Foundations for Actively Secure Card-based Cryptography

Alexander Koch and Stefan Walzer

Karlsruhe Institute of Technology (KIT), TU Ilmenau,
alexander.koch@kit.edu, stefan.walzer@tu-ilmenau.de

Abstract. Card-based cryptography allows to do secure multiparty computation in simple and elegant ways, using only a deck of playing cards, as first proposed by den Boer (EUROCRYPT 1989). Many protocols as of yet come with an "honest-but-curious" disclaimer. However, a central goal of modern cryptography is to provide security also in the presence of *malicious attackers*. At the few places where authors argue for the active security of their protocols, this is done ad-hoc and restricted to the concrete operations needed, often even using additional physical tools, such as envelopes or sliding cover boxes.
This paper provides the first systematic and comprehensive approach to *active security* in card-based protocols. We show how a large and natural class of *shuffling* operations, namely those which (opaquely) permute the cards according to a *uniform* distribution on a permutation *group*, can be implemented using only a linear number of helping cards. This ensures that any (information-theoretically) secure cryptographic protocol in the model of Mizuki and Shizuya (Int. J. Inf. Secur., 2014), restricted to this natural class of shuffles, can be realized in an actively secure fashion. For this, we develop an alternative computational model for card-based cryptography, which we believe to be of independent interest.

Keywords: Card-based protocols · Card shuffling · Secure multiparty computation · Active security · Cryptography without computers

1 Introduction

The elegant "five-card trick" of den Boer [Boe89] allows two players – here called Alice and Bob – to compute a logical AND of two private bits, using five playing cards. For instance, if the bit of a player encodes whether they have romantic interest for the other player, the protocol will result in a "yes"-output if and only if there is mutual interest, sparing a party with an unrequited crush the embarrassment of having this information revealed.

More generally, using a deck of playing cards (usually with symbols ♡, ♣), Alice and Bob can jointly compute an arbitrary boolean function on multiple secret inputs such that neither player learns anything (new) about the input, except, possibly, what can be learned from looking at the output. One distinctive feature is that these protocols do not need a computer, which makes their security tangible with no danger of malware. This issue is long-known in the context of cryptographic voting which therefore often work with physical objects, such as special ballot papers or receipts, cf. e.g. [PH10; BMR07; MN06b].

Card-based protocols are often elegant and simple and have become popular when introducing secure multiparty computation in lectures and to non-experts.

The feasibility of general secure multiparty computation with cards was shown in [Boe89; NR98; Sti01]. Since then, researchers proposed a wide range of protocols with different objectives and parameters. One line of research has been to minimize the number of cards for use in protocols. In this regard, [MS09; MKS12; KWH15; NNH^{+}15] try to minimize the number of cards for AND, XOR or bit copy protocols, achieving, for instance, the minimum number of four cards for AND protocols both in committed and non-committed format. Moreover, [NHMS15; Miz16; KWH15] are concerned with protocols for general circuits, using the least number of cards. However, some of these protocols "buy" their minimality with very general shuffle operations, where it is not yet clear how they can be performed in the real world, especially in an actively secure manner.

All early protocols relied solely on a uniform *random cut* shuffling to ensure privacy, i.e. a cyclic shift on the cards by a random offset, unknown to the players as they are unable to keep track of quick, repeated cuts. Niemi and Renvall [NR98, Sect. 3] and den Boer [Boe89] plausibly argue that random cuts can be performed privately in this manner. Other shuffle operations were justified, including "dihedral group" shuffles [NR98] and [Sti01, Sect. 7], *random bisection cuts* [MS09; UNH^{+}16] and unequal division shuffles [CHL13; NNH^{+}15; NHMS16].

In the formal computational model of Mizuki and Shizuya [MS14a], a protocol specification may use any of the most general shuffling operations, namely randomly applying a permutation from an *arbitrary permutation set*. This computational model is very useful when showing impossibility results and *lower bounds* on cards, cf. [KWH15], but significantly *overestimates* what can be done with playing cards. There is to this day still no *positive* account of what can be done with playing cards beyond the justification of individual protocols, and even then, most work with "honest-but-curious" assumptions, with no guarantees when one of the players deviates from the protocol.

Related Work. Other works have investigated the question of active attacks, albeit with a different focus. Mizuki and Shizuya [MS14b] address active security against adversaries who deviate from the input encoding, e.g. giving input $(\heartsuit, \heartsuit)$ instead of $(\heartsuit, \clubsuit)$ and markings of cards. We sketch in Appendix F how our results subsume this, using a separate input phase. Moreover, they stress the necessity of non-symmetric backs to avoid marking cards by turning them upside down. Finally, using a secret sharing-like mechanism, they specify how to avoid security breaches by scuff marks on the backs of the cards. Shinagawa et al. [SMS^{+}15] describe a method against injection attacks in their model using polarizing plates.

Besides short ad-hoc discussions of the shuffle security, we believe that this is an exhaustive list of all investigations into active security so far. In particular, the issue of ensuring that only permutations allowed in the protocol description can be performed during a shuffling step has not been addressed for cases where this non-trivial. Very recently and independently, Ueda et al. [UNH$^+$16] give a nice and elaborate implementation of the special case of random bisection cuts, including experiments showing the real-world security of the shuffle.

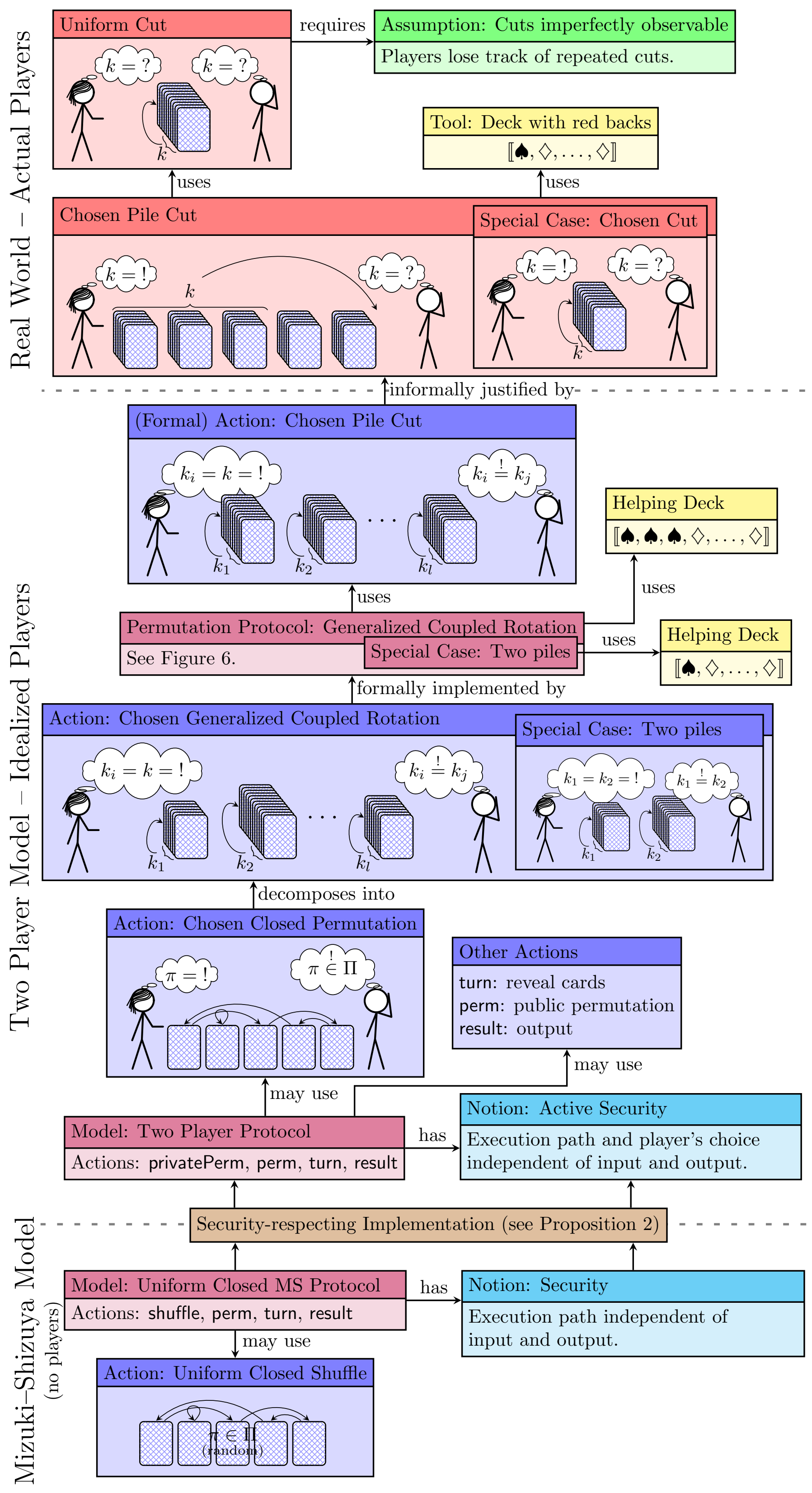
Our Contribution. At several places in the literature, e.g. [CHL13, Sect. 8] and [KWH15, Sect. 9], the open question of achieving actively secure shuffles and protocols is posed. In this paper, we answer a significant part of this question by explaining how any protocol in the model of [MS14a] that is restricted to *uniform closed shuffles* can be transformed into an actively secure protocol using only a linear number of helping cards. Uniform closed shuffles, namely those that rearrange the cards according to a *uniform* distribution on a permutation *group*, have already been identified in [KWH15, Sect. 8] as a natural class of operations. Moreover, we define a new model for card-based cryptography, which we call two-player protocols, where there is a procedure that allows Alice to apply a $\pi \in \mathit{\Pi}$ of her choosing to a sequence of face-down cards, such that Bob learns nothing about her choice. We believe this to be of independent interest, e.g. as an approach to formalize protocols such as the three-card AND protocol by Karun Singh as described in [MWS15, Sect. 3.2] that does not fit into the model of Mizuki and Shizuya.

