

Foundations for Actively Secure Card-Based Cryptography

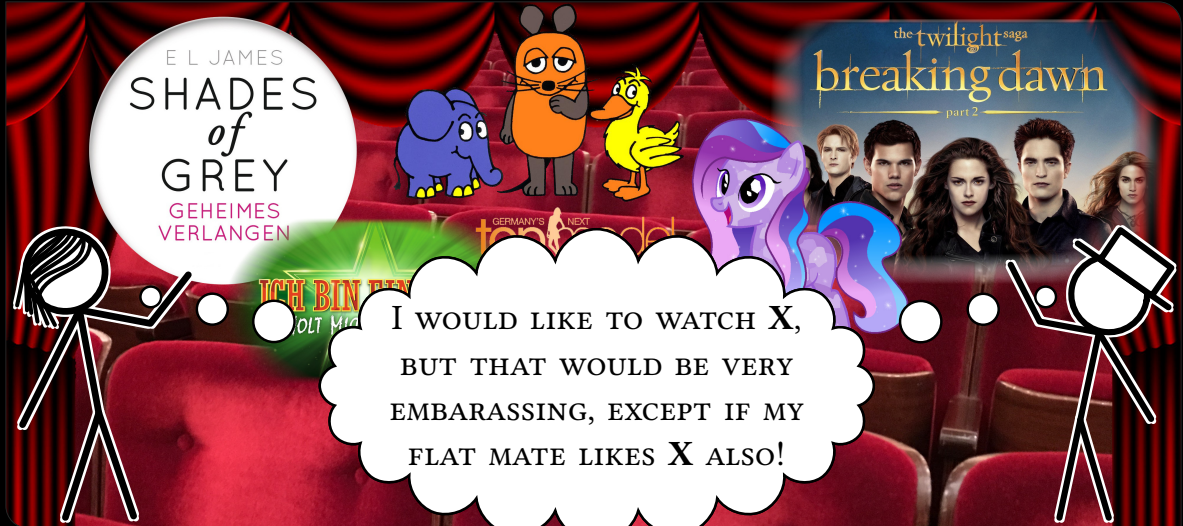
FUN 2020 @ FUN 2022 @ FUNvignana (Not my fav. FUN pun)

Alexander Koch (KIT) and Stefan Walzer (University of Cologne) | 31. May 2022



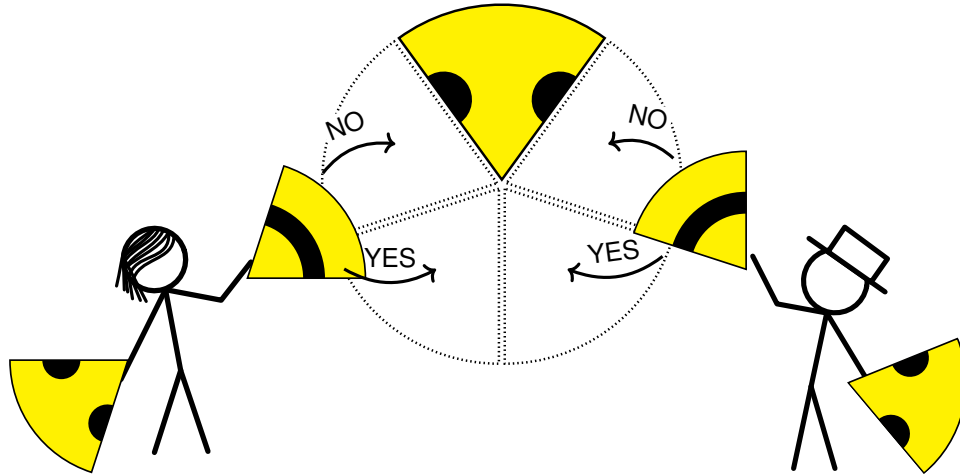
Parts from "Castel del Monte, Andria" by Luca Lombardi, CC BY-SA 4.0 Int.

Problems at a Movie Evening...



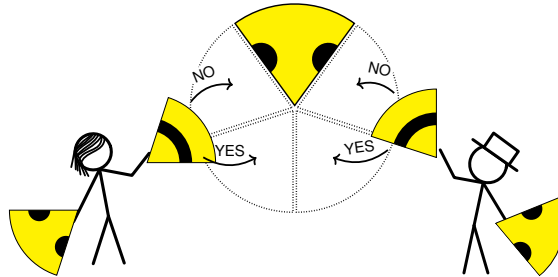
Graphic by S. Walzer, Seats/curtains CC0, Alice/Bob adapted from xkcd (by Randal Munroe) CC BY-NC 2.5, logos copyrighted

The “Five-Card Trick”

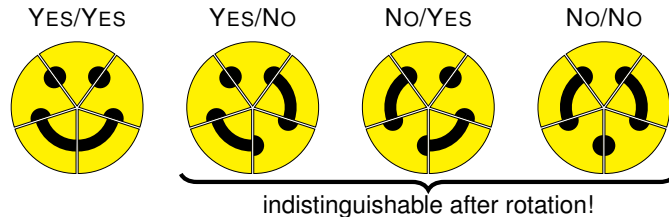


Den Boer (1989). Version with tiles by Verhoeff (2014), Graphic by S. Walzer

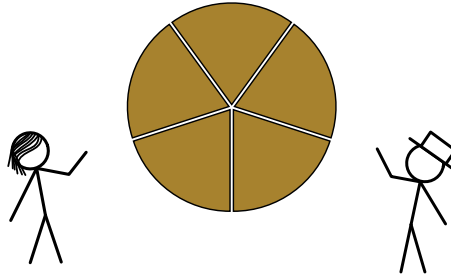
The “Five-Card Trick”



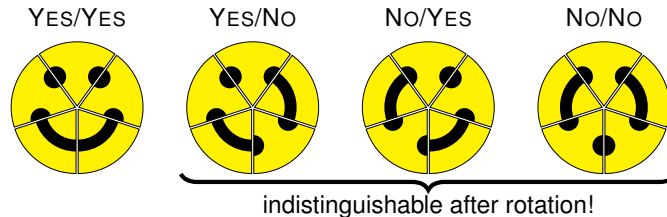
Configurations:



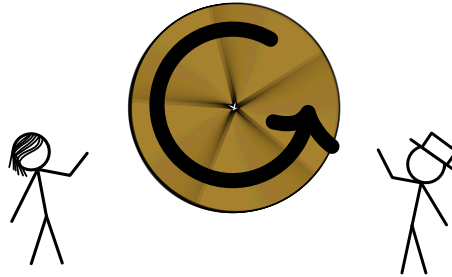
The “Five-Card Trick”



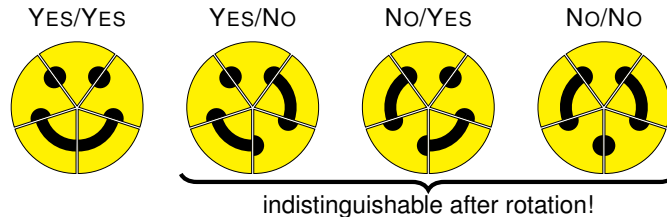
Configurations:



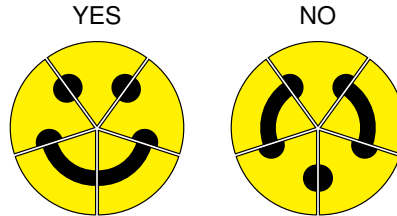
The “Five-Card Trick”



Configurations:

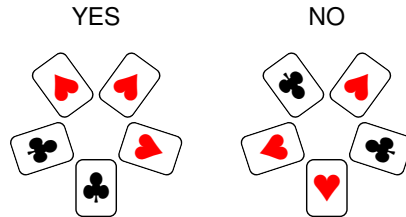


Revealing Tiles...



