論文 / 著書情報
Article / Book Information

題目(和文)	実演に適したカードベースプロトコルの構成について
Title(English)	On the Construction of Easy to Perform Card-Based Protocols
著者(和文)	品川和雅
Author(English)	Kazumasa Shinagawa
出典(和文)	学位:博士(理学), 学位授与機関:東京工業大学, 報告番号:甲第11391号, 授与年月日:2020年3月26日, 学位の種別:課程博士, 審査員:渡辺 治,田中 圭介,伊東 利哉,尾形 わかは,西崎 真也,縫田 光司
Citation(English)	Degree:Doctor (Science), Conferring organization: Tokyo Institute of Technology, Report number:甲第11391号, Conferred date:2020/3/26, Degree Type:Course doctor, Examiner:,,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

On the Construction of Easy to Perform Card-Based Protocols

Kazumasa Shinagawa

A dissertation submitted in partial fulfillment
of the requirement for the degree of

Doctor of Science

Department of Mathematical and Computing Science
Tokyo Institute of Technology

February 2020

Abstract

Secure computation allows a set of players holding secret inputs to compute a joint function of the inputs without revealing the inputs. Although secure computation is usually done by the use of computers over a network, it can also be done by using a deck of physical cards. This research area is called *card-based cryptography*. In card-based protocols, input information is encoded by a sequence of face-down cards, which is called a *commitment*. Then, operations are applied in front of all players to the sequence of commitments produced by the players. These operations include a *permutation*, which rearranges the order of the sequence, *turning*, which opens/closes the symbol of a card by turning the card over, and *shuffle*, which applies a random permutation chosen from a probability distribution so as not to reveal which permutation is applied. The goal of card-based protocols is either to publicly reveal the output value or to produce a commitment to the output value without revealing the inputs. Protocols of the former type are called *non-committed format protocols*, while protocols of the latter type are called *committed format protocols*. Since the output commitment of a committed format protocol can be used as the input commitment of another protocol, to compute any Boolean circuit, it is sufficient to construct committed format protocols for the NOT, AND, and COPY functions. It should be noted that a commitment is usually designed so that the NOT computation is trivial. In addition, the COPY computation is necessary since protocols usually consume and destroy the input commitments.

In 1989, den Boer (Eurocrypt 1989) proposed the first card-based protocol called the *Five-Card Trick*, which is a non-committed format AND protocol that uses five cards and a *random cut*. A random cut is a shuffle operation that randomly and cyclically shifts the order of a sequence. In 1993, Crépeau and Killian (CRYPTO 1993) showed that every function $f : \{0,1\}^n \to \{0,1\}$ has a committed format protocol by constructing a 10-card AND protocol, a 12-card XOR protocol, and an eight-card COPY protocol in a committed format. Again, their protocols use random cuts only. Subsequently, these protocols were improved in terms of the number of cards. State-of-the-art protocols include a four-card AND protocol proposed by Koch, Walzer, and Härtel (Asiacrypt 2015) and a five-card COPY protocol proposed by Nishimura, Nishida, Hayashi, Mizuki, and Sone (TPNC 2015). Although these protocols achieve the minimum number of cards, they are not considered to be easy to perform since they use a heavy shuffle operation. However, using a heavy operation is sometimes unavoidable.

In fact, Koch (ePrint 2018) proved that any four-card AND protocol requires either a *non-uniform shuffle* or a *non-closed shuffle*, both of which are not easy to perform. We believe that it is important to consider not only from the number of cards but also from the aspect of being easy to perform.

In the first part of this dissertation (Chapter 3), we study card-based protocols using a small number of shuffles restricted to *uniform closed shuffles*, which are shuffles whose permutation set is closed and in which each permutation is chosen uniformly and randomly. To the best of our knowledge, there is no general protocol in the literature that uses a constant number of shuffles even with the use of non-uniform or non-closed shuffles. In this work, we construct a general protocol with a single uniform closed shuffle. This protocol achieves the minimum number of shuffles, as it is impossible to securely compute any non-trivial function without shuffles (i.e., with permutations and turnings only). In addition, the proposed protocol only requires $24q+2n$ cards, where n is the input length and q is the number of cards in a circuit computing the function. This is achieved by introducing a card-based variant of the *garbled circuit methodology*. We also construct a general protocol with two *pile-scramble shuffles*, which are one of the easiest to perform shuffles among uniform closed shuffles. This is accomplished by introducing a *batching technique*, which combines multiple pile-scramble shuffles in parallel into one pile-scramble shuffle using a relatively small number of additional cards.