2 Preliminaries

Permutations. A *permutation* of a set $X = \{1, \ldots, n\}$ for some $n \in \mathbb{N}$, is a bijective map $\pi : X \to X$. The set S_n of all permutations of $\{1, \ldots, n\}$ is called *symmetric group*. It has group structure with the identity map id as neutral element and composition ($\circ$) as group operation. We can apply a permutation π of X to a set $S \subseteq X$ writing $\pi(S) := \{\pi(s) \mid s \in S\}$. We say that π *respects* S if $\pi(S) = S$. In that case, π also respects the complement $X \setminus S$ and we can define the *restriction* of π to S as the permutation τ with domain S and $\tau(s) = \pi(s)$ for all $s \in S$. For elements $x_1, \ldots, x_k$ the *cycle* $(x_1\ x_2\ \ldots\ x_k)$ denotes the *cyclic* permutation π with $\pi(x_i) = x_{i+1}$ for $1 \leq i < k$ and $\pi(x_k) = x_1$ and $\pi(x) = x$ for all x not occurring in the cycle. The domain of π is not apparent from the cycle alone but can be any superset of the numbers occurring in the cycle. If several cycles act on pairwise disjoint sets, we write them next to one another to denote their composition. For instance $(1\ 2)(3\ 4\ 5)$ denotes a permutation with mappings $\{1 \mapsto 2, 2 \mapsto 1, 3 \mapsto 4, 4 \mapsto 5, 5 \mapsto 3\}$. (We omit the composition operator $\circ$.) Every permutation can be written in such a *cycle decomposition*.

By a *conjugate* of a permutation $\pi \in S_n$ we mean any permutation of the form $\pi' := \tau^{-1} \circ \pi \circ \tau$ where $\tau \in S_n$. For a set $\mathit{\Pi} \subseteq S_n$ of permutations and $\tau \in S_n$ the set $\tau^{-1} \circ \mathit{\Pi} \circ \tau := \{\tau^{-1} \circ \pi \circ \tau \mid \pi \in \mathit{\Pi}\}$ is a *conjugate* of $\mathit{\Pi}$. Given an arbitrary sequence of objects $\mathit{\Gamma} = (\mathit{\Gamma}[1], \ldots, \mathit{\Gamma}[n])$ and a permutation $\pi \in S_n$ then *applying* π to $\mathit{\Gamma}$ yields the sequence $\pi(\mathit{\Gamma}) = (\mathit{\Gamma}[\pi^{-1}(1)], \mathit{\Gamma}[\pi^{-1}(2)], \ldots, \mathit{\Gamma}[\pi^{-1}(n)])$. Intuitively, the object in position i is transported to position $\pi(i)$.

Sets and Groups. If $g_1, g_2, \ldots, g_k \in G$ are group elements, $\langle g_1, \ldots, g_k \rangle$ is the smallest subgroup of G containing $g_1, \ldots, g_k$ and called the *subgroup generated*


Real World – Actual Players
Uniform Cut
$k = ?$
$k = ?$
k
requires
Assumption: Cuts imperfectly observable
Players lose track of repeated cuts.
Tool: Deck with red backs
$[\![\spadesuit, \diamondsuit, \ldots, \diamondsuit]\!]$
uses
uses
Chosen Pile Cut
$k = !$
k
$k = ?$
Special Case: Chosen Cut
$k = !$
$k = ?$
k
informally justified by
Two Player Model – Idealized Players
(Formal) Action: Chosen Pile Cut
$k_i = k = !$
$k_i \overset{!}{=} k_j$
k_1 k_2 $\cdots$ k_l
Helping Deck
$[\![\spadesuit, \spadesuit, \spadesuit, \diamondsuit, \ldots, \diamondsuit]\!]$
uses
uses
Permutation Protocol: Generalized Coupled Rotation
See Figure 6.
Special Case: Two piles
uses
Helping Deck
$[\![\spadesuit, \diamondsuit, \ldots, \diamondsuit]\!]$
formally implemented by
Action: Chosen Generalized Coupled Rotation
$k_i = k = !$
$k_i \overset{!}{=} k_j$
k_1 k_2 $\cdots$ k_l
Special Case: Two piles
$k_1 = k_2 = !$
$k_1 \overset{!}{=} k_2$
k_1 k_2
decomposes into
Action: Chosen Closed Permutation
$\pi = !$
$\pi \overset{!}{\in} \Pi$
Other Actions
turn: reveal cards
perm: public permutation
result: output
may use
may use
Model: Two Player Protocol
Actions: privatePerm, perm, turn, result
has
Notion: Active Security
Execution path and player's choice independent of input and output.
Security-respecting Implementation (see Proposition 2)
Mizuki–Shizuya Model
(no players)
Model: Uniform Closed MS Protocol
Actions: shuffle, perm, turn, result
has
Notion: Security
Execution path independent of input and output.
may use
Action: Uniform Closed Shuffle
$\pi \in \Pi$
(random)

A uniform cut rotates a pile of cards by a uniformly random value unknown to Alice and Bob. From this we build chosen cuts leaving a pile rotated by a value chosen by Alice but unknown to Bob. When generalized to chosen pile cuts and formalized we obtain a chosen pile cut action that rotates a sequence of equally-sized piles by a value k chosen by Alice. Bob remains oblivious of that value but he can be sure that the cards are not rearranged in any other way. In particular he knows each pile is rotated by the same amount, even if Alice is dishonest.
With the help of a permutation protocol this is extended to the case where piles may have different sizes. This yields chosen coupled rotations in the case of two piles and chosen generalized coupled rotations in the case of more than two piles.
These are powerful enough to build arbitrary chosen closed permutations. In that setting, Alice may choose any permutation π from a group of permutations Π. Bob will not learn π but can be sure that no permutation outside the set Π is performed.
A two player protocol may make use of these chosen closed permutation actions as well as the other actions turn, perm and result.
Uniform closed Mizuki–Shizuya (MS) protocols are a large natural subset of protocols as formalized by Mizuki and Shizuya. Our main result is that for any such protocol there is a two player protocol computing the same function that is actively secure if the original protocol is secure. This security respecting implementation replaces each uniform closed shuffle with two corresponding chosen closed permutations.
Active security is bought with helping cards needed in several places; intuitively to prove the legitimacy of Alice's actions to Bob.

Fig. 1: Overview of the content of this paper. The images of Alice of Bob are adapted from xkcd (by Randall Munroe), which is licensed as CC-BY-NC-2.5.

by $\{g_1, \ldots, g_k\}$. For $g \in G$ the *order* of g is $\mathsf{ord}(g) = |\langle g \rangle| = \min\{k \geq 1 \mid g^k = \mathsf{id}\}$. In the following, a group is implicitly also the set of its elements.

Multisets and Decks. $[\![\diamondsuit, \diamondsuit, \diamondsuit, \spadesuit, \spadesuit]\!]$ is the multiset containing three copies of $\diamondsuit$ and two copies of $\spadesuit$, also written as $[\![3 \cdot \diamondsuit, 2 \cdot \spadesuit]\!]$. If such a multiset represents cards, it is called a *deck*. All cards are implicitly assumed to have the same back, unless stated otherwise. Cards can lie face-up or face-down. When face-down, all cards are indistinguishable (unless they have different backs). When face-up, cards with the same symbol are indistinguishable. Throughout this paper, cards are always face-down with the exception of special operations that reveal the front of some card(s). To simplify the specification of protocols, such operations immediately turn the card(s) face-down again. Unions of multisets are denoted by $\cup$, disjoint unions are denoted by $+$, e.g. $[\![\diamondsuit, \spadesuit, \spadesuit]\!] \cup [\![\diamondsuit, \heartsuit, \spadesuit, \clubsuit]\!] = [\![\diamondsuit, \heartsuit, \spadesuit, \spadesuit, \clubsuit]\!]$ whereas $[\![\diamondsuit, \spadesuit, \spadesuit]\!] + [\![\diamondsuit, \heartsuit, \spadesuit, \clubsuit]\!] = [\![\diamondsuit, \diamondsuit, \heartsuit, \spadesuit, \spadesuit, \spadesuit, \clubsuit]\!]$.

3 Implementing Cuts and Pile Cuts with Choice

A set $\Pi \subseteq S_n$ of permutations has an *actively secure implementation with choice*, or *is implemented* for short, if there is a procedure that allows Alice to apply a $\pi \in \Pi$ of her choosing to a sequence of face-down cards, such that Bob learns nothing about her choice, but is certain that Alice did not choose $\pi \notin \Pi$. Also, no player learns anything about the face-down cards if the other player is honest.

Example: Bisection Cut with Envelopes. Mizuki and Sone [MS09] make use of the following procedure on six cards: The cards in positions 1, 2 and 3 are stacked and put in one envelope and the cards in position 4, 5 and 6 are put into another. Behind her back, Alice then swaps the envelopes or leaves them as they are – her choice. Unpacking yields either the original sequence or the sequence $4, 5, 6, 1, 2, 3$. The bisection cut $\Pi = \{\mathsf{id}, (1\ 4)(2\ 5)(3\ 6)\}$ is therefore implemented (with active security and choice) using two indistinguishable envelopes. For a model of secure envelopes, cf. [MN06a; MN10].

Example: Unequal division shuffle. A bisection cut on n cards can be interpreted as "either do nothing or rotate the sequence by $n/2$ positions". Generalizing this, we now want to "either do nothing or rotate the sequence by l positions" for some $0 < l < n$ i.e. implement $\Pi_l = \{\mathsf{id}, (1\ 2\ \ldots\ n)^l\}$. Nishimura et al. [NNH+15] describe a corresponding mechanism using two card cases with sliding covers. The card cases behave like envelopes but are heavy enough to mask inequalities in weight caused by different numbers of cards, and support joining the content of two card cases – for details refer to their paper (or Appendix E).

While we are very fond of creative ideas such as these, we shall make it our mission to implement card based protocols using only one tool: additional cards.

3.1 Cutting the Cards

By the *cut on n cards* we mean the permutation set $\Pi = \langle (1\ \ldots\ n) \rangle$ and, ordinarily, Alice would *cut* a pile of n cards by taking the top-most k cards (for some $0 \leq k < n$) from the top of the pile and setting them aside and then placing the remaining $n-k$ cards on top. In this form, Alice can *only approximately* pick k while allowing Bob to approximately observe k, two things we have to fix.

Uniform Cut. As an intermediate goal we implement a *uniform cut* on n cards, i.e. we perform a permutation $(1\ 2\ \dots\ n)^k$ for $0 \leq k < n$ chosen uniformly at random and unknown to the players. As proposed in [Boe89], this is done by repeatedly cutting the pile in quick succession until both players lost track of what happened. More formally, under reasonable assumptions, the players observe a Markov chain that converges quickly to the uniform stable distribution, yielding an almost uniform distribution after a finite number of steps.