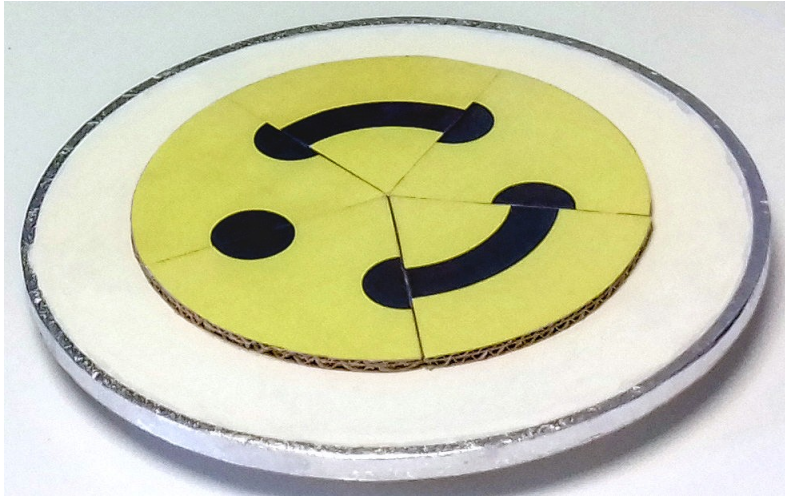
If a player said **NO**, then that player cannot know what the other player said!

Revealing Cards... (equivalent)



If a player said **NO**, then that player cannot know what the other player said!

“We have an implementation”

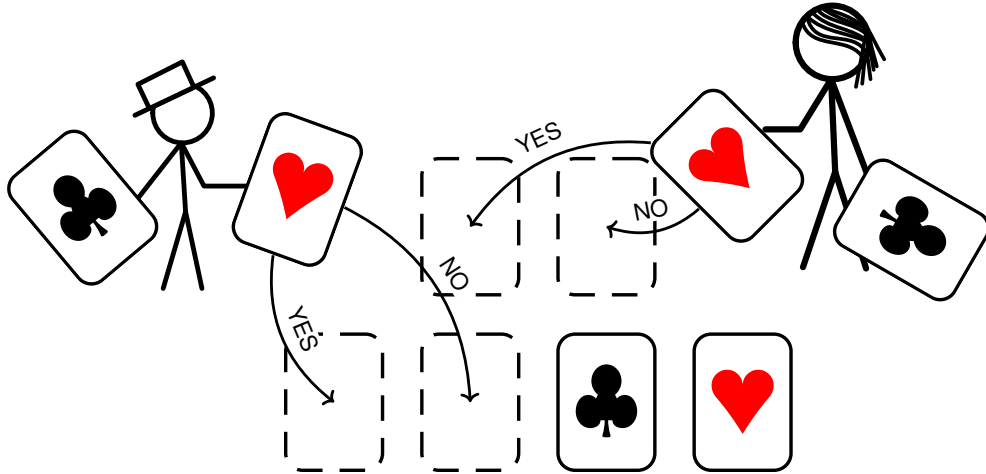


Motivation: Explain Cryptography to Students



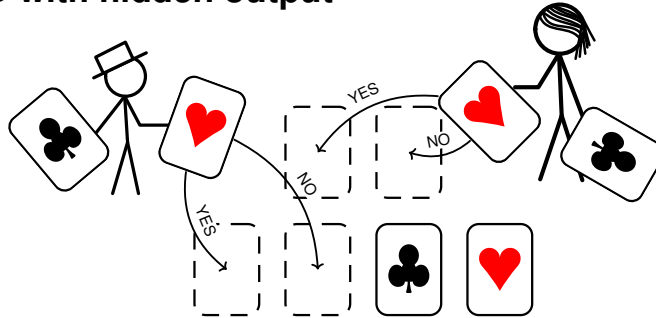
By brett jordan via flickr CC BY 2.0

Computing AND with hidden output

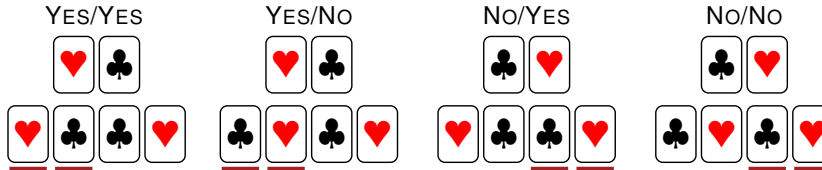


6-Card protocol by Mizuki and Sone (2009)

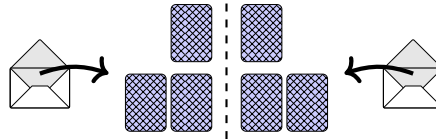
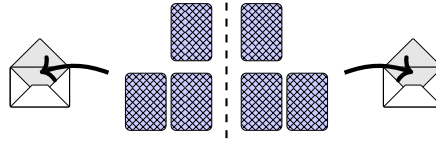
Computing AND with hidden output



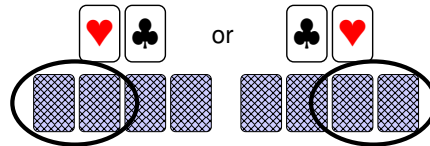
Configurations:



Computing AND with hidden output



Computing AND with hidden output



Research Question

We want to compute *arbitrary Boolean circuits*

- For this, the output needs to stay *hidden*, in 2-card encoding
- We need protocols for *AND*, *NOT* and *Bit-Copy*

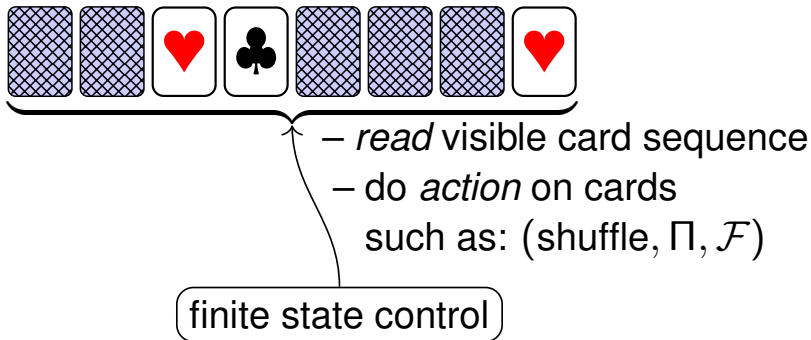
Question: What are the *best* protocol with hidden output?

Criteria:

- 1 Number of cards used
- 2 Running time behavior (finite vs. Las Vegas)
- 3 Number of steps
- 4 Practically of Shuffling steps

Motivation II: Studying Unconventional Machine Models

For Example, the Model of Mizuki and Shizuya, 2014



Context: Physical Assumptions in Cryptography

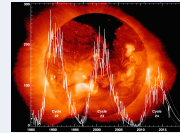
Idealized Hardware



Physical Objects



Physical Processes



Main advantages:

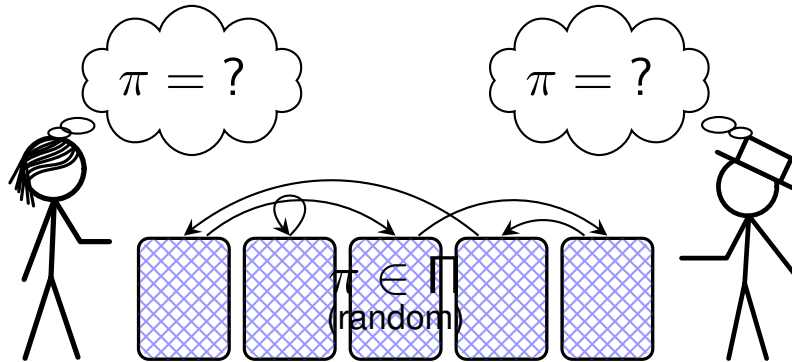
- Transparency and ease of understanding for the user
- Strong, otherwise unachievable, security guarantees

ID card CC0; TAN generator/Scratch-off cards copyrighted; "Miscellaneous Playing Cards" (excerpt) by Philippa Willitts CC BY-NC 2.0; Solar cycle CC0; Schrödinger's cat (excerpt) by ADA&Neagoe CC BY-SA 3.0