In the second part of this dissertation (Chapter 4), we study card-based protocols using *private permutations* instead of shuffles. A private permutation is an operation that *covertly* applies a permutation chosen by a player according to the player's input bit. A private permutation is considered easier to perform than a shuffle since *every* private permutation can be easily physically performed while it is not known how to physically perform certain shuffles. However, since private permutations are necessarily performed at a physically isolated location so as not to reveal the chosen permutation, protecting against malicious attacks in private permutations is difficult. Thus, protocols with private permutations are considered easier to perform but less secure than protocols with shuffles. We solve this dilemma by defining a new security notion called *active security* that captures malicious attacks in private permutations. Furthermore, we construct several protocols with active security. In particular, for any function $f : \{0,1\}^n \to \{0,1\}$, we construct a general protocol with n private permutations, which has the minimum number of private permutations. We also construct a general protocol with $2n + 7$ cards, which is optimized in terms of the number of cards. In addition, we construct several protocols for concrete functions that are efficient with a small number of cards and private permutations.

In the third part of this dissertation (Chapter 5), we study card-based protocols based on *polygon-shaped cards*. Suppose that we wish to compute a function $f : (\mathbb{Z}/m\mathbb{Z})^n \to \mathbb{Z}/m\mathbb{Z}$ for some integer $m \in \mathbb{N}$. Although a protocol for any Boolean function implies a protocol for any $f : (\mathbb{Z}/m\mathbb{Z})^n \to \mathbb{Z}/m\mathbb{Z}$, this conversion usually increases the number of cards and shuffles by a factor of $O(\log m)$. To circumvent this inefficiency, we introduce two types of polygon-shaped cards

possessing a $(360/m)^\circ$ rotational symmetry: *cyclic cards* and *dihedral cards*. Based on cyclic cards, we construct efficient protocols for concrete functions such as addition, subtraction, copy, and multiplication. It is also possible to construct a protocol for any function $f : (\mathbb{Z}/m\mathbb{Z})^n \to \mathbb{Z}/m\mathbb{Z}$ based on our new technique, *oblivious conversion*; however, a large number of cards is required. Based on dihedral cards, we construct efficient protocols for interesting predicates such as a carry predicate, equality predicate, and greater than predicate. Because every protocol based on cyclic cards also work based on dihedral cards, by combining the addition protocol and the carry protocol, we can efficiently compute addition and subtraction over large integers.

Acknowledgements

I am deeply grateful to Professor Osamu Watanabe, whom I first met at the ELC Autumn School of Computational Complexity in 2015 when I was a student at the University of Tsukuba. When I began my doctoral course at the Tokyo Institute of Technology, he kindly welcomed me to Watanabe laboratory. During the doctoral course, he supported me as my supervisor and gave me many helpful comments on my research and school life.

I would like to thank Professor Keisuke Tanaka. He supported me as my sub-supervisor since the second year of my doctoral course and warmly welcomed me to Tanaka laboratory and the lab seminar.

I would also like to express my gratitude to Associate Professor Koji Nuida at the University of Tokyo. He gave me a lot of useful advice and many important comments as a member of a study group *Shin-Akarui Ango Benkyo-Kai* and co-author of some of my papers. Without his support, some results in this dissertation would not have been possible.

In addition to the aforementioned examiners of this dissertation, I would like to offer my special thanks to other examiners: Professor Toshiya Itoh, Professor Wakaha Ogata, and Professor Shin-ya Nishizaki.

I would like to express appreciation to Dr. Goichiro Hanaoka at the National Institute of Advanced Industrial Science and Technology. I have been part of *Shin-Akarui Ango Benkyo-Kai* that he organized since I was a third-year university student in 2014. Dr. Goichiro Hanaoka encouraged me to study cryptography and gave me many helpful comments on my research. He also gave me many opportunities to speak with many excellent researchers.

I owe a great debt to Associate Professor Takashi Nishide at the University of Tsukuba. He was my supervisor at the University of Tsukuba and supported me in starting my research on cryptography. I was very lucky to spend time at Nishide laboratory at the beginning of my research.

I would like to thank my family and colleagues for their support and encouragement. I thank all members of Shin-Akarui Ango Benkyo-Kai, all members and staff of Watanabe laboratory, all members and staff of Tanaka laboratory, and all members and staff of Nishide laboratory.