Arguably, if the pile is too small, say two cards, the number of cards taken during each cut is perfectly observable. In that case, we put a sufficiently large number c of cards with different backs behind each card, repeatedly cut this larger pile and remove the auxiliary cards afterwards. Note that [UNH+16] found it to work well in practice even for $n = 2$ and $c = 3$.[1]

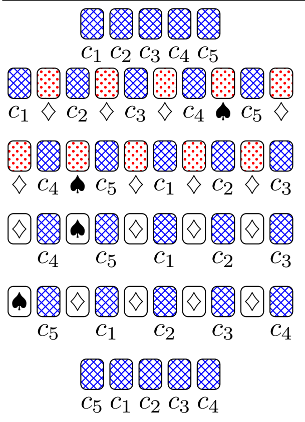
Uniform Cut with alternating backs. Later we apply the uniform cut procedure to piles of $n \cdot (\ell + 1)$ cards with n cards of red back, each preceded by ℓ cards of blue back. From a "uniform cut" on such a pile, we expect a cut by $0 \leq k < n \cdot (\ell + 1)$ where $\lfloor k/(\ell + 1) \rfloor$ is uniformly distributed in $\{0, ..., n-1\}$ and independent of the observable part $k \pmod{\ell + 1}$. We leave it to the reader to verify that the iterated cuts still work under the same assumptions.

Chosen Cut. We now show that $\Pi = \langle (1\ \dots\ n) \rangle$ is implemented with active security and choice. Say Alice wants to rotate the pile of n cards by exactly k positions for a secret $0 \leq k < n$. We propose the process illustrated in Fig. 2.

Alice is handed the helping deck $[\![\spadesuit, (n-1) \cdot \heartsuit]\!]$ with red backs and secretly rearranges these cards in her hand (unobserved by Bob), putting $\spadesuit$ in position k. The helping cards are put face-down on the table and interleaved with the pile to be cut (each blue card followed by a red card). The $\spadesuit$ is now to the right of the card that was the k-th card in the beginning. To obscure Alice's choice of k, we perform a uniform cut on all cards as described previously. The red helping cards are then turned over. Rotating the sequence so as to put $\spadesuit$ in front, and removing the helping cards afterward leaves the cards in the desired configuration. Bob is clueless about k since he can merely observe the position of $\spadesuit$ *after* the cut, which is independent of the position of $\spadesuit$ before the cut (which encodes k).

Chosen Pile Cut. Chosen cuts can be generalized in an interesting way. Given n piles of ℓ cards each and $0 \leq k < n$, Alice wants to rotate the sequence of piles by exactly k positions, meaning the i-th pile will end up where pile $i + k$ has been (modulo n). Again, k must remain hidden from Bob and he, on the other hand, wants to be certain that Alice does not tamper with the piles in any other

[1] If not satisfied, the reader may be more inclined to accept some variant of Berry's turntable as described by Verhoeff [Ver14, Sect. 3]. There, cards are attached to a wheel-of-fortune-esque device.

Example $(n = 5, k = 4)$	**General Description**
	Alice inserts helping cards, puts ♠ right of c_k. A uniform cut is performed. The helping cards are revealed. The ♠ is rotated to the front. The helping cards are discarded.

c_1 c_2 c_3 c_4 c_5
c_1 ♢ c_2 ♢ c_3 ♢ c_4 ♠ c_5 ♢
♢ c_4 ♠ c_5 ♢ c_1 ♢ c_2 ♢ c_3
c_4 c_5 c_1 c_2 c_3
c_5 c_1 c_2 c_3 c_4
c_5 c_1 c_2 c_3 c_4

Fig. 2: Alice cuts a pile of n cards, here $[\![c_1, \ldots, c_5]\!]$, with back ▨ at position k with a helping deck of n helping cards $[\![♠, 4 \cdot ♢]\!]$ with back ▨. In this illustration we annotated face-down cards with the symbol they contain.

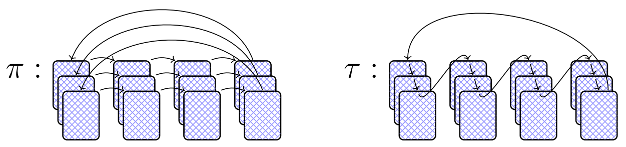


Fig. 3: Rotating a sequence of four piles of three cards each by one position *(left)* is described by a permutation π with three cycles of length 4. Alternatively, we can think of π as $\pi = \tau^3$ where τ is the cyclic permutation of length 12 *(right)*.

than the stated way. Note that this is equivalent to cutting a pile of $n\ell$ cards where only cutting by multiples of ℓ is allowed, see Fig. 3. In that interpretation, the i-th pile is made up of the cards in positions $(i-1)\ell + 1, \ldots, i\ell$.

We apply the same procedure as before with n helping cards, except this time, instead of a single blue card we have ℓ blue cards (a pile) before each of the n gaps that Alice may fill with her red deck $[\![♠, (n-1) \cdot ♢]\!]$. Now the special ♠-card marks the end of the k-th pile and is (after a uniform cut) rotated to the beginning of the sequence, ensuring that after removing the helping cards again we end up having rotated the $n \cdot \ell$ cards by a multiple of ℓ as desired. Note that, uniform (non-chosen) pile cuts have been proposed in [ICM15] as “pile-scramble shuffles”, with and implementation using rubber bands, clips or envelopes.

Summary. If $\Pi = \langle (1\ 2\ \ldots\ n \cdot \ell)^\ell \rangle$ for $n, \ell \in \mathbb{N}$, then Π is implemented with active security and choice using the helping deck $[\![♠, (n-1) \cdot ♢]\!]$. For $\ell = 1$ it is called a *cut*, for $\ell > 1$ a *pile cut*. We use the same name for conjugates of Π, i.e. if cards are relabeled. Any subset $\emptyset \neq \Pi' \subset \Pi$ of a (pile) cut is also implemented: Let Alice place ♠ only in some positions, the others are publicly filled with ♢. The adequacy of our implementation of (pile) cuts is ultimately a plausible-but-informal claim about what *real people* can and cannot physically achieve. However, in all that follows we proceed with formal rigor.

4 Permutation Protocols for Arbitrary Groups

We introduce a formal concept that allows to compose simple procedures in order to implement more complicated permutation sets.

Definition 1. *A* permutation protocol $\mathcal{P} = (n, \mathcal{H}, \Gamma, A)$ *is given by a number* n *of* object cards*, a deck of helping cards* $\mathcal{H}$ *with initial arrangement* $\Gamma\colon \{n+1, \dots, n+|\mathcal{H}|\} \to \mathcal{H}$*, and a sequence* A *of* actions *where each action can be either*

– $(\mathsf{privatePerm}, \Pi)$ *for* $\Pi \subseteq S_{n+|\mathcal{H}|}$ *implemented with active security and choice, and respecting* $\{1, \dots, n\}$ *(i.e.* $\forall \pi \in \Pi\colon \pi(\{1, \dots, n\}) = \{1, \dots, n\}$*), or*
– (check, p, s) *for a position* p *of a helping card (i.e.* $n < p \leq n + |\mathcal{H}|$*) and a supposed outcome* $s \in \mathcal{H}$*.*

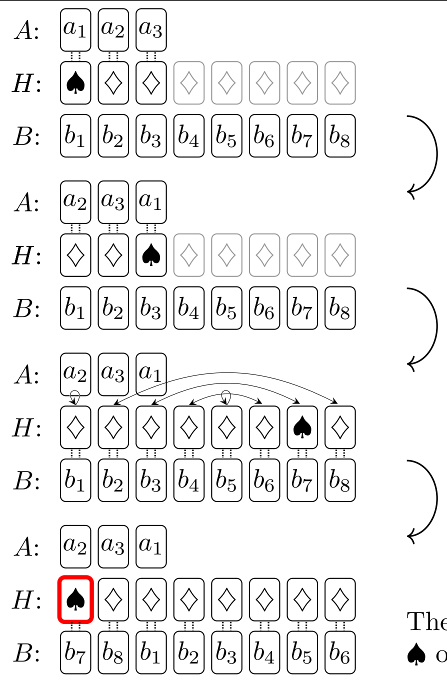
We are interested in the set $\mathsf{comp}(\mathcal{P}) \subseteq S_{n+|H|}$ of permutations *compatible* with $\mathcal{P}$. If there are k $\mathsf{privatePerm}$ actions with permutations sets $\Pi_1, \dots, \Pi_k$ and $\pi_i \in \Pi_i$, then $\pi_k \circ \dots \circ \pi_1$ is compatible with $\mathcal{P}$ if each check *succeeds*, meaning if (check, p, s) happens after the i-th $\mathsf{privatePerm}$ action (and before the $i+1$st, if $i < k$) then $\Gamma[(\pi_i \circ \dots \circ \pi_1)^{-1}(p)] = s$. We argue that this implements $\Pi' = \mathsf{comp}(\mathcal{P})|_{\{1,\dots,n\}}$ using $\mathcal{H}$ (and, possibly, helping cards to implement Π_i).

Indeed, consider the following procedure: We start with n object cards lying on a table (positions $1, \dots, n$). We place the sequence Γ next to it, at positions $n+1, \dots, n+|\mathcal{H}|$, and go through the actions of $\mathcal{P}$. Whenever the action $(\mathsf{privatePerm}, \Pi_i)$ is encountered, we use the procedure $\mathcal{P}_i$ implementing Π_i to let Alice apply a permutation on the current sequence. When an action (check, p, s) is encountered, the p-th card is revealed. If its symbol is s, Bob continues, otherwise he aborts, declaring Alice as dishonest. In the end, the helping cards are removed, yielding a permuted sequence of object cards. (All permutations respect $\{1, \dots, n\}$, hence, the helping and the object cards remain separated).

Alice can freely pick any $\pi' \in \Pi'$; using an appropriate decomposition, all checks will succeed. In this case, Bob knows that the performed permutation is from Π'. No player learns anything about the object cards (they remain face-down) and conditioned on Alice being honest, the outcome of the checks is determined, so Bob learns nothing about π'.

Coupled Rotations. Let $\varphi = (1\ 2\ \dots\ s)$ and $\psi = (s{+}1\ s{+}2\ \dots\ s{+}t)$, assume $s < t$. For $\pi = \psi \circ \varphi = \varphi \circ \psi$ we call $\Pi = \{\pi^k \mid 0 \leq k < s\}$ the *coupled rotation* with parameters s and t. Note that Π is not a group since $\pi^s \notin \Pi$. We aim to implement Π. We make use of a helping deck $[\![♠, (t{-}1) \cdot ♢]\!]$ available in positions $H = \{h_0, h_1, \dots, h_{t-1}\}$ with ♠ at position h_0. Then define $\hat{\varphi} := \varphi \circ (h_0\ \dots\ h_{s-1})$ and $\hat{\psi} := \psi \circ (h_0\ \dots\ h_{t-1})^{-1}$ and consider the permutation protocol $\mathcal{P}$ in Fig. 5(a).

Example $(s = 3, t = 8, k = k' = 2)$ **General Description**



The sequences A and H (first s cards) are rotated to the right by the same value $k \in \{0, 1, \dots, s-1\}$ chosen by Alice.
(This is a pile cut.)

H is rearranged so as to represent $-k \pmod t$.

