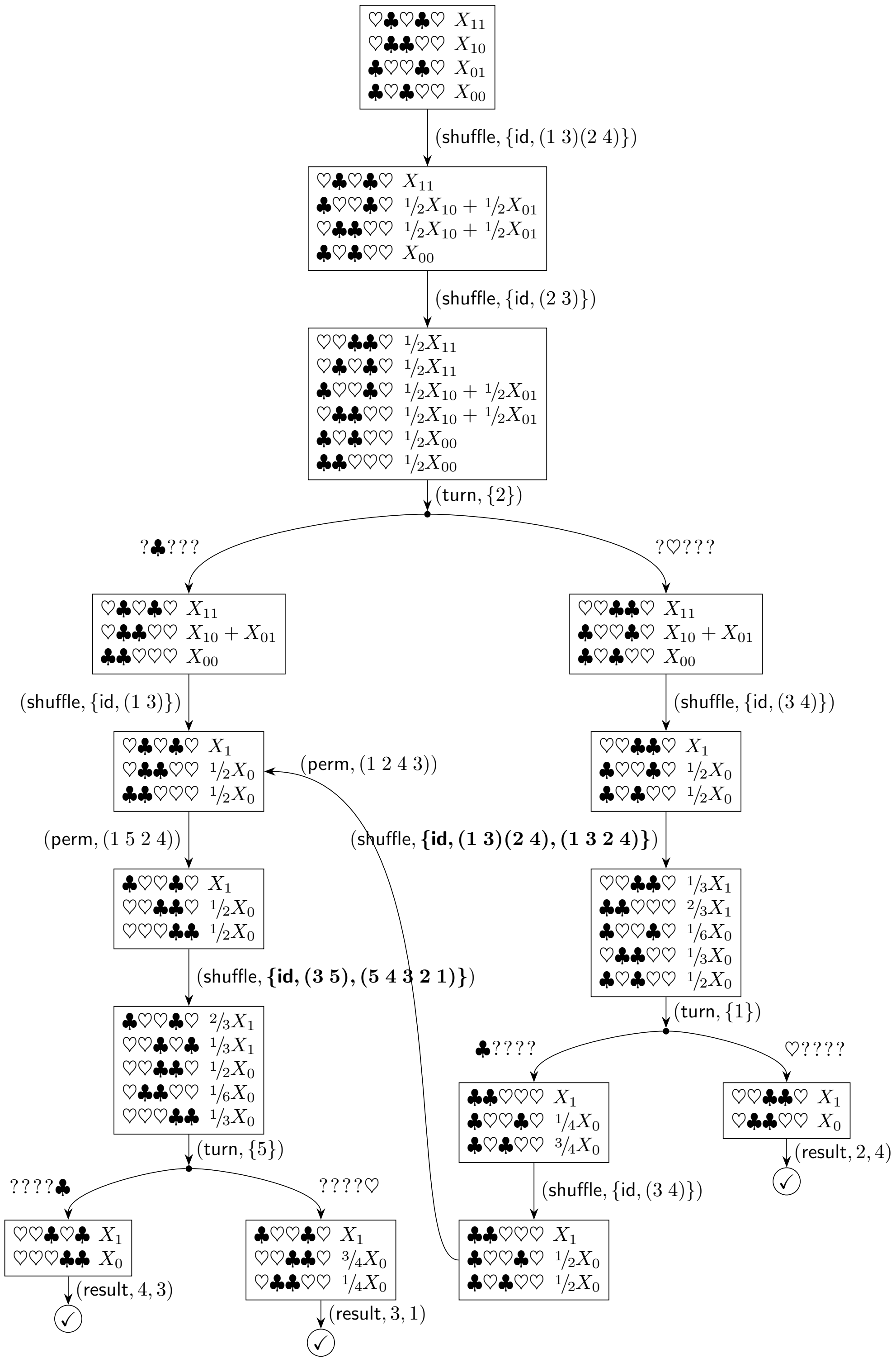


Figure 6: Finite-runtime five-card AND protocol, changed shuffles are in bold. Note that each turned card is implicitly turned back again.

3. For the third statement, observe that because of item 2 it remains to exclude the case $(2, 2, 2)$ and to show that all five orbit set sizes are attainable by giving an example. Let us assume for a contradiction that the orbit partition has set sizes $(2, 2, 2)$. Then, it holds for all $s \in X$ and all $\pi \in \Pi$ that $\pi^2(s) \in \{s, \pi(s)\}$ and hence $\pi^2(s) = s$. This means that in the cycle decomposition of any $\pi \in \Pi$ there cannot be cycles of length 3 or larger, and hence any $\pi \in \Pi$ has at most order 2. Then, Π is a subset of $\langle(1\ 2), (3\ 4)\rangle$ (up to joint conjugation of the generators) of size 2 or 4. Thus, $\{\heartsuit\heartsuit\clubsuit\clubsuit\}$ is an orbit of size 1, a contradiction.