Contents

Abstract **i**

Acknowledgements **v**

1 Introduction **1**
1.1 Background . . . 1
1.1.1 Protocols with shuffles . . . 2
1.1.2 Protocols with private permutations . . . 6
1.1.3 Protocols based on non-standard cards . . . 7
1.2 Contribution . . . 9
1.3 Overview of techniques . . . 11
1.3.1 Overview of protocols with uniform closed shuffles . . . 11
1.3.2 Overview of active protocols with private permutations . 14
1.3.3 Overview of protocols based on polygon-shaped cards . . 16
1.4 Publication overview . . . 17

2 Preliminaries **19**
2.1 Basic notations . . . 19
2.2 Model of protocols . . . 19
2.2.1 Deck, sequence, and visible sequence . . . 19
2.2.2 Operation . . . 21
2.2.3 View . . . 22
2.2.4 Protocol . . . 23
2.2.5 Functionality . . . 26
2.2.6 Correctness . . . 27
2.2.7 Security . . . 28
2.2.8 Composition of protocols . . . 28
2.3 Terminologies . . . 30
2.3.1 Circuit . . . 30
2.3.2 Branching program . . . 31

3 Protocols with Uniform Closed Shuffles **33**
3.1 Notations . . . 33
3.2 Base protocol . . . 34

3.3 Protocol with a single uniform closed shuffle 41
3.4 Protocol with two pile-scramble shuffles 42
3.4.1 Extended pile-scramble shuffle 42
3.4.2 Batching multiple pile-scramble shuffles 43
3.4.3 Protocol with two extended pile-scramble shuffles 45
3.4.4 Protocol with two standard pile-scramble shuffles 46

4 Protocols with Private Permutations 47
4.1 Introducing private permutations 47
4.1.1 Private permutations 47
4.1.2 Notations 47
4.1.3 Existing protocols in our model 48
4.2 Active security 50
4.2.1 Executed permutation list 50
4.2.2 Active security 52
4.3 One-round protocol in the envelope model 53
4.4 Protocol compiler in the envelope model 54
4.4.1 Commit-and-prove technique 54
4.4.2 Protocol compiler 55
4.5 Card-efficient protocol in the envelope model 58
4.5.1 Passive protocol based on branching program 58
4.5.2 Active protocol using $2n + 7$ cards 59
4.6 Efficient protocols for specific functions 59
4.6.1 Passive protocol for symmetric functions 59
4.6.2 Active protocol for symmetric functions 60
4.6.3 Active protocol for the AND function 61
4.6.4 Active protocol for equality of two strings 62

5 Protocols Based on Polygon-Shaped Cards 63
5.1 Protocols based on cyclic cards 63
5.1.1 Cyclic cards 63
5.1.2 Operations for cyclic cards 64
5.1.3 Notations 66
5.1.4 Outline of protocols 66
5.1.5 Subtraction protocol 67
5.1.6 Addition protocol from a rotation shuffle 68
5.1.7 Addition protocol from a backward rotation shuffle 69
5.1.8 Constant multiplication protocol 71
5.1.9 Multiplication protocol 73
5.1.10 Oblivious conversion from a pile random cut 75
5.1.11 Oblivious conversion from a flower shuffle 78
5.1.12 General protocol from oblivious conversion 79
5.2 Protocols based on dihedral cards 81
5.2.1 Dihedral cards 81
5.2.2 Operations for dihedral cards 83
5.2.3 Notations 85

5.2.4 Outline of protocols 85
5.2.5 Initialization protocol 86
5.2.6 Addition protocol 87
5.2.7 Sign normalization protocol 88
5.2.8 Sign-to-value protocol 90
5.2.9 Carry protocol 91
5.2.10 Equality with zero protocol 92
5.2.11 Equality protocol 93
5.2.12 Greater-than protocol 95

6 Conclusion **97**

Chapter 1

Introduction

1.1 Background

Suppose that three players – Alice, Bob, and Carol – wish to know who is the richest person among them; however, everyone wishes to avoid revealing their amount of money. A cryptographic protocol called *secure multiparty computation* [12,63,64] can solve this problem. However, conventional secure multiparty computation protocols tend to be based on deep mathematics. Therefore, it is unlikely that all participants executing a given protocol will concretely understand its correctness and security.