H and B are rotated to the right by $k' \in \{0, 1, \dots, t-1\}$ chosen by Alice. If Alice is *honest* she must choose $k = k'$.
(This is a pile cut.)

The first card of H is revealed. A ♠ occurs iff Alice was honest.

Fig. 4: The sequence A of length s and B of length t are to be rotated by the same value k chosen privately by Alice. A helping sequence ensures that the same value is used. All cards are face-down, except for the highlighted card in the last step. The dotted lines indicate that cards are belonging to the same pile in a pile cut, i.e. they maintain their relative position during the cut. The rearrangement of the helping cards is useful in this visualization (so that H and B can be rotated in the same direction) but is not reflected in the formal description.

The idea here is that Alice may choose k and k' and perform $\hat{\varphi}^k$ and $\hat{\psi}^{k'}$ to the sequence. However, k is "recorded" in the configuration of a helping sequence and $-k'$ is "added" on top. A check ensures that the helping sequence is in its original configuration, implying $k = k'$ as required. Note that $\langle\hat{\varphi}\rangle$ and $\langle\hat{\psi}\rangle$ are pile cuts, which we already know how to implement. In total, we implemented

$$\begin{aligned}\mathsf{comp}(\mathcal{P}) &= \{\hat{\psi}^{k'} \circ \hat{\varphi}^k \mid 0 \le k < s, 0 \le k' < t, \Gamma[(\hat{\psi}^{k'} \circ \hat{\varphi}^k)^{-1}(h_0)] = \spadesuit\}|_{\{1,\dots,n\}} \\ &= \{\hat{\psi}^{k'} \circ \hat{\varphi}^k \mid 0 \le k < s, 0 \le k' < t, k' = k\}|_{\{1,\dots,n\}} \\ &= \{\psi^k \circ \varphi^k \mid 0 \le k < s\}|_{\{1,\dots,n\}} = \Pi.\end{aligned}$$

Products, conjugates and syntactic sugar. The protocol in Fig. 5(b) implements $\Pi_2 \circ \Pi_1$ using Π_1 and Π_2, showing that if Π_1 is implemented us-

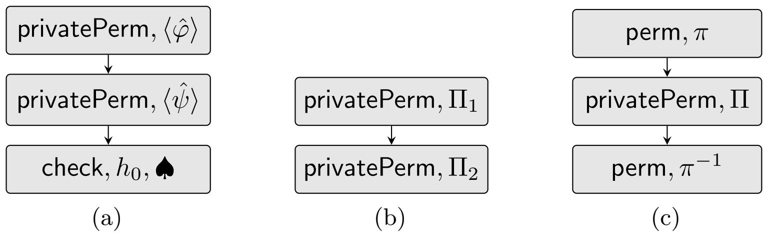


Fig. 5: Protocols implementing a coupled rotation (a), the product of two permutation sets (b) and the conjugation of a permutation set (c).

ing $\mathcal{H}_1$ and Π_2 is implemented using $\mathcal{H}_2$, then $\Pi_2 \circ \Pi_1$ is implemented using $\mathcal{H}_1 \cup \mathcal{H}_2$. As a corollary, if Π is implemented using $\mathcal{H}$ then so is any conjugate $\Pi' = \{\pi^{-1}\} \circ \Pi \circ \{\pi\}$. Figure 5(c) uses (perm, π) instead of $(\mathsf{privatePerm}, \{\pi\})$ to emphasize that such deterministic actions can be carried out publicly.

Generalized Coupled Rotations. We generalize the idea of a coupled rotation to more than two sequences. Let $\pi \in S_n$ with cycle decomposition $\pi = \varphi_0 \circ \ldots \varphi_\ell$ for $\ell \geq 2$ and increasingly ordered cycle lengths $t_0 \leq t_1 \leq t_2 \leq \ldots \leq t_\ell$. We aim to implement $\Pi = \{\pi^k \mid 0 \leq k < t_0\}$ using $t_\ell + 2 \cdot t_0$ helping cards, originally available in the following positions which we label as shown.

$$\underbrace{\overset{\spadesuit}{m_0}\ \overset{\diamondsuit}{m_1}\ \overset{\cdots}{\ldots}\ \overset{\diamondsuit}{m_{t_k-1}}}_{\text{main}}\ \underbrace{\overset{\spadesuit}{x_0}\ \overset{\diamondsuit}{x_1}\ \overset{\cdots}{\ldots}\ \overset{\diamondsuit}{x_{t_0-1}}}_{\text{temp}}\ \underbrace{\overset{\spadesuit}{s_0}\ \overset{\diamondsuit}{s_1}\ \overset{\cdots}{\ldots}\ \overset{\diamondsuit}{s_{t_0-1}}}_{\text{store}}$$

We think of the three areas as "registers" *containing* values indicated by the position of ♠ (initially 0). The registers have associated rotations:

$$\psi_{\mathsf{temp}} := (x_0\ \ldots\ x_{t_0-1}), \qquad \psi_{\mathsf{store}} := (s_0\ \ldots\ s_{t_0-1}), \qquad \psi_i := (m_0\ \ldots\ m_{t_i-1}).$$

The idea behind the protocol is that Alice performs $\varphi_0^{k_0} \circ \ldots \circ \varphi_\ell^{k_\ell}$ where checks will ensure $k_1 = k_2 = \ldots = k_\ell$. To this end, k_0 is recorded in the store register (we use $\langle \varphi_0 \circ \psi_{\mathsf{store}} \rangle$). Then, for each round $i \in \{1, 2 \ldots, \ell - 1\}$ the value k_0 is cloned into the main register by first swapping it to the temp register and then moving it to the store *and* main register using $\psi_{\mathsf{copy}} := \psi_{\mathsf{temp}}^{-1} \circ \psi_{\mathsf{store}} \circ \psi_0$. The cloned copy of k_0 in main is consumed when forcing Alice to do $\widehat{\varphi}_i^{\,k_0}$ where $\widehat{\varphi}_i := \varphi_i \circ \psi_i^{-1}$. The last round is similar. Using the following two swappings, the protocol is formally given in Fig. 6.

$$\pi_{\mathsf{store}\leftrightarrow\mathsf{tmp}} := (s_0\ x_0) \ldots (s_{t_0-1}\ x_{t_0-1}), \pi_{\mathsf{store}\leftrightarrow\mathsf{main}} := (s_0\ m_0) \ldots (s_{t_0-1}\ m_{t_0-1}).$$

It is easy to check that this implements the generalized coupled rotation Π using the helping cards $[\![3 \cdot \spadesuit, (t_\ell{+}2t_0{-}3) \cdot \diamondsuit]\!]$, see Appendix A. The main ingredient of a formal proof is the loop invariant:

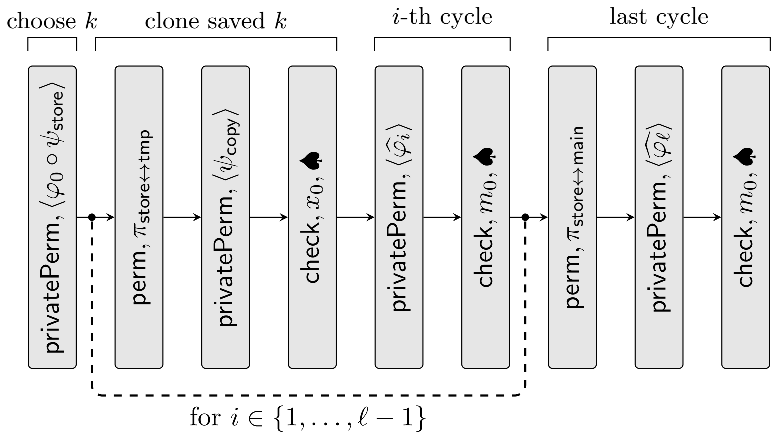

choose k
clone saved k
i-th cycle
last cycle
privatePerm, $\langle \varphi_0 \circ \psi_{\mathsf{store}} \rangle$
perm, $\pi_{\mathsf{store} \leftrightarrow \mathsf{tmp}}$
privatePerm, $\langle \psi_{\mathsf{copy}} \rangle$
check, x_0, ♠
privatePerm, $\langle \widehat{\varphi_i} \rangle$
check, m_0, ♠
perm, $\pi_{\mathsf{store} \leftrightarrow \mathsf{main}}$
privatePerm, $\langle \widehat{\varphi_\ell} \rangle$
check, m_0, ♠
for $i \in \{1, \dots, \ell - 1\}$

Fig. 6: Protocol to implement a generalized coupled rotation with $\ell + 1$ cycles of length $t_0, t_1, \dots, t_\ell$. Notation is explained in the text.

If π is a permutation compatible with the actions until after the i-th execution of the loop and S is the starting sequence then:

- $\pi|_{\{1,\dots,n\}} = \varphi_i^k \circ \dots \circ \varphi_1^k \circ \varphi_0^k$ *for some* $k \in \{0, \dots, t_0 - 1\}$,
- *in* $\pi(S)$ *all registers contain* 0 *except for store, which contains* k.

We remark that by introducing additional check-operations, any subset of a generalized coupled rotation can be implemented as well.

Subgroups of S_n. Generalized coupled rotations are sufficient to show:

Proposition 1. *Any subgroup Π of S_n can be implemented with active security and choice using only the helping deck $[\![3 \cdot ♠, (n-3) \cdot ♢]\!]$ for (generalized) coupled rotations, the helping deck $[\![♠, (n-1) \cdot ♢]\!]$ for (pile) cuts and whatever is needed for uniform cuts (see page 6).*

Proof. Note that $\Pi = \prod_{\pi \in \Pi} \langle \pi \rangle$, i.e. Π can be written as the product of cyclic subgroups. Moreover, any cyclic subgroup can be written as $\langle \pi \rangle = \{\pi^0, \dots, \pi^{k-1}\}^\ell$, where k is the length of the shortest cycle in the cycle decomposition of π and $\ell = \mathsf{ord}(\pi)/k$. Hence, Π can be written as the product of rotations and (generalized) coupled rotations, each of which are implemented with the required helping decks. Using the implementation of products (see page 9), we are done. □

A *simple* decomposition of Π into products of previously implemented permutation sets is desirable in terms of running time. We do not deal with this here and merely state that $|\Pi|$ is an upper bound on the number of terms required.

5 Computational Model with Two Players

In the following, two players jointly manipulate a sequence of cards to compute a randomized function, i.e. they transform an input sequence into an output sequence. Both have incomplete information about the execution and the goal is to compute with no player learning anything about input or output[2].