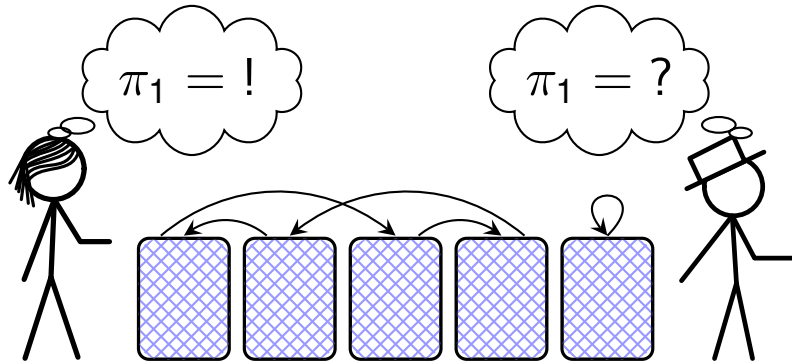
Motivation for our Work

- 1 Mizuki and Shizuya (2014) wanted to capture what can be done with card-based cryptographic protocols, with very general shuffles, but without a description on how they can be performed.
- 2 There have been many ad-hoc solutions, but you have to be careful
- 3 So-far, authors have mainly considered honest-but-curious security
- 4 Our Goal: Implement a large class of shuffles in an actively secure fashion

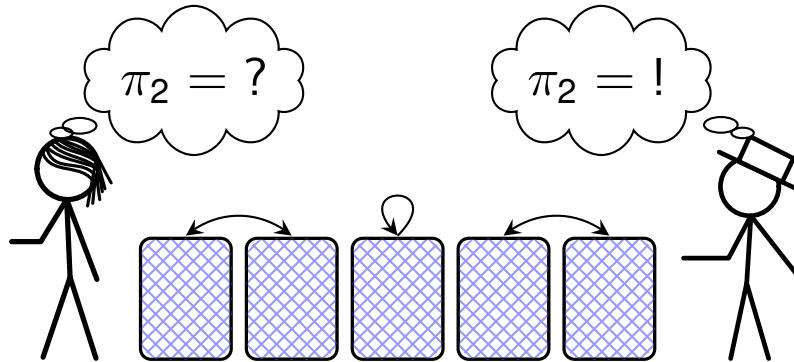
Shuffle from Group: Passively-Secure Implementation



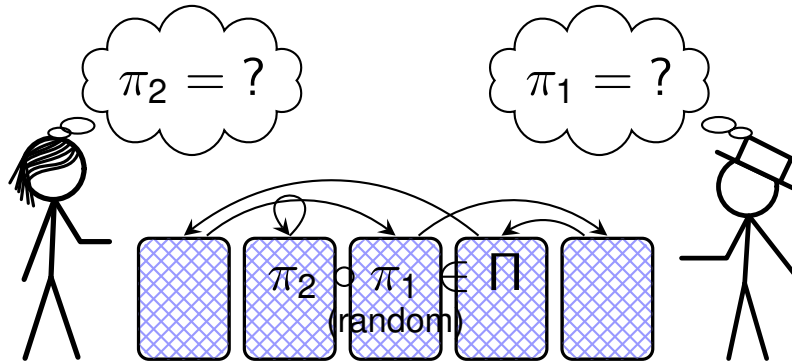
Shuffle from Group: Passively-Secure Implementation



Shuffle from Group: Passively-Secure Implementation



Shuffle from Group: Passively-Secure Implementation



Contribution

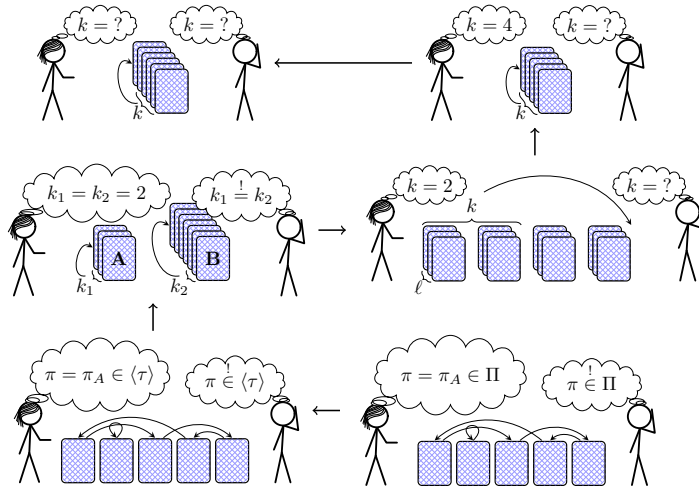
Attacker Model

Could maybe be characterized as “Dishonest but slow”

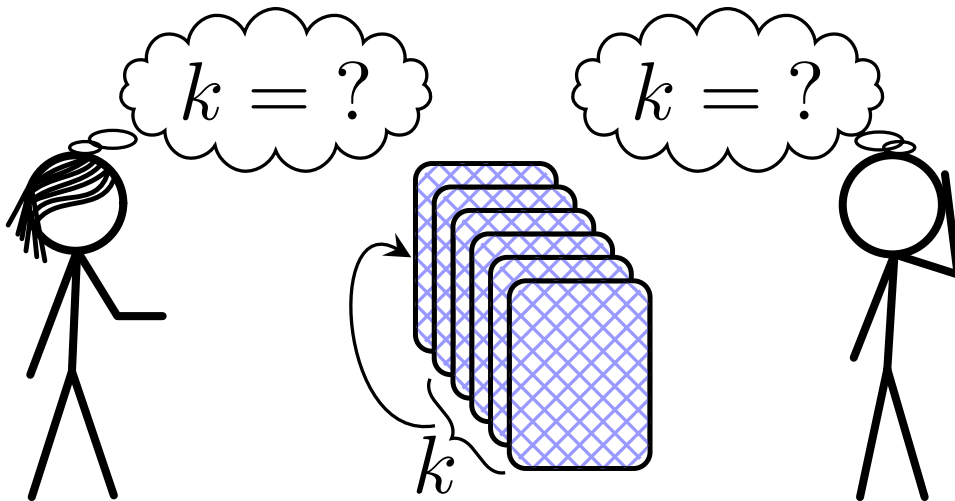
Main Theorem (informal)

Any uniform closed shuffle (i.e. a shuffle with a subgroup as shuffle set, and uniform distribution) can be implemented in the presence of an *active attacker*.

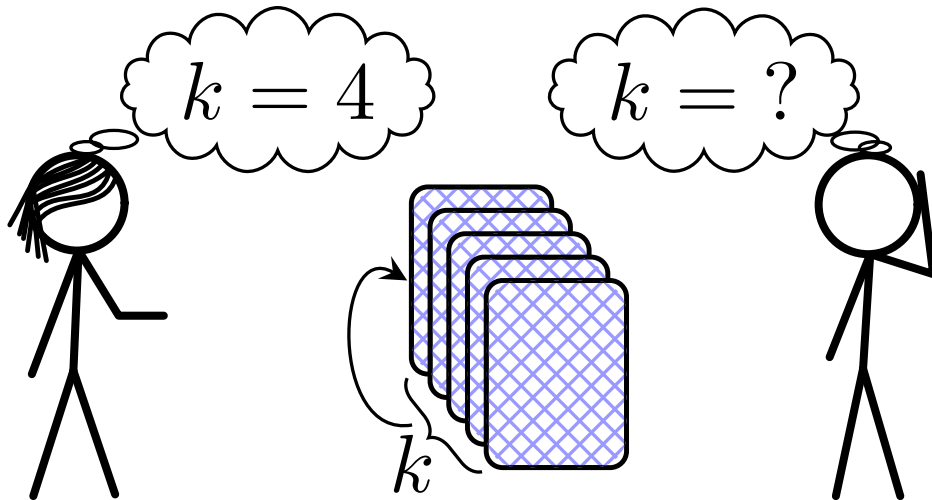
Overview of our Procedure



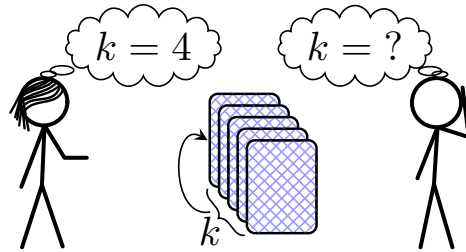
Uniform Cut



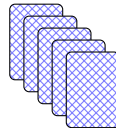
Chosen Cut



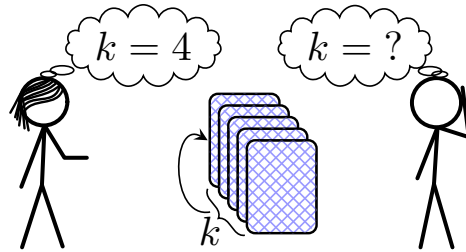
Chosen Cut



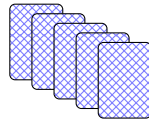
Implementation:



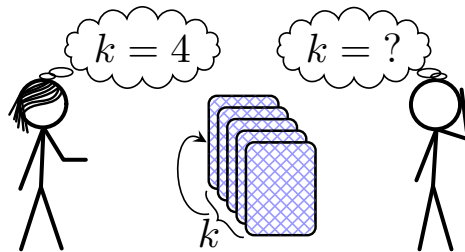
Chosen Cut



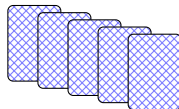
Implementation:



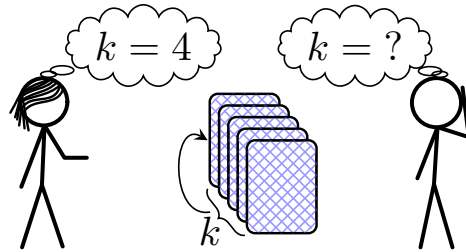
Chosen Cut



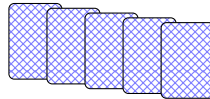
Implementation:



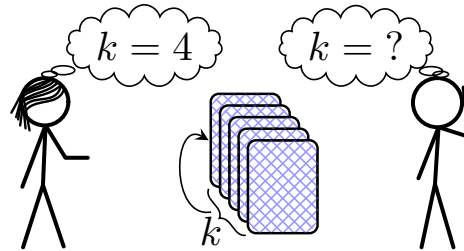
Chosen Cut



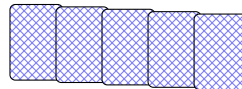
Implementation:



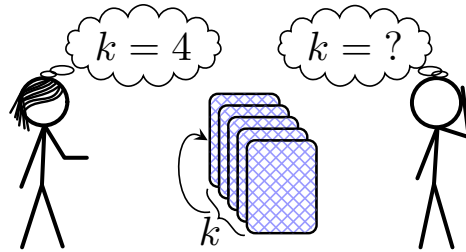
Chosen Cut



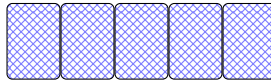
Implementation:



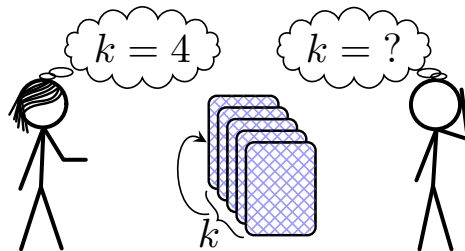
Chosen Cut



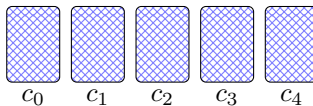
Implementation:



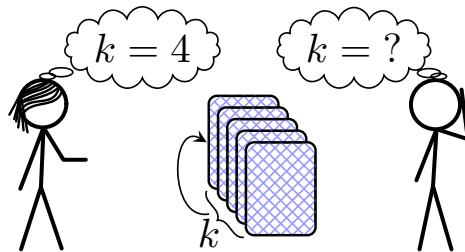
Chosen Cut



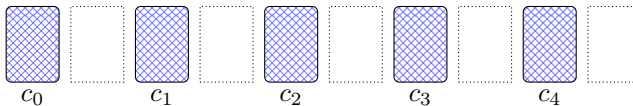
Implementation:



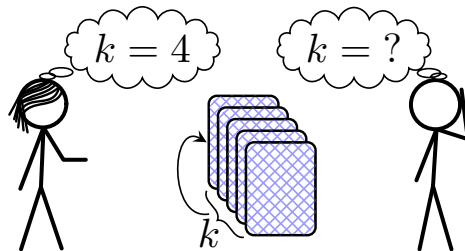
Chosen Cut



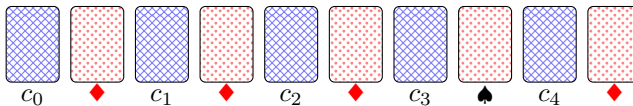
Implementation:



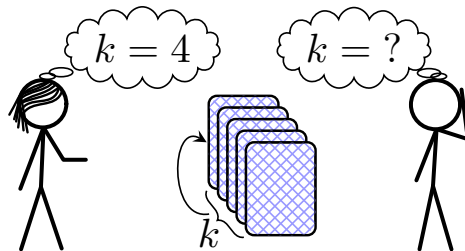
Chosen Cut



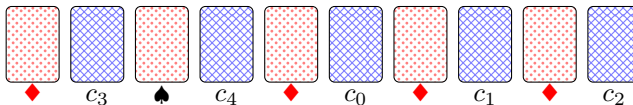
Implementation:



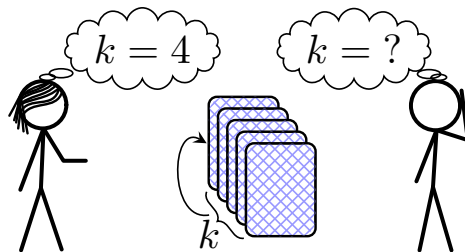
Chosen Cut



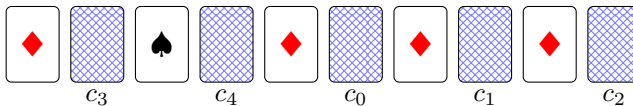
Implementation:



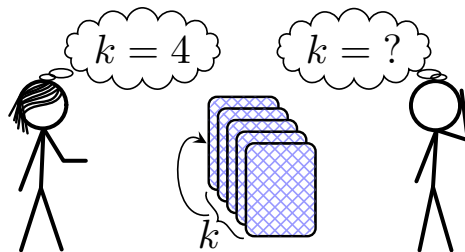
Chosen Cut



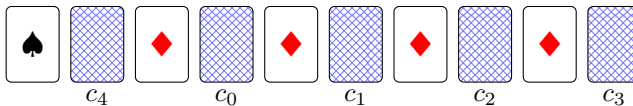
Implementation:



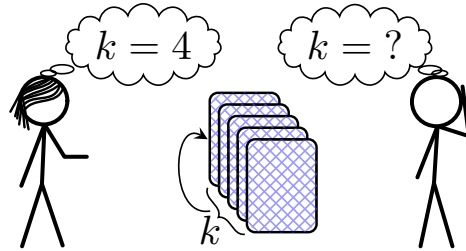
Chosen Cut



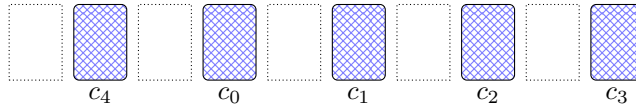
Implementation:



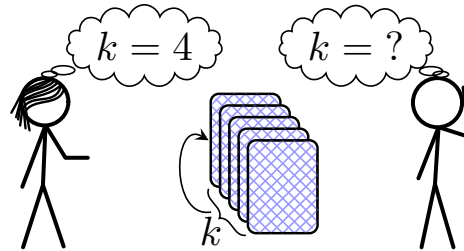
Chosen Cut



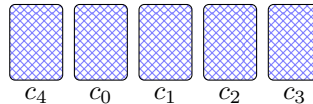
Implementation:



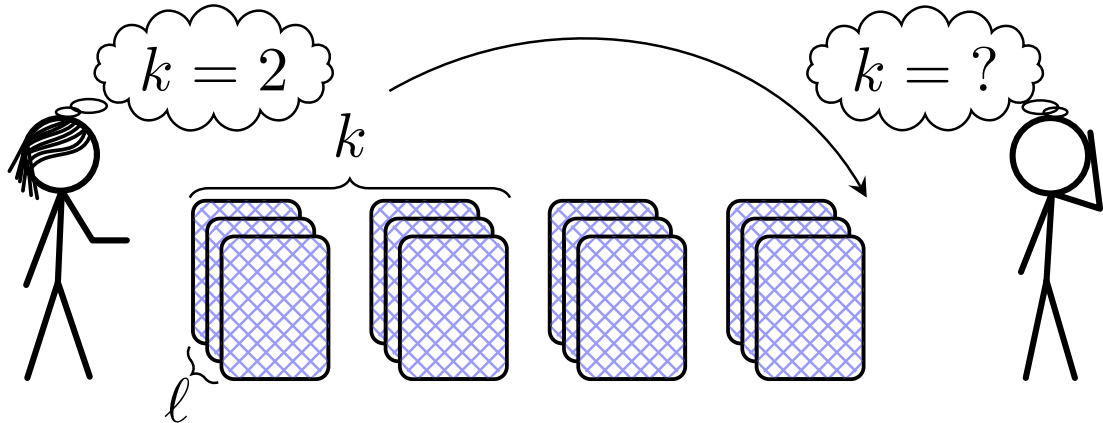
Chosen Cut



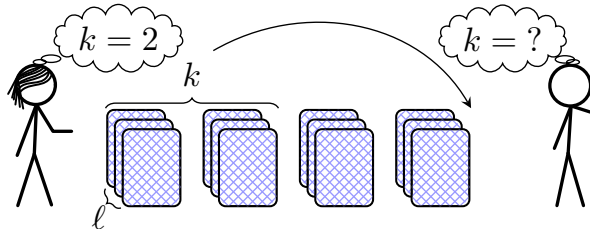
Implementation:



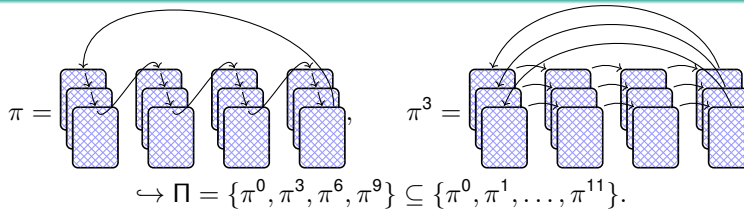
Chosen Pile Cut



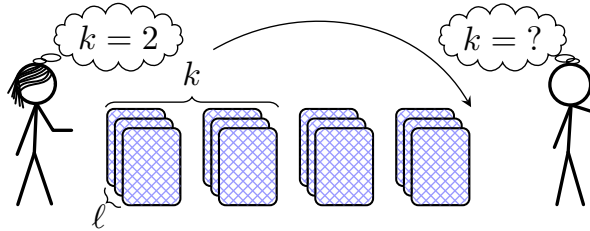
Chosen Pile Cut



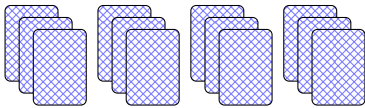
Implementation as Restricted Chosen Cut



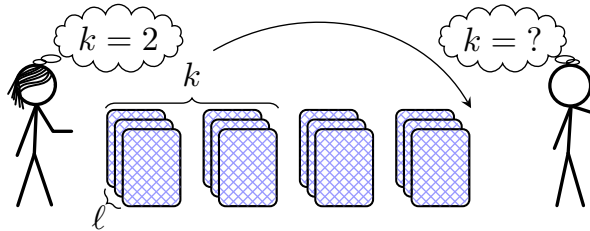
Chosen Pile Cut



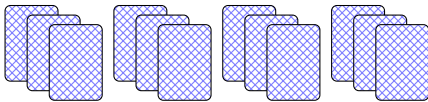
Visualization:



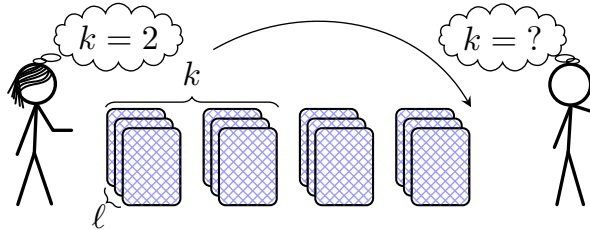
Chosen Pile Cut



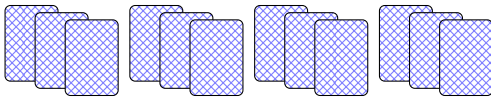
Visualization:



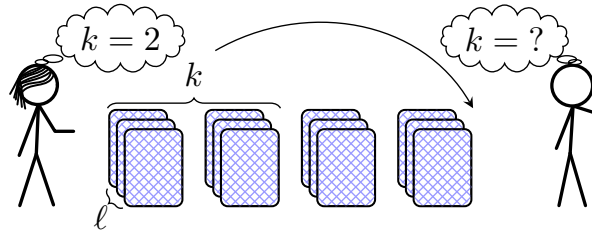
Chosen Pile Cut



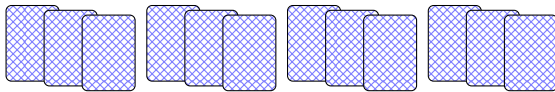
Visualization:



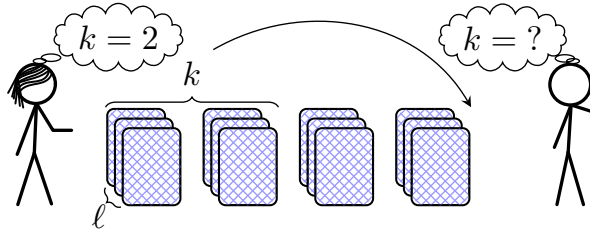
Chosen Pile Cut



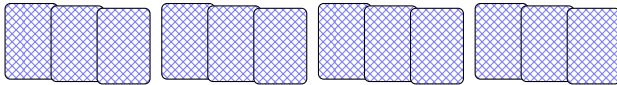
Visualization:



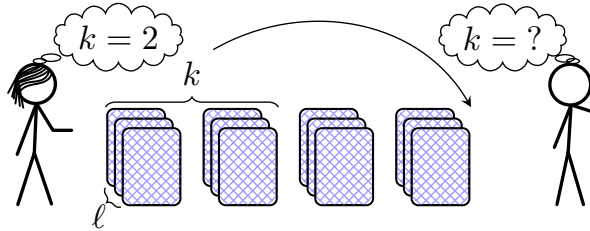
Chosen Pile Cut



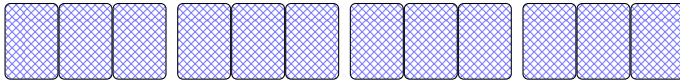
Visualization:



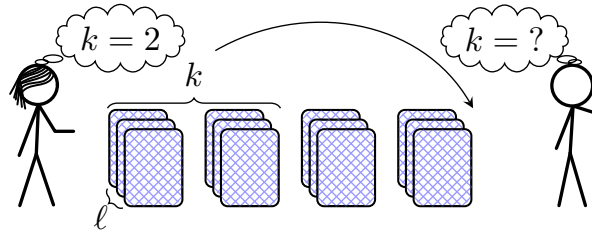
Chosen Pile Cut



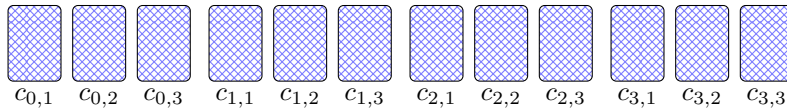
Visualization:



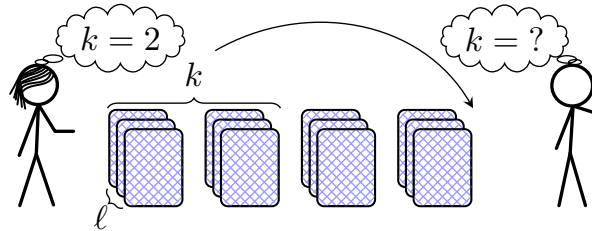
Chosen Pile Cut



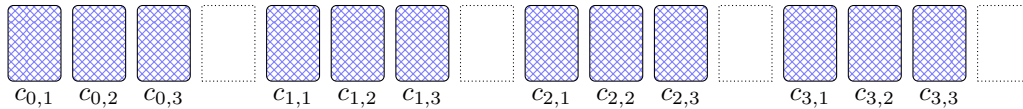
Visualization:



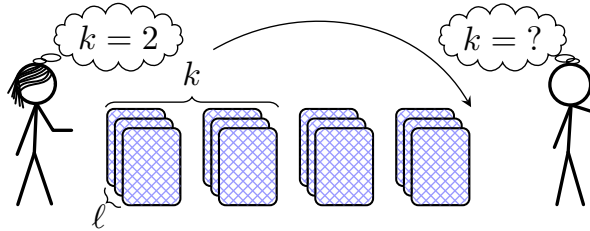
Chosen Pile Cut



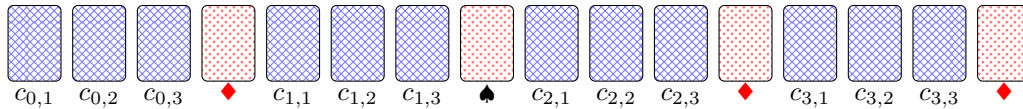
Visualization:



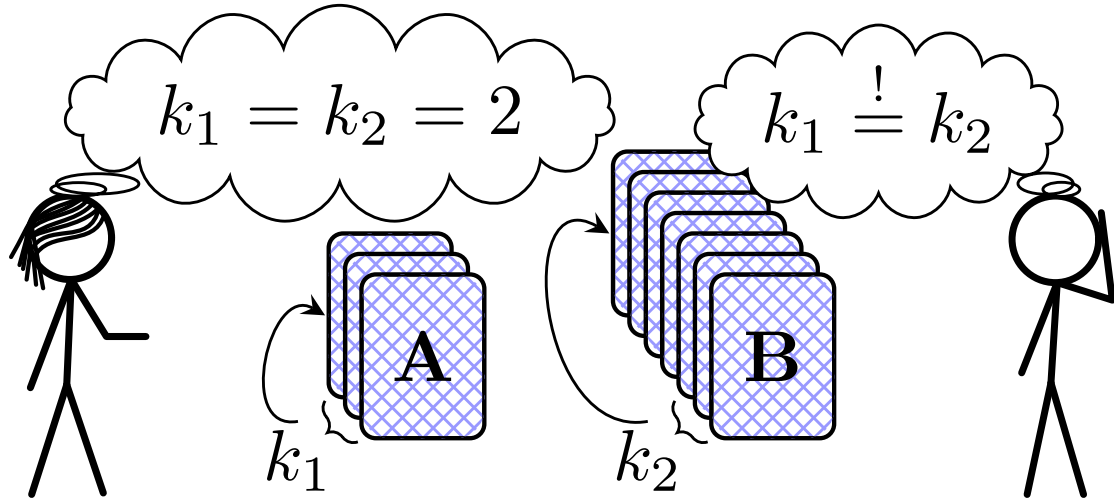
Chosen Pile Cut



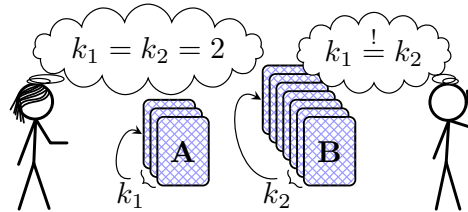
Visualization:




Coupled Rotation




Coupled Rotation



Implementation:

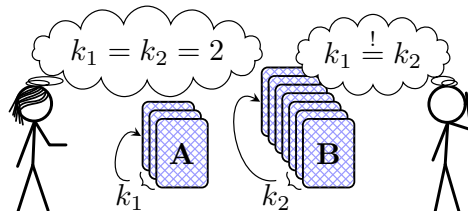
$A :$


 $a_0 \quad a_1 \quad a_2$

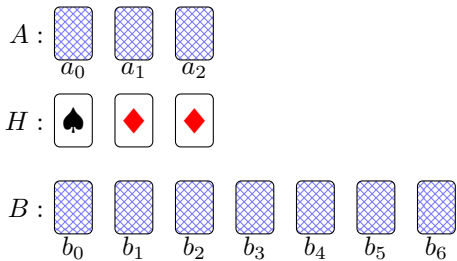
$B :$


 $b_0 \quad b_1 \quad b_2 \quad b_3 \quad b_4 \quad b_5 \quad b_6$

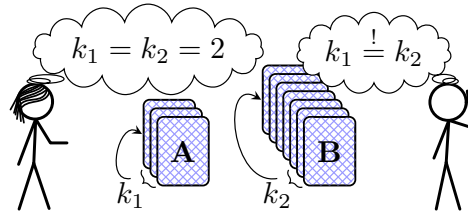
Coupled Rotation



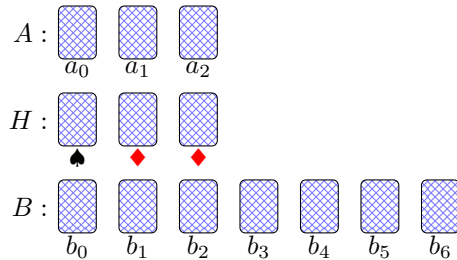
Implementation:



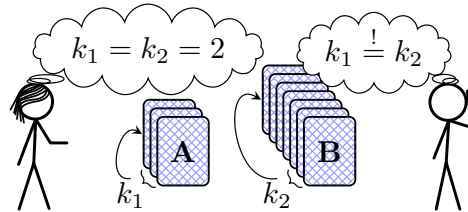
Coupled Rotation



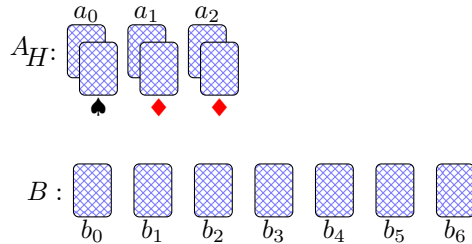
Implementation:



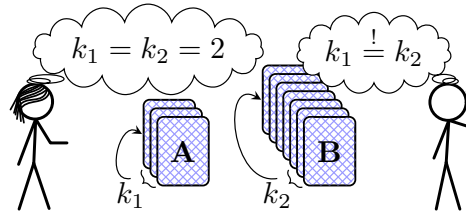
Coupled Rotation



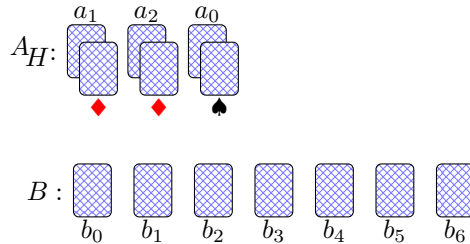
Implementation:



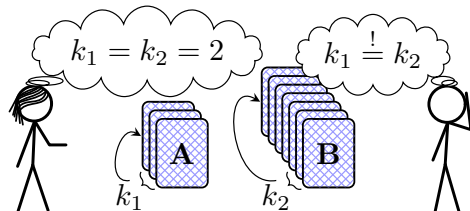
Coupled Rotation



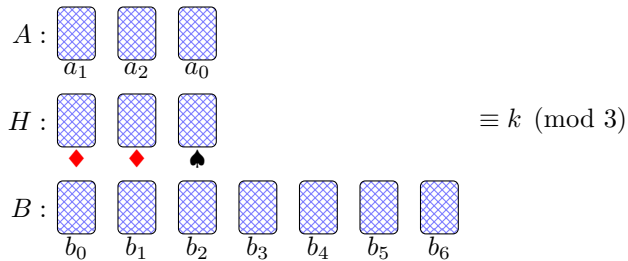
Implementation:



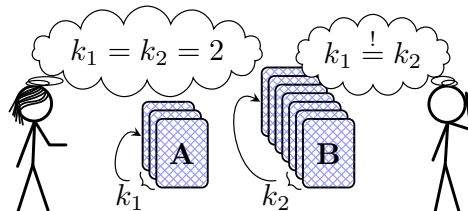
Coupled Rotation



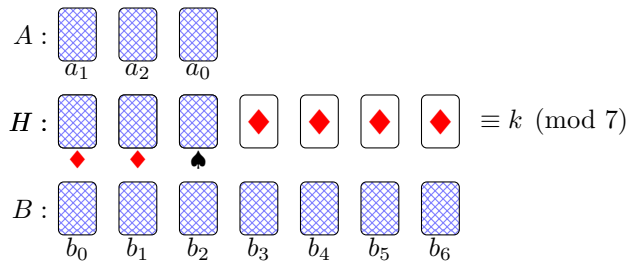
Implementation:



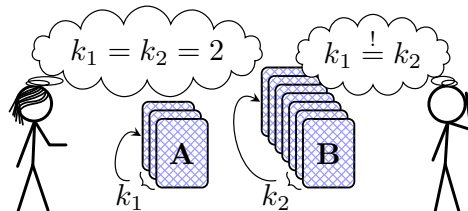
Coupled Rotation



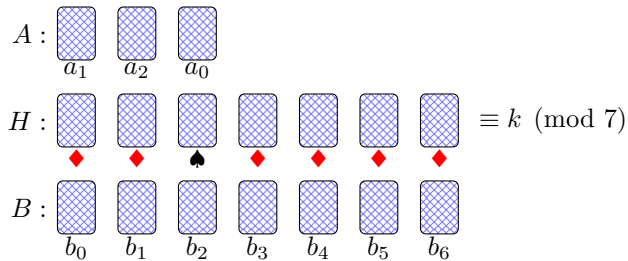
Implementation:



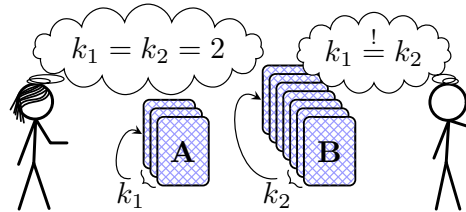
Coupled Rotation



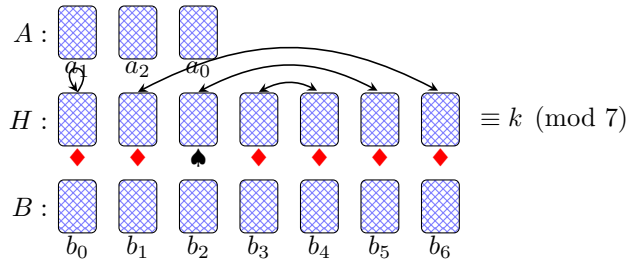
Implementation:



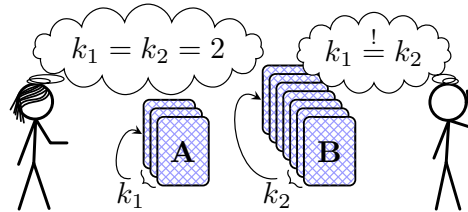
Coupled Rotation



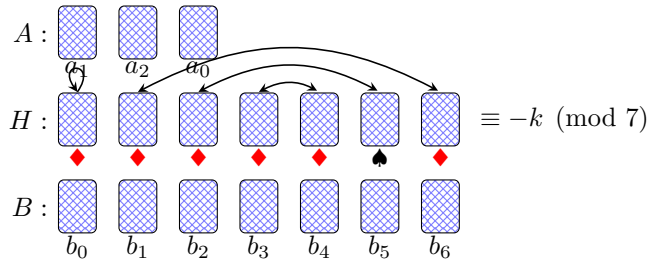
Implementation:



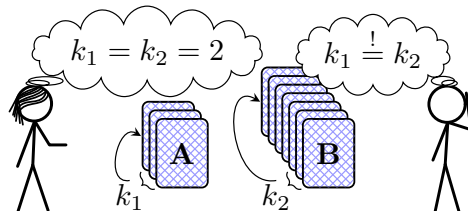
Coupled Rotation



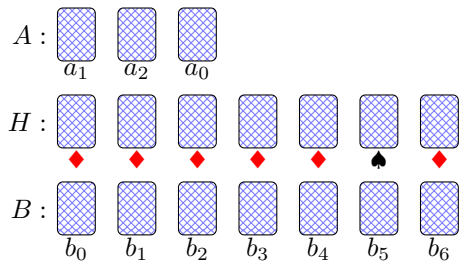
Implementation:



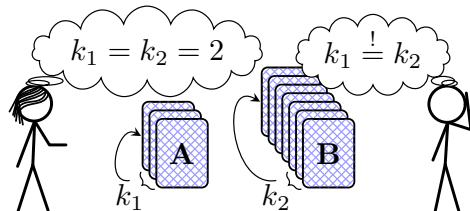
Coupled Rotation



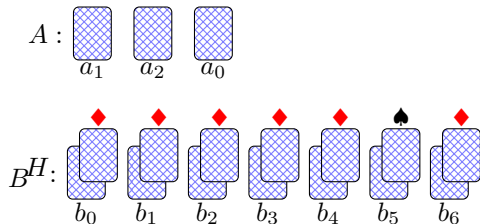
Implementation:



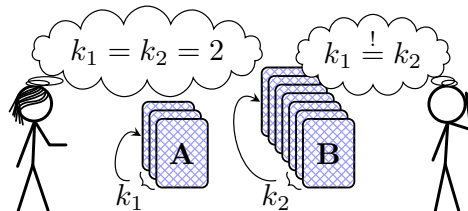
Coupled Rotation



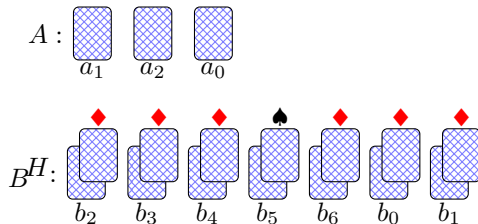
Implementation:



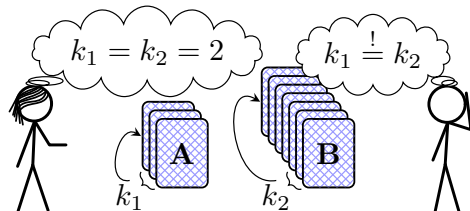
Coupled Rotation



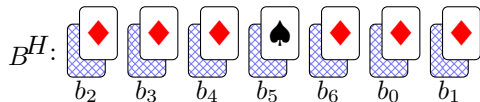
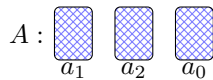
Implementation:



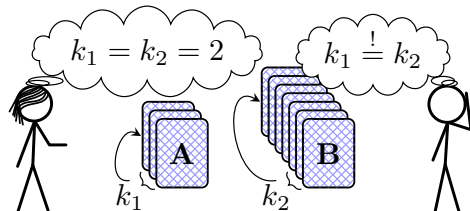
Coupled Rotation



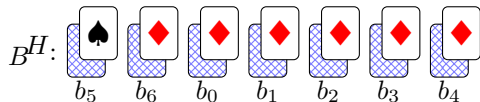
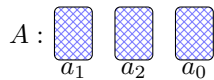
Implementation:



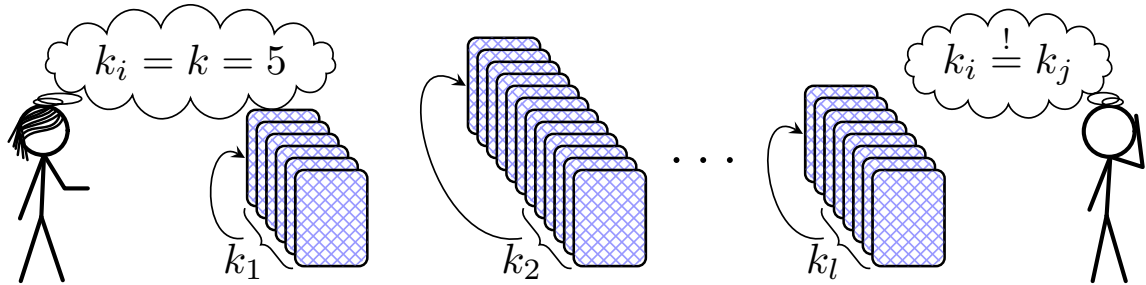
Coupled Rotation



Implementation:



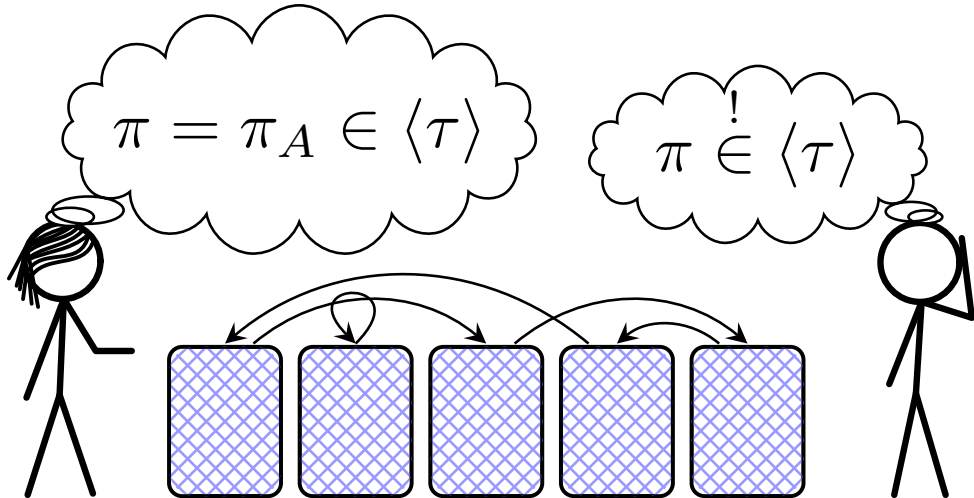
Generalized Coupled Rotation



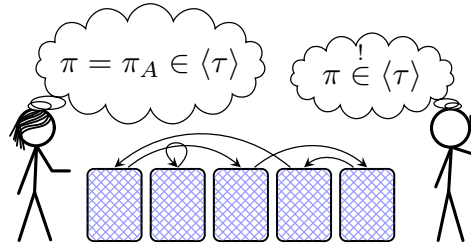
Implementation:

- Deal with piles one by one.
- Reuse the recorded k several times.

Chosen Permutation from Cyclic Group $\langle \tau \rangle$ for given $\tau \in S_n$



Chosen Permutation from Cyclic Group $\langle \tau \rangle$ for given $\tau \in S_n$



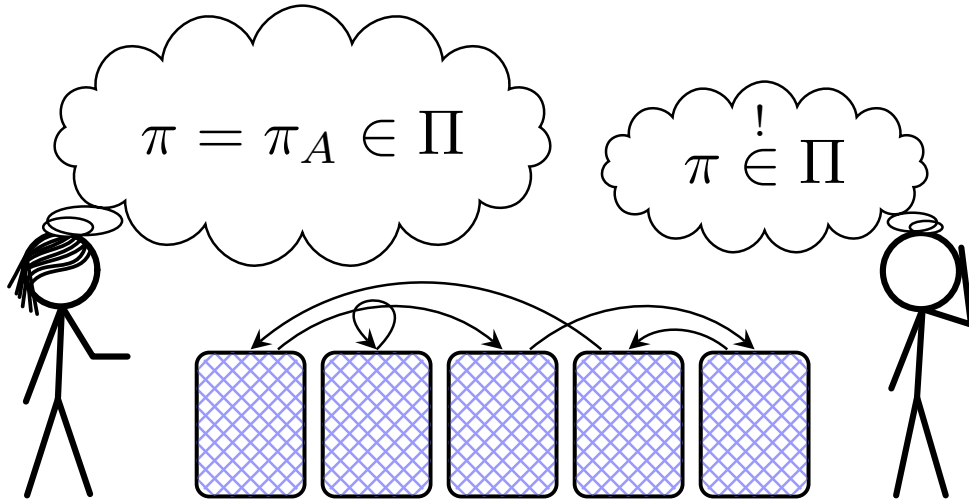
Implementation:

Assume $\tau = (1\ 2\ 3)(4\ 5\ 6\ 7\ 8\ 9)$.

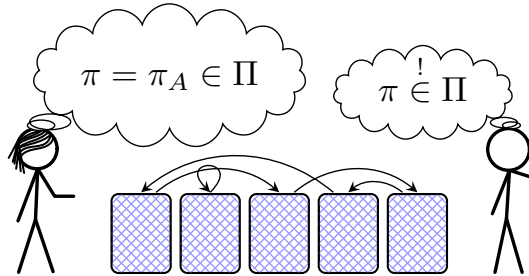
Let $\ell = 3$ be the length of a shortest non-trivial cycle in τ .

- $R = \{\tau^0, \tau^1, \dots, \tau^{\ell-1}\}$ is a generalised coupled rotation.
- $\langle \tau \rangle = \underbrace{R \circ R \circ \dots \circ R}_{\text{ord}(\tau)/\ell \text{ times}}$
- Perform $\text{ord}(\tau)/\ell$ coupled rotations with R .

Chosen Permutation from Group



Chosen Permutation from Group



Implementation:

Note that every group can be written as a product of cyclic groups:

$$\Pi = \prod_{\pi \in \Pi} \langle \pi \rangle$$

Now, we can use the previous step. (Admittedly this is not efficient.)

Conclusion and Thanks!

Main take-away

Next time you shuffle in a game (fancier than full S_n), consider active security.




Not covered:

- Protocols with permutations encoding the players' inputs and their representation as certain kinds of “state trees” (where security can be verified by a local check at branching nodes and leaves).

Open Problems

- What are all the shuffles that have an actively-secure implementation
- What are all the shuffles that have a *nice and efficient* actively-secure implementation
- Impossibility results for protocols with permutation-encoded inputs.

Literature

-  Bert den Boer. “More Efficient Match-Making and Satisfiability: The Five Card Trick”. In: *EUROCRYPT 1989*. Ed. by Jean-Jacques Quisquater and Joos Vandewalle. Vol. 434. LNCS. Springer, 1989, pp. 208–217. ISBN: 3-540-53433-4. DOI: 10.1007/3-540-46885-4_23.
-  Takaaki Mizuki and Hideaki Sone. “Six-Card Secure AND and Four-Card Secure XOR”. In: *FAW 2009*. LNCS 5598. Springer, 2009, pp. 358–369. DOI: 10.1007/978-3-642-02270-8_36.
-  Tom Verhoeff. *The Zero-Knowledge Match Maker*. June 18, 2014. URL: <https://www.win.tue.nl/~wstomv/publications/liber-AMiCorum-arjeh-bijdrage-van-tom-verhoeff.pdf> (visited on 08/23/2019).