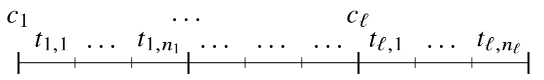
   To see that the other orbit decompositions are in fact possible, consider the following examples (any joint conjugation of the generators will do as well):

   i) $\langle(1\ 2)\rangle$ and $\langle(1\ 2)(3\ 4)\rangle$ has set sizes $(1, 1, 2, 2)$.
   ii) $\langle(1\ 2), (3\ 4)\rangle$ has set sizes $(1, 1, 4)$.
   iii) $\langle(1\ 2\ 3)\rangle$ has set sizes $(3, 3)$.
   iv) $\langle(1\ 2\ 3\ 4)\rangle$ has set sizes $(2, 4)$.
   v) S_4 has set sizes (6). □

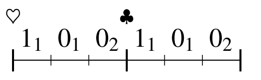
C TURN-SPLIT REPRESENTATION OF (SEMI-)REDUCED STATES

We introduce a turn-split representation of certain turnable states, used in the impossibility proof given in Section 5.

Turn-Split Representation of a (Semi-)Reduced State. For turnable states, we would like to introduce an additional, compressed method to depict the state, where we are only interested in the partitioning of the state and how it behaves relative to the split of the state due to the turn. For this, let us regard empty sequences (sequences not in the state, i.e. with probability 0) as a *sequence of type* $\emptyset$. We sort the state according to a column i (usually a turnable column) and assume that the deck is chosen over an alphabet $c_1, \dots, c_\ell$, which carries some ordering $c_1 \leq \cdots \leq c_\ell$. The ordering inside the blobs for each c_k is irrelevant, and we may only assume an ordering by type. For this, we will use the following *turn-split representation* w.r.t. a position k of a state:


c_1
$\dots$
c_ℓ
$t_{1,1}$ $\dots$ t_{1,n_1} $\dots$ $\dots$ $\dots$ $t_{\ell,1}$ $\dots$ t_{ℓ,n_ℓ}

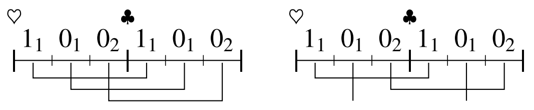
where $t_{1,1}, \dots, t_{1,n_1}, \dots, t_{\ell,1}, \dots, t_{\ell,n_\ell}$ are (indexed) types of corresponding sequences. The indicated ordering means: if the state is turnable at position k, $t_{i,j}$ belongs to the c_i-branch of a turn at position k, for all $1 \leq j \leq n_i$. As an example, let us depict the state from Fig. 2 (left) via this split-state representation w.r.t. the turnable column 2:



We will make use of this representation after we derive a criterion for states that can arise via a closed shuffle.

Enriching Turn-Split Representations with Orbit Partitions. As we just saw, a necessary prerequisite to show that we can reach a state μ' via a uniform closed shuffle, is to show that it is compatible with an orbit partition, i.e. that there is a Π with an orbit partition that is finer than the partition of μ'. Hence, we want to depict a possible orbit partition in a simple way that makes it is obvious that it is finer than the partition of the state.

We do this via connecting horizontal brackets between the sequences representing the orbit decomposition of an admissible shuffle. Here, all connected sequences are meant to be in the same orbit. A single line drawn downward will mean that the sequence is in an orbit of size 1. We use our previous example of Fig. 2 in a turn-split representation w.r.t. the turnable position 2. We depict two possible orbit decompositions (on the left with three sets of size two, on the right with one orbit split into two, which are represented by single lines pointing downwards). It shows that the state is generateable by a shuffle with such an orbit decomposition, as it is not coarser than the partition of the state (specified by the indices).[7] (It would then be left to show that such an orbit decomposition is in fact possible. For example, we will see later in Proposition 1, that the left decomposition is impossible.)



The example abstracts from many details and from the form of Π, but if one can already show (and we do so in Section 5) that there is *no* orbit partition as fine as the state partition, then direct reachability of such a state via a closed uniform shuffle is impossible.

[7]Note that there are also orbit compositions of sizes 1, 1, 4, sizes 3, 3, sizes 2, 4 and size 6, which are however incompatible as they are coarser than the partition of the state and hence are not displayed.