Two player protocols. A *two player protocol* is a tuple $(\mathcal{D}, U, Q, A)$ where $\mathcal{D}$ is a deck, U is a set of input sequences, Q is a (possibly infinite, computable) rooted tree with labels on some edges, and $A\colon V(Q) \to \mathsf{Action}$ is an action function that assigns to each vertex an action which can be perm, turn, result, and privatePerm, with parameters as explained below. All input sequences have the same length n and are formed by cards from $\mathcal{D}$. Vertices with a perm or privatePerm action have exactly one child, vertices with a result action have no children, and those with a turn action have one child for each possible sequence of symbols the turned cards might conceal, and the edge to that child is annotated with that sequence.

When a protocol is *executed on an input sequence* $I \in U$, we start with the face-down sequence $\Gamma = I$ at the root of Q and empty *permutation traces* $\mathcal{T}_1$ and $\mathcal{T}_2$. Execution proceeds along a descending path in Q and for each vertex v that is encountered, the action $A(v)$ is executed on the current sequence of cards:

(perm, π) for a permutation $\pi \in S_n$. This replaces the current sequence Γ by the permuted sequence $\pi(\Gamma)$. Execution proceeds at the unique child of v.

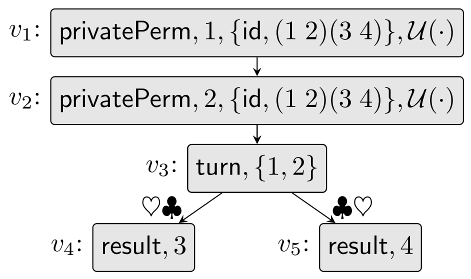
(turn, T) for some set $T \subseteq \{1, \ldots, n\}$. If $T = \{t_1 < t_2 < \ldots < t_k\}$, the cards $\Gamma[t_1], \ldots, \Gamma[t_k]$ are turned face-up, revealing their symbols. The vertex v must have an outgoing edge labeled $(\Gamma[t_1], \ldots, \Gamma[t_k])$. Execution proceeds at the corresponding child after the cards are all turned face-down again.

$(\mathsf{privatePerm}, p, \Pi, \mathcal{F}(\cdot))$ for a player $p \in \{1, 2\}$, a permutation set $\Pi \subseteq S_n$ and $\mathcal{F}$ being a parameterized distribution on Π. Formally, $\mathcal{F}$ is a function that maps the current permutation trace $\mathcal{T}_p$ of player p to a distribution $\mathcal{F}(\mathcal{T}_p)$ on Π. If $\mathcal{F}(\mathcal{T}_p)$ is the uniform distribution on Π for each $\mathcal{T}_p$ we denote this as $\mathcal{U}(\cdot)$. Player p picks a permutation $\pi \in \Pi$. The current sequence Γ is replaced by the permuted sequence $\pi(\Gamma)$ and π is appended to the player's permutation trace $\mathcal{T}_p$. If player p is *honest* she picks π according to $\mathcal{F}(\mathcal{T}_p)$. Execution proceeds at the unique child of v.

$(\mathsf{result}, p_1, \ldots, p_k)$ for distinct positions $p_1, \ldots, p_k \in \{1, \ldots, n\}$. Execution terminates with the *output* $O = (\Gamma[p_1], \ldots, \Gamma[p_k])$.

The execution yields an *execution trace* $(I, O, \mathcal{T}_1, \mathcal{T}_2, W)$, containing the input, output, permutation traces of the players and the path W in Q that was taken, see Fig. 7 for an example. The output of non-terminating protocols is $O = \bot$. We

[2] Why protect the output? We give two answers: *Firstly*, it allows us to offer an "oblivious computation service": A client can walk into the room with a tray of face-down cards encoding an input. Alice and Bob perform the protocol, producing a sequence of face-down cards with which the client can walk out of the room again. *Secondly,* even for the more natural case where Alice and Bob compute a result for themselves, they may require sub-protocols that hide both input and output.



v_1: privatePerm, 1, {id, (1 2)(3 4)}, $\mathcal{U}(\cdot)$
v_2: privatePerm, 2, {id, (1 2)(3 4)}, $\mathcal{U}(\cdot)$
v_3: turn, {1, 2}
♡♣
♣♡
v_4: result, 3
v_5: result, 4

$$\mathcal{D} = [\![3 \cdot ♣, 2 \cdot ♡]\!],$$

$$\begin{aligned} U = \{&(♣, ♡, ♣, ♣),\\ &(♣, ♡, ♡, ♣),\\ &(♡, ♣, ♣, ♣),\\ &(♡, ♣, ♡, ♣)\},\end{aligned}$$

Q, A: as shown on the left

Fig. 7: A protocol example in the two player model, with possible execution trace: $(I = (♡, ♣, ♡, ♣), O = (♡), \mathcal{T}_1 = (\mathsf{id}), \mathcal{T}_2 = ((1\ 2)(3\ 4)), W = (v_1, v_2, v_3, v_5))$. This is an actively secure implementation of the AND protocol in [Miz16, Sect. 3.2]. The first two cards encode an input a as $(♣, ♡) \hat{=} 0$, $(♡, ♣) \hat{=} 1$, the third card encodes an input b as $♣ \hat{=} 0$, $♡ \hat{=} 1$. This encoding is also used for output $a \wedge b$.

say $\mathcal{P}$ is *implemented* using a helping deck $\mathcal{H}$ if each permutation set occurring in a privatePerm action is implemented using $\mathcal{H}$. Security is discussed next. Note that the way we defined it, existence, security and implementability of a protocol are separate issues, but, arguably, little is achieved unless one has all three.

6 Passive and Active Security

The following definition captures security for honest but curious players.

Definition 2 (Passive Security). *A two player protocol $\mathcal{P} = (\mathcal{D}, U, Q, A)$ is* secure against passive attackers *if for any random variable $I \in U$ the following holds. If $(I, O, \mathcal{T}_1, \mathcal{T}_2, W)$ is the execution trace when executing $\mathcal{P}$ with honest players on input I, then (I, O) is independent of $(\mathcal{T}_p, W)$ for both $p \in \{1, 2\}$.*

This definition captures that a player p cannot – from what she chooses to do $(\mathcal{T}_p)$ and ends up observing (W) – learn anything on sensitive data (I, O).

When a $(\mathsf{privatePerm}, p, \Pi, \mathcal{F}(\cdot))$ is encountered and player p is an *active attacker* with trace $\mathcal{T}_p$, she may disrespect the distribution $\mathcal{F}(\mathcal{T}_p)$. This should not allow her to learn anything about (I, O), provided the other player is honest.

Definition 3. *Let $\mathcal{P} = (\mathcal{D}, U, Q, A)$ be a two player protocol.*

(i) *An* active attack ξ on $\mathcal{P}$ as player $p \in \{1, 2\}$ *specifies for each vertex $v \in V(Q)$ with an action of the form $A(v) = (\mathsf{privatePerm}, p, \Pi, \mathcal{F}(\cdot))$, a permutation $\xi(v) \in \Pi$. Replacing such $\mathcal{F}(\cdot)$ with the (point) distributions that always choose $\xi(v)$, yields the* attacked protocol $\mathcal{P}^\xi$.

(ii) *An attack ξ is* unsuccessful *if the following holds. Whenever $I \in U$ is a random variable denoting an input and $(I, O, \mathcal{T}_1, \mathcal{T}_2, W)$ and $(I, O^\xi, \mathcal{T}_1^\xi, \mathcal{T}_2^\xi, W^\xi)$ are the resulting execution traces of $\mathcal{P}$ and $\mathcal{P}^\xi$, then for any values i, o, w:*

$$\Pr[W^\xi{=}w] > 0 \Longrightarrow \Pr[(I, O^\xi){=}(i, o) \mid W^\xi{=}w] = \Pr[(I, O){=}(i, o)]. \qquad (\star)$$

(iii) We say $\mathcal{P}$ is secure against active attacks *(has active security) if each active attack on $\mathcal{P}$ is unsuccessful.*

Intuitively, a protocol has active security if: No matter what permutations a player chooses ($\forall \xi$), and no matter what the turn actions end up revealing ($\forall W^{\xi}$), the best guess for the in- and output (distribution of (I, O^{ξ}), given W^{ξ}) is no different from what he would have said, had he not been involved in the computation at all (distribution of (I, O)). We make a few remarks.

- Passively secure protocols terminate almost surely, otherwise $O = \bot$ can be recognised from an infinite path W. For similar reasons, an active attacker can never cause an actively secure protocol to run forever.[3]
- In our definition, active attackers are deterministic without loss of generality. Intuitively, if an attacker learns nothing *no matter what she does*, then choosing what she does *randomly* is just a fancy way of determining in what way she is going to learn nothing.
- For similar reasons, active security implies passive security, since playing honestly is just a weighted mixture of "pure" attacks.
- We cannot say anything if *both* players are dishonest or if they share their execution traces with one another. We also cannot guarantee that player p learns nothing if player $2 - p$ is dishonest.

Active security from passive security. There is an important special case in which the powers of an active attacker turn out to be blunt:

Proposition 2. *Let $\mathcal{P} = (\mathcal{D}, U, Q, A)$ be a passively secure protocol where for each action of the form $(\mathsf{privatePerm}, p, \Pi, \mathcal{F}(\cdot))$ and each permutation trace $\mathcal{T}_p$ of player p, $\mathcal{F}(\mathcal{T}_p)$ has support Π[4]. If for each attack ξ the attacked protocol $\mathcal{P}^{\xi}$ terminates with probability 1[5], then $\mathcal{P}$ has active security.*

In $\mathcal{P}$ an active attacker can only choose permutations that an honest player may have chosen randomly, so non-trivial information he obtains about in- and output is also available to a passive attacker with positive probability. The protocol from Fig. 7 has active security precisely for this reason. We give a proof in Appendix D.

7 Implementing Restricted Mizuki–Shizuya Protocols

In [MS14b], Mizuki and Shizuya's self-proclaimed goal was to define a "computational model which captures what can possibly be done with playing cards". Hence, any secure real-world procedure to compute something with playing cards can be formalized as a secure protocol in their model. The other direction is not so clear. Given a secure protocol in the model, can it be implemented in the real world? We believe the answer is probably "no" (or, at least, not clearly "yes").

[3] Protocols that almost surely output $\bot$ are a pathological exception.
[4] Otherwise, active attackers may pick $\pi \in \Pi$ which honest players never choose.
[5] this excludes a pathological case

However, our work of identifying implementable actions in the two player model implies that a very natural subset of actions in Mizuki and Shizuya's model is implementable, even in an actively secure fashion: Uniform closed shuffles.

Mizuki–Shizuya Protocols. We will state a modified version of Mizuki and Shizuya's model, in that instead of state machine semantics we stick to a tree of actions as in the two player model. We also make a few other simplifications which are inconsequential for this discussion, cf. Appendix G.