Recreational cryptography provides a simpler and more convenient solution, achieving secure multiparty computation using *everyday objects*. An incomplete list of studies on recreational cryptography is as follows:

- Card-based cryptography [9, 10];
- PEZ protocols [1, 3];
- Protocols using a dial lock [30];
- Protocols using the 15 puzzle [30];
- Protocols using light and shadow [23];
- Visual secret sharing scheme [37].

The correctness and security of these protocols can be easily understood, as all participants can perform the protocols themselves and observe the actual execution process. Thus, the protocols are suitable for use as educational materials on cryptography for beginner students. Indeed, some universities including Cornell University [24], University of Waterloo [8], and Tohoku University [26] introduced card-based protocols in their cryptography courses.

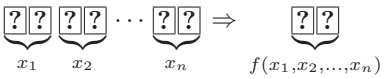
In this dissertation, we focus on card-based cryptography, which provides secure multiparty computation protocols using a deck of cards. The deck consists of a number of physical cards, where cards having the same symbol are

indistinguishable. The initial sequence is a line of face-down cards whose order potentially depends on the inputs. The goal of card-based protocols is to compute a function $f : X^n \to Y$ on n inputs $x_1, x_2, \ldots, x_n \in X$ without revealing the inputs. We introduce three models of protocols studied in previous works: *protocols with shuffles* (Section 1.1.1), *protocols with private permutations* (Section 1.1.2), and *protocols based on non-standard cards* (Section 1.1.3).

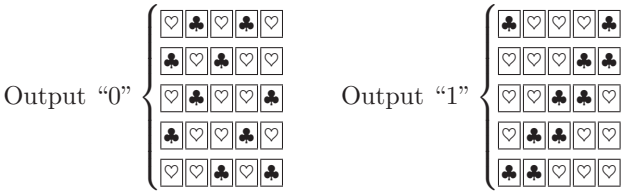
1.1.1 Protocols with shuffles

The protocols with shuffles is the most standard and popular model. The setting of this model is as follows. The computing function is a Boolean function $f : \{0,1\}^n \to \{0,1\}$, and the deck consists of a number of cards with two colors ♣♡ with the same back ? (hereafter we refer to this deck as a *deck of binary cards*). A single bit $x \in \{0,1\}$ is encoded by two face-down cards ?? whose front sides are ♣♡ if $x = 0$ and ♡♣ otherwise. This is called a *commitment to* x. During a protocol execution, three types of operations can be applied to a sequence of cards. The first type of operation is a *permutation*, which rearranges the order of the sequence. The second type of operation is a *turning*, which turns a card so that a face-down (resp. face-up) card changes to a face-up (resp. face-down) card. The third type of operation is a *shuffle*, which covertly applies a random permutation $\pi \in \Pi$ according to a probability distribution D over Π. We assume that the chosen permutation π is hidden from all players (i.e., no player knows which sequence is obtained). This is the most important property for achieving the input privacy. The goal of these protocols is to compute a function $f(x_1, x_2, \ldots, x_n)$ given n commitments to $x_1, x_2, \ldots, x_n$ and a number of additional cards (as a computational resource) without revealing the inputs.

There are two types of protocols: *committed format* and *non-committed format*. A protocol in a committed format produces a commitment to the output value $f(x_1, x_2, \ldots, x_n)$ as follows:



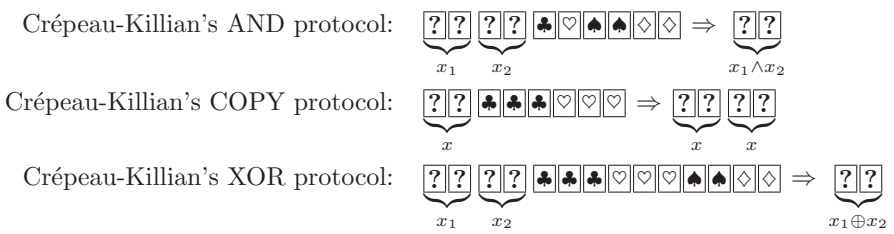
In contrast, a protocol in a non-committed format computes and reveals the value $f(x_1, x_2, \ldots, x_n)$ directly. For example, a five-card AND protocol in a non-committed format proposed by den Boer [10] opens all cards in the final step and determines the output value according to the opened symbols as follows:



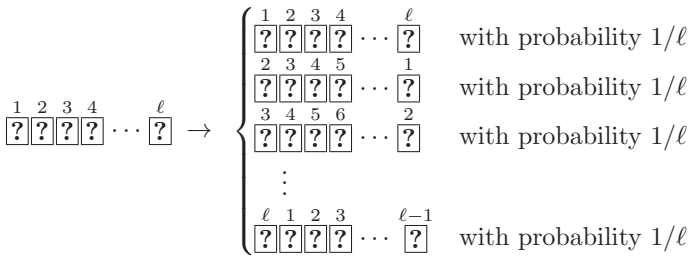
The advantage of protocols in a committed format is the composability of protocols; that is, the output commitment of a protocol can be used as the input commitment of another protocol. Since every Boolean function $f : \{0,1\}^n \to \{0,1\}$

can be computed by a circuit with NOT, AND, and COPY[1] gates, protocols for NOT, AND, and COPY in a committed format imply a protocol for f. Note that a NOT protocol is trivial: given a commitment to x, the protocol simply swaps the cards. Thus, to compute any function $f : \{0,1\}^n \to \{0,1\}$, it is sufficient to construct an AND protocol and a COPY protocol in a committed format.

Known results in a committed format. In 1993, Crépeau and Killian [9] showed that any function $f : \{0,1\}^n \to \{0,1\}$ can be computed by constructing a 10-card AND protocol[2], an eight-card COPY protocol, and a 14-card XOR protocol in a committed format as follows:



The required shuffles of these protocols are *random cuts* only. A random cut (for ℓ cards) is a shuffle that cyclically rotates the sequence as follows:



Formally, a random cut for ℓ cards is defined by a set of permutations Π and a probability distribution D, where $\Pi := \{\sigma^k \mid 0 \le k \le \ell - 1\}$ for a cyclic permutation[3] $\sigma = (1\ 2\ 3\ \cdots\ \ell)$ and D is a uniform distribution over Π. A random cut can be easily performed by hand or using a rotating table or roulette. Although these protocols are efficient in terms of the shuffle type (i.e., random cut), they are *Las-Vegas* protocols[4], whose expected number of steps is finite but does not terminate in the worst case. The desired notion, however, is *finite*

[1]Since a card-based protocol usually consumes the input commitments, the COPY gate, that takes a commitment to x and produces two copies of commitments to x, is required.

[2]The AND protocol and XOR protocol use a deck of cards having four colors ♣♡♠♢. All commitments for the input and output are of cards ♣♡ and the cards having two additional colors ♠♢ are used as helping cards.

[3]We denote a cyclic permutation by $(i_1\ i_2\ \cdots\ i_\ell) =: \pi$ which means $\pi(i_j) = i_{j+1}$ for $1 \le j \le \ell - 1$ and $\pi(i_\ell) = i_1$. For the other index $i' \notin \{i_1, i_2, \ldots, i_\ell\}$, $\pi(i') = i'$.

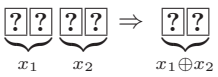
[4]The AND/XOR protocols are Las-Vegas but the COPY protocol is finite runtime.

runtime, which terminates at a finite number of steps. A number of improved AND/XOR protocols [34, 38, 59] have been proposed, all of which use random cuts only and are Las-Vegas protocols.

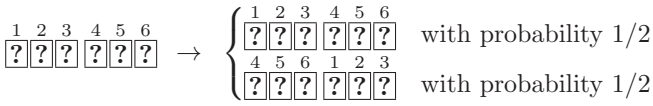
In 2009, Mizuki and Sone [33] proposed a six-card AND protocol, a six-card COPY protocol, and a four-card XOR protocol, which are finite runtime protocols, using a new shuffle called a *random bisection cut*, as follows:

Mizuki-Sone's AND protocol: 

Mizuki-Sone's COPY protocol: 

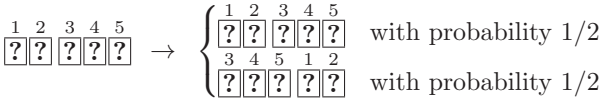
Mizuki-Sone's XOR protocol: 

A random bisection cut is a shuffle that divides the deck into two piles and randomly swaps them as follows (the following is the case of six cards):

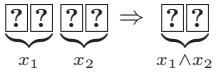


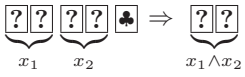
Ueda, Nishimura, Hayashi, Mizuki, and Sone [61] showed how to perform a random bisection cut. After the invention of random bisection cuts, a number protocols for "rich" functions based on random bisection cuts were proposed. For