A *Mizuki–Shizuya protocol* is a tuple $\mathcal{P} = (\mathcal{D}, U, Q, A)$ similar to a two player protocol. The actions perm, result and turn are available as before, but instead of $\mathsf{privatePerm}$ actions there are $\mathsf{shuffle}$ actions of the form $(\mathsf{shuffle}, \Pi, \mathcal{F})$ where Π is a set of permutations and $\mathcal{F}$ is a probability distribution on Π. Executing a protocol works as before, but there are no separate permutation traces for players (there are no players at all), instead there is a single permutation trace $\mathcal{T}$. The actions perm, turn and result work as before. When an operation $(\mathsf{shuffle}, \Pi, \mathcal{F})$ is encountered, a permutation $\pi \in \Pi$ is chosen according to $\mathcal{F}$ (independent from any previous choices). This permutation π is applied to the current sequence of cards without anyone learning π and appended to the permutation trace $\mathcal{T}$.

For any input $I \in U$, an execution is described by the execution trace $(I, O, \mathcal{T}, W)$ where O is the output ($O = \perp$ if the protocol did not terminate), $\mathcal{T}$ the resulting permutation trace and W the path of the execution in Q. It is assumed that only W can be observed, suggesting the following security notion:

Definition 4 (Security of Mizuki–Shizuya Protocols). *A Mizuki–Shizuya protocol $\mathcal{P}$ is* secure *if for each random variable $I \in U$ and resulting execution trace $(I, O, \mathcal{T}, W)$ of the protocol, (I, O) is independent from W.*

Implementing Uniform Closed Mizuki–Shizuya Protocols. A shuffle $(\mathsf{shuffle}, \Pi, \mathcal{F})$ is *uniform* if $\mathcal{F}$ is the uniform distribution on Π, and it is *closed* if Π is a group. We call a Mizuki–Shizuya protocol uniform closed if each of its shuffle actions is uniform and closed. We are ready to state our main theorem.

Main Theorem. *Let $\mathcal{P} = (\mathcal{D}, U, Q, A)$ be a secure uniform closed Mizuki–Shizuya protocol. Then there is an actively secure two player protocol $\hat{\mathcal{P}} = (\mathcal{D}, U, \hat{Q}, \hat{A})$ computing the same function as $\mathcal{P}$.*

Moreover, $\hat{\mathcal{P}}$ is implemented using as helping deck only $[\![3 \cdot \spadesuit, (n-3) \cdot \heartsuit]\!]$ for (generalized) coupled rotations, $[\![\spadesuit, (n-1) \cdot \heartsuit]\!]$ for chosen (pile) cuts and cards needed for uniform cuts (see page 6). Here, n is the length of the input sequences.

We sketch the proof here and give details in Appendix D. Each uniform closed shuffle $(\mathsf{shuffle}, \Pi, \mathcal{U})$ of $\mathcal{P}$ is replaced by two actions $(\mathsf{privatePerm}, p, \Pi, \mathcal{U})$ for $p \in \{1, 2\}$. For $\pi_2 \circ \pi_1$ to be uniformly random in Π, it suffices if π_1 *or* π_2 is chosen uniformly random in Π (while the other is fixed/known). Therefore, the joint permutation applied to the sequence after both $\mathsf{privatePerm}$ actions looks uniformly random to both players. Hence, they learn nothing from the execution of $\hat{\mathcal{P}}$ that they would not have also learned from executing $\mathcal{P}$. Since $\mathcal{P}$ is secure, $\hat{\mathcal{P}}$ is passively secure and by Proposition 2 also actively secure. Moreover, by Proposition 1 all Π are implemented using the stated decks of helping cards.

8 Conclusion

Central to our notion of active security is the concept of a *permutation set implemented with active security and choice*, indicating that a player Alice can choose to perform a permutation from the set while Bob can know that Alice did not cheat, but nothing else. We argued that *cuts* and *pile cuts* have this property and used *permutation protocols* to build more sophisticated procedures handling any group of permutations. Moreover, we defined security for Mizuki–Shizuya protocols, active and passive security for our own *two player protocols* and showed how secure Mizuki–Shizuya protocols using only uniform closed shuffles can be transformed into actively secure two player protocols. This is a solid foundation for actively secure card-based cryptography.

Open Problems. The current card-minimal committed format protocols, namely the four- and five-card AND protocols and the general k-ary boolean function protocol of [KWH15] and the copy protocol of [NNH$^+$15], use non-closed or non-uniform shuffles. As we have determined that uniform closed shuffles are a natural class of actions which can be done actively secure, it is interesting to find card-minimal protocols for these functions using only uniform closed shuffles.

Another natural problem is to implement more general shuffles, and even to characterize the shuffles which are possible with (a linear number of) helping cards, and the assumption of the security of a uniform random cut. To give one non-trivial example, we show in Appendix E how any subset of a cut can be implemented as a shuffle.

Moreover, it is interesting which functionalities are realizable – at all or more efficiently – in the two-player setting compared to the computational model of [MS14a], possibly by composing privatePerm operations in more sophisticated ways than done in Section 7. For example, the three-card AND protocol in [MWS15, Sect. 3.2], which is not in committed format, is naturally formalized in our two-player model.

References

[BMR07] J. Bohli, J. Müller-Quade, and S. Röhrich. “Bingo Voting: Secure and Coercion-Free Voting Using a Trusted Random Number Generator”. In: *VOTE-ID 2007*. Ed. by A. Alkassar and M. Volkamer. LNCS 4896. Springer, 2007, pp. 111–124. DOI: 10.1007/978-3-540-77493-8_10.

[Boe89] B. den Boer. “More Efficient Match-Making and Satisfiability: The Five Card Trick”. In: *EUROCRYPT ’89*. Ed. by J. Quisquater and J. Vandewalle. LNCS 434. Springer, 1989, pp. 208–217. DOI: 10.1007/3-540-46885-4_23.

[CHL13] E. Cheung, C. Hawthorne, and P. Lee. “CS 758 Project: Secure Computation with Playing Cards”. 2013. (Visited on 02/10/2015).

[ICM15] R. Ishikawa, E. Chida, and T. Mizuki. “Efficient Card-Based Protocols for Generating a Hidden Random Permutation Without Fixed Points”. In: *UCNC 2015*. Ed. by C. S. Calude and M. J. Dinneen. LNCS 9252. Springer, 2015, pp. 215–226. DOI: 10.1007/978-3-319-21819-9_16.

[KWH15] A. Koch, S. Walzer, and K. Härtel. “Card-based Cryptographic Protocols Using a Minimal Number of Cards”. In: *ASIACRYPT 2015, Part I*. Ed. by T. Iwata and J. H. Cheon. LNCS 9452. Springer, 2015, pp. 783–807. DOI: 10.1007/978-3-662-48797-6_32.

[Miz16] T. Mizuki. “Card-based protocols for securely computing the conjunction of multiple variables”. In: *Theoretical Computer Science* 622 (2016), pp. 34–44. DOI: 10.1016/j.tcs.2016.01.039.

[MKS12] T. Mizuki, M. Kumamoto, and H. Sone. “The Five-Card Trick Can Be Done with Four Cards”. In: *ASIACRYPT 2012*. Ed. by X. Wang and K. Sako. LNCS 7658. Springer, 2012, pp. 598–606. DOI: 10.1007/978-3-642-34961-4_36.

[MN06a] T. Moran and M. Naor. “Polling with Physical Envelopes: A Rigorous Analysis of a Human-Centric Protocol”. In: *EUROCRYPT 2006*. Ed. by S. Vaudenay. LNCS 4004. Springer, 2006, pp. 88–108. DOI: 10.1007/11761679_7.

[MN06b] T. Moran and M. Naor. “Receipt-Free Universally-Verifiable Voting with Everlasting Privacy”. In: *CRYPTO 2006*. Ed. by C. Dwork. LNCS 4117. Springer, 2006, pp. 373–392. DOI: 10.1007/11818175_22.

[MN10] T. Moran and M. Naor. “Basing cryptographic protocols on tamper-evident seals”. In: *Theoretical Computer Science* 411.10 (2010), pp. 1283–1310. DOI: 10.1016/j.tcs.2009.10.023.

[MS09] T. Mizuki and H. Sone. “Six-Card Secure AND and Four-Card Secure XOR”. In: *FAW 2009*. Ed. by X. Deng, J. E. Hopcroft, and J. Xue. LNCS 5598. Springer, 2009, pp. 358–369. DOI: 10.1007/978-3-642-02270-8_36.

[MS14a] T. Mizuki and H. 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.

[MS14b] T. Mizuki and H. Shizuya. “Practical Card-Based Cryptography”. In: *FUN 2014*. Ed. by A. Ferro, F. Luccio, and P. Widmayer. LNCS 8496. Springer, 2014, pp. 313–324. DOI: 10.1007/978-3-319-07890-8_27.

[MWS15] A. Marcedone, Z. Wen, and E. Shi. *Secure Dating with Four or Fewer Cards*. 2015. Cryptology ePrint Archive, Report 2015/1031.

[NHMS15] T. Nishida, Y. Hayashi, T. Mizuki, and H. Sone. “Card-Based Protocols for Any Boolean Function”. In: *TAMC 2015*. Ed. by R. Jain, S. Jain, and F. Stephan. LNCS 9076. Springer, 2015, pp. 110–121. DOI: 10.1007/978-3-319-17142-5_11.

[NHMS16] A. Nishimura, Y.-i. Hayashi, T. Mizuki, and H. Sone. “An Implementation of Non-Uniform Shuffle for Secure Multi-Party Computation". In: *Workshop on ASIA Public-Key Cryptography, Proceedings*. AsiaPKC '16. New York, NY, USA: ACM, 2016, pp. 49–55. DOI: 10.1145/2898420.2898425.

[NNH+15] A. Nishimura, T. Nishida, Y. Hayashi, T. Mizuki, and H. Sone. "Five-Card Secure Computations Using Unequal Division Shuffle". In: *TPNC 2015*. Ed. by A. H. Dediu, L. Magdalena, and C. Martín-Vide. LNCS 9477. Springer, 2015, pp. 109–120. DOI: 10.1007/978-3-319-26841-5_9.

[NR98] V. Niemi and A. Renvall. "Secure Multiparty Computations Without Computers". In: *Theoretical Computer Science* 191.1-2 (1998), pp. 173–183. DOI: 10.1016/S0304-3975(97)00107-2.

[PH10] S. Popoveniuc and B. Hosp. "An Introduction to PunchScan". In: *Towards Trustworthy Elections*. Ed. by D. C. et al. LNCS 6000. Springer, 2010, pp. 242–259. DOI: 10.1007/978-3-642-12980-3_15.

[SMS+15] K. Shinagawa, T. Mizuki, J. C. N. Schuldt, K. Nuida, N. Kanayama, T. Nishide, G. Hanaoka, and E. Okamoto. "Secure Multi-Party Computation Using Polarizing Cards". In: *IWSEC 2015*. Ed. by K. Tanaka and Y. Suga. LNCS 9241. Springer, 2015, pp. 281–297. DOI: 10.1007/978-3-319-22425-1_17.

[Sti01] A. Stiglic. "Computations with a deck of cards". In: *Theoretical Computer Science* 259.1-2 (2001), pp. 671–678. DOI: 10.1016/S0304-3975(00)00409-6.

[UNH+16] I. Ueda, A. Nishimura, Y.-i. Hayashi, T. Mizuki, and H. Sone. "How to Implement a Random Bisection Cut". In: *Theory and Practice of Natural Computing, TPNC 2016, Proceedings*. Ed. by C. Martín-Vide, T. Mizuki, and M. A. Vega-Rodríguez. Springer International Publishing, 2016, pp. 58–69. DOI: 10.1007/978-3-319-49001-4_5.

[Ver14] T. Verhoeff. "The Zero-Knowledge Match Maker". 2014. URL: https://www.win.tue.nl/~wstomv/publications/liber-AMiCorum-arjeh-bijdrage-van-tom-verhoeff.pdf.

A Analysis of the Protocol for Generalized Coupled Rotations

We proof that the protocol in Fig. 6 implements generalized coupled rotations as stated by verifying the loop invariant from page 11 by induction.

$i = 0$. The protocol starts with all registers containing the value 0. In the first action, Alice picks $0 \le k < t_0$ and performs $\pi = \varphi_0^k \circ \psi_{\mathsf{store}}^k$. Clearly, $\pi|_{\{1,\dots,n\}} = \varphi_0^k$ and in $\pi(S)$ the store register contains k with both other registers containing 0. This establishes the invariant for $i = 0$, i.e. before the first execution of the loop.

$i \to i + 1$. Assume the loop invariant holds for i. In the beginning of the i+1st loop, the contents of store and temp are swapped, which leads to temp containing k, while the other registers contain 0. The permutation ψ_{copy} decrements the value of the temp register while simultaneously incrementing the value of store and main (each modulo t_0). Since the operation $(\mathsf{check}, x_0, \spadesuit)$ expects the value of temp to be 0, the only power of ψ_{copy} that will allow the check to pass is k. Assuming this happens, temp and main both contain k, while temp contains 0. Similar to before, $\widehat{\varphi_i}$ decrements main modulo t_i and since the operation $(\mathsf{check}, x_0, \spadesuit)$ expects main to contain 0, the only power of $\widehat{\varphi_i}$ that allows the check to succeed is k. Afterwards, the current iteration of the loop permuted the object cards by φ_i^k and left store containing k while the other registers contain 0. This establishes the loop invariant.

The three actions following the loop are essentially the ℓ-th iteration of the loop without the copying step so it is straightforward to verify that $\pi \in S_n$ is compatible with the protocol, iff $\pi|_{\{1,\ldots,n\}} = \varphi_\ell^k \circ \ldots \circ \varphi_0^k$ for some $0 \leq k < t_0$.

B The Issue of Reusing Helping Cards

Assume we already implemented some permutation protocols $\mathcal{P}_1$ and $\mathcal{P}_2$ for permutation sets Π_1 and Π_2 using some helping decks $\mathcal{H}_1$ and $\mathcal{H}_2$. Now we design another permutation protocol $\mathcal{P}_3$ implementing Π_3 and using its own deck of helping cards $\mathcal{H}_3$. Assume some $\mathsf{privatePerm}$ actions of $\mathcal{P}_3$ involve Π_1 and Π_2 and we intend use $\mathcal{P}_1$ and $\mathcal{P}_2$ as "subroutines". It is hence interesting to ask, what helping deck do we need for $\mathcal{P}_3$ in total.

Within $\mathcal{P}_3$ the deck $\mathcal{H}_3$ is *in use*, potentially encoding important information, so unless we make further assumptions, subroutines must treat those cards as *object cards*. If, however, the subroutines $\mathcal{P}_1$ and $\mathcal{P}_2$ are used sequentially, they may share resources. So all in all, we need $(\mathcal{H}_1 \cup \mathcal{H}_2) + \mathcal{H}_3$.

This assumes that the required helping cards from $\mathcal{H}_1$ can be re-used in $\mathcal{P}_2$ after they were used in $\mathcal{P}_1$. In particular, they need to be turned, which assumes that the arrangement of $\mathcal{H}_1$ after use does not contain sensitive information any more. This is reasonable: Not only do all of our own protocols end with the helping cards in canonical order, it would also be easy to destroy any information encoded in them by shuffling them after use, e.g. by using repeated uniform cuts.

C On the Interplay of our Definitions

Note that our definition of *implemented with active security and choice* makes sure that the $\mathsf{privatePerm}$ actions of a two player protocol can be executed in a way that makes our security definitions adequate.

Consider some action $(\mathsf{privatePerm}, \text{Alice}, \Pi, \mathcal{F}(\cdot))$, e.g. $\Pi = \{\mathsf{id}, (1\,2\,3)(4\,5\,6\,7)\}$ with probabilities $\frac{1}{3}$ and $\frac{2}{3}$ for the two options. The fact that Π is implemented with active security and choice (by virtue of being subset of a coupled rotation) guarantees:

– Alice may choose $\pi \in \Pi$ according to the distribution (privately in her head or using some private random device) and follow a sequence of actions that ultimately will permute the current sequence of cards by π. She will remember π for later, in particular, we record π in her permutation trace and let future distributions depend on π.

– Alice's choice is hidden from Bob. Also, neither player learns anything about the current sequence as no cards from it are being turned. In particular it makes sense to *not* attribute any observations to the players during privatePerm operations (other than Alice "observing" her choice π).
– Alice cannot perform any $\pi \notin \mathit{\Pi}$, which justifies that Definition 3 only considers attacks that respect $\mathit{\Pi}$.
– The procedure implementing $\mathit{\Pi}$ fails gracefully if Alice cheats, without revealing anything about the object cards. Note that we do not consider the case of aborts in two-player protocols. This is reasonable: Firstly, the only thing an attacker can hope to achieve is destroy information (by shuffling cards in non-permitted ways, blowing his cover in the process) without learning anything he would not have learned otherwise. Secondly, all protocols we suggested can recover even if checks fail[6]. In our cases, the check operations would have to reveal an entire "register" of helping cards and would be succeeded by permutations that "fix" any discrepancy between expected result and actual result.

D Proof of Proposition 2 and the Main Theorem

Proof (Proposition 2). Consider an attack ξ on $\mathcal{P}$ as player $p \in \{1,2\}$ and let $I \in U$ be any random variable denoting an input and $(I, O, \mathcal{T}_1, \mathcal{T}_2, W)$ and $(I, O^\xi, \mathcal{T}_1^\xi, \mathcal{T}_2^\xi, W^\xi)$ be the execution traces of $\mathcal{P}$ and $\mathcal{P}^\xi$.

Let w be any path in Q with $\Pr[W^\xi = w] > 0$ and t the permutation trace that ξ prescribes for player p along w (whenever $W^\xi = w$, then $\mathcal{T}_p^\xi = t$). For any i, o we have:

$$\begin{aligned}\Pr[(I, O^\xi) = (i,o) \mid W^\xi = w] &= \Pr[(I, O^\xi) = (i,o) \mid (\mathcal{T}_p^\xi, W^\xi) = (t, w)]\\ &= \Pr[(I, O\ \ ) = (i,o) \mid (\mathcal{T}_p\ , W\ \ ) = (t, w)]\\ &= \Pr[(I, O\ \ ) = (i,o)].\end{aligned}$$

From the first to the second line, note that firstly, since w is finite, the sequence t of choices is finite as well, so, using the assumption that $\mathsf{supp}(\mathcal{F}(\mathcal{T}_p)) = \mathit{\Pi}$ in all cases, there is some positive probability that an honest player behaves exactly like the attacker with respect to this finite sequence of choices. Therefore, the conditional probability in the second line is well defined. Secondly, the attacked protocol and the original protocol behave alike once we fix the behavior of player p so we have the stated equality. From the second to the third line we use the passive security of $\mathcal{P}$. □

[6] Admittedly, we cannot recover from Alice flipping over the table or running off with the cards, though.

Proof (Main Theorem). As already mentioned, we obtain $\hat{\mathcal{P}}$ by replacing each shuffle action in $\mathcal{P}$ by two $\mathsf{privatePerm}$ actions. More precisely, let $(v_1, v_2, \ldots)$ be the sequence of those vertices in Q with shuffle actions. Then for each i we have $A(v_i) = (\mathsf{shuffle}, \Pi_i, \mathcal{U}_i)$, where Π_i is some group and $\mathcal{U}_i$ the uniform distribution on Π_i. The tree $\hat{Q}$ is obtained from Q by replacing each v_i with two vertices $v_i^{(1)}$ and $v_i^{(2)}$ with $\hat{A}(v_i^{(p)}) = (\mathsf{privatePerm}, p, \Pi_i, \mathcal{U}_i)$, where $v_i^{(2)}$ is the child of $v_i^{(1)}$ and $p \in \{1, 2\}$. All other vertices remain unchanged.

By Proposition 1, each Π_i is implemented using the stated helping decks.

Permutation schemes. To simplify the following argument we shall pick all relevant permutations a priori instead of producing them on-demand: A *permutation scheme* is a sequence $(\pi_1, \pi_2, \ldots)$ of permutations with $\pi_i \in \Pi_i$. We shall imagine that $\mathcal{P}$ is executed by first choosing a permutation scheme $\mathcal{T} = (\pi_1, \pi_2, \ldots)$ uniformly at random (each π_i uniformly at random from Π_i and independent of the rest) and then executing the protocol as usual, except that we are already determined in our choice of permutations. When reaching a shuffle action in vertex v_i we are already determined to use π_i.
Clearly, this does not affect the execution of $\mathcal{P}$ in the following sense. If I is a random input then the tuple $(I, O, W, \mathcal{T}^W)$ of input, output, execution path and permutation trace, resulting from this new way of executing $\mathcal{P}$ has the same distribution as the ordinary permutation trace of $\mathcal{P}$. By $\mathcal{T}^W$ we mean the projection of the scheme $\mathcal{T}$ to those components used in the execution, i.e. those on vertices occurring in W (in descending order).
In the same way we think of the execution of $\hat{\mathcal{P}}$, having players 1 and 2 pick permutation schemes $\mathcal{T}_1 = (\pi_1, \pi_2, \ldots)$ and $\mathcal{T}_2 = (\tau_1, \tau_2, \ldots)$ in advance and having player 1 use π_i if vertex $v_i^{(1)}$ is encountered and player 2 use τ_i if vertex $v_i^{(2)}$ is encountered. Then the tuple $(I, \hat{O}, \hat{W}, \mathcal{T}_1^{\hat{W}}, \mathcal{T}_2^{\hat{W}})$ of input, output, execution path and permutation traces obtained from this new way of executing $\hat{\mathcal{P}}$ has the same distribution as the ordinary execution trace of $\hat{\mathcal{P}}$. We use these modified execution traces in the following.

Computing the same function. In the following we shall make heavy use of the fact that $X \perp Y$ implies $f(X) \perp g(Y)$ whenever X and Y are (vectors of) random variables, f and g deterministic functions and $\perp$ denotes independence of random variables.
Note that O is determined by $(I, \mathcal{T})$, meaning there is a deterministic function f with $O = f(I, \mathcal{T})$. By construction, any permutation done by player 1 in $\hat{\mathcal{P}}$ is immediately followed by a corresponding permutation done by player 2 and we see $\hat{O} = f(I, \mathcal{T}_2 \circ \mathcal{T}_1)$. Clearly, the folded permutation scheme $\mathcal{T}_2 \circ \mathcal{T}_1$ has the same distribution as $\mathcal{T}$, so $\hat{O}$ has the same distribution as O. Since we made no assumptions on I, we conclude that $\mathcal{P}$ and $\hat{\mathcal{P}}$ compute the same randomized function.

Passive Security. Note that W is determined by $(I, \mathcal{T})$, meaning there is a deterministic function g with $W = g(I, \mathcal{T})$. If ext is the function acting on paths by replacing occurrences v_i with the two vertices $v_i^{(1)}$ and $v_i^{(2)}$ then we have by construction $\hat{W} = (\mathsf{ext} \circ g)(I, \mathcal{T}_2 \circ \mathcal{T}_1)$. We now see that $(I, O, W) =$

$(I, f(I,\mathcal{T}), g(I,\mathcal{T}))$ has exactly the same distribution as $(I, \hat{O}, \mathsf{ext}^{-1}(\hat{W})) = (I, f(I, \mathcal{T}_2 \circ \mathcal{T}_1), g(I, \mathcal{T}_2 \circ \mathcal{T}_1))$. Therefore, the passive security of $\mathcal{P}$ reflected in $(I, O) \perp W$ translates into $(I, \hat{O}) \perp \mathsf{ext}^{-1}(\hat{W})$ which implies $(I, \hat{O}) \perp \hat{W}$. This is the crucial step in the following chain of reasoning:

$\mathcal{T}_p \perp \mathcal{T}_2 \circ \mathcal{T}_1$	Players choice masked by other players choice
$\Rightarrow \mathcal{T}_p \perp (I, \mathcal{T}_2 \circ \mathcal{T}_1)$	All schemes are chosen a priori, independent of I
$\Rightarrow \mathcal{T}_p \perp (I, \mathcal{T}_2 \circ \mathcal{T}_1, \hat{O}, \hat{W})$	Since $\hat{O} = f(I, \mathcal{T}_2 \circ \mathcal{T}_1)$ and $\hat{W} = (\mathsf{ext} \circ g)(I, \mathcal{T}_2 \circ \mathcal{T}_1)$
$\Rightarrow \mathcal{T}_p \perp (I, \hat{O}, \hat{W})$	Projection
$\Rightarrow (\mathcal{T}_p, \hat{W}) \perp (I, \hat{O})$	Using $\hat{W} \perp (I, \hat{O})$
$\Rightarrow (\mathcal{T}_p^{\hat{W}}, \hat{W}) \perp (I, \hat{O})$	$\mathcal{T}_p^{\hat{W}}$ is a function of $\mathcal{T}_p$ and $\hat{W}$.

This also shows the corresponding independence for the ordinary execution trace, proving passive security of $\hat{\mathcal{P}}$.

Active Security. This follows immediately from passive security and Proposition 2. □

E Implementing a Non-closed Shuffle Operation

Our focus on uniform closed shuffle operations has its reasons, but this should not distract from the fact that many other shuffle operations are both important and implementable. Let us take a special case of $(\mathsf{shuffle}, \{\mathsf{id}, (1\ 2\ 3\ 4\ 5)^3\}, \mathcal{U})$. It was for instance put to use in [CHL13], albeit without further elaboration on its security or implementation.

Note that $\mathit{\Pi}$ has an implementation with active security and choice (by virtue of being the subset of a cut), but performing $(\mathsf{playerPerm}, p, \mathit{\Pi}, \mathcal{U}(\cdot))$ for $p \in \{1, 2\}$ one after the other as we do for closed permutation sets could result in the permutation $(1\,2\,3\,4\,5)^6$. However, we can use a similar idea as we did when implementing cuts. We propose the procedure shown in Fig. 8 which is a "protocol implementing a shuffle".

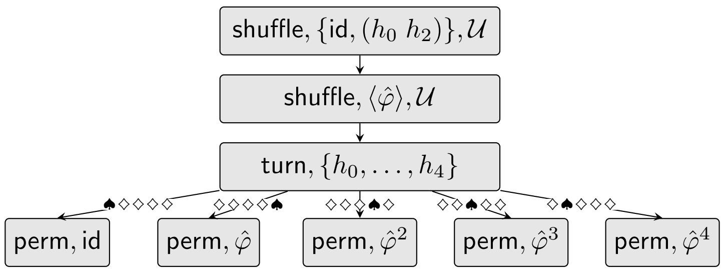

shuffle, $\{\mathsf{id}, (h_0\ h_2)\}, \mathcal{U}$
shuffle, $\langle\hat{\varphi}\rangle, \mathcal{U}$
turn, $\{h_0, \ldots, h_4\}$
♠◇◇◇◇
◇◇◇◇♠
◇◇◇♠◇
◇◇♠◇◇
◇♠◇◇◇
perm, id
perm, $\hat{\varphi}$
perm, $\hat{\varphi}^2$
perm, $\hat{\varphi}^3$
perm, $\hat{\varphi}^4$

Fig. 8: Protocol implementing $(\mathsf{shuffle}, \mathit{\Pi} = \{\mathsf{id}, (1\ 2\ 3\ 4\ 5)^3\}, \mathcal{U})$ with five helping cards.

We assume that we initially have five object cards in positions 1 through 5 and a helping deck $\mathcal{H} = [\![\spadesuit, 4{\cdot}\diamondsuit]\!]$ originally lying in positions h_0 through h_4 (say $h_i = i{+}6$). The $\spadesuit$ starts at position h_0, but after the first shuffle operation will end up at a position h_s, where s can be 0 or 2 with equal probability. We now perform some power of $\hat{\varphi} = (1\ 2\ 3\ 4\ 5) \circ (h_0\ h_1\ h_2\ h_3\ h_4)$ which rotates both the helping sequence and the object cards by some uniformly random $0 \leq k < 5$, leaving $\spadesuit$ in position $h_{s+k \pmod 5}$. The turn step reveals the helping cards and thereby $s + k \pmod 5$. Now $\hat{\varphi}^{-s-k \pmod 5}$ is performed, leaving the helping sequence in its original state and the object cards rotated by $k - s - k = -s$. Since $-s$ is with equal probability 0 or 3 (mod 5) a uniformly random permutation from Π happened as desired. The only information that was revealed is $s + k$ which is independent of $-s$. Note that the two involved shuffle operations are uniform closed and may therefore be implemented as in Section 7. With this, we implement the non-closed shuffle with more basic shuffle operations.

We are confident that a clean formalization and generalization of this concept is possible and excited about future research that explores what other shuffle operations can be implemented in this sense.

F Achieving Input Integrity

Mizuki and Shizuya [MS14b] consider malicious players disrespecting the input format, in their case by giving $\clubsuit\clubsuit$ or $\heartsuit\heartsuit$ as an input, even though only $\heartsuit\clubsuit$ and $\clubsuit\heartsuit$ are permitted. Note that we leave this problem aside when assuming that inputs are already lying on the table when the protocol starts.

It is beneficial for composability/modularity to not assume knowledge about the inputs from the players running the protocol, as they might work with hidden intermediate results of the surrounding protocol. We can think of two strategies to integrate a player input procedure into our model, encompassing input security. In both approaches sketched below, we assume a unique starting sequence s, which does not yet encode any inputs, and both players $p \in \{1, 2\}$ have their input I_p in mind. Distributions in playerPerm actions may depend on I_p.

- Introduce a separate input phase before the computation phase of the protocol. In this phase, the sequence s is transformed into a sequence that reflects the input of both players. We require that no player learns anything about the input of the other player and after this phase, the sequence of cards is an admissible input sequence reflecting the honest player's choices accurately, even when one player is malicious. In the case of committed format protocols where both players independently provide a sequence of input bits, it is easy to see that privatePerm-actions, where players perform chosen transpositions, are sufficient.
- The input phase and the computation phase are arbitrarily interleaved. This may allow to save cards if not all bits are needed at the same time. However, the notions of passive and active security would need to be updated, since now the output of the protocol is not necessarily independent of the permutation

trace of a player. In the case of passive security we would want only that the input of the other player and the output *conditioned on* I_p are protected. For active security the notion is more tricky still.

G Notes on Existing Literature and Liberties we Took

On Specifying Protocols as Trees. In previous literature protocols were usually finite state machines that, given a current machine state and a current observation, compute the next machine state and the next action. The reader may find it curious that we chose trees instead. There are several reasons.

– There is another very central kind of "state" namely the state of public knowledge captured for instance in the "state calculus" of [KWH15, Sect. 3] which has nothing to do with the state of the state machine. In our model, both notions coincide, as the current vertex of the execution in Q captures the result of all turns done in the past and also what happens next. This seems less confusing.
– When illustrating protocols, the authors where usually tempted to draw trees anyway (or directed acyclic graphs for recursive trees).
– Trees are somewhat easier to handle mathematically when talking about all possible strategies of a player which just have to assign one behavior for each vertex.
– State machines are associated with computability theory. But the computability of the protocol (the complexity of computing "what action should happen next") is completely beside the point.

The price to pay is that writing down complicated protocols with repeating parts is either cumbersome or in need of some creative notation.

Other Differences. *Turning back:* When we perform a turn action we implicitly turn all cards face-down again. In Mizuki and Shizuya [MS14a] this is not the case. There, a turned card lies face-up until it is explicitly turned again. Since we do not see any benefit in keeping cards face-up, we opted for our semantics so we do not have to keep track of which cards are face-up and which are face-down.

Randomized Flip: Mizuki and Shizuya [MS14a] proposed another operation rflip that chooses a set T at random and turns over the cards in positions from T. The authors state that "a random flip operation has not been used in [...] any [...] protocols, and at present its usefulness and potential power are unclear". We agree with their assessment, which is why we did not include this operation in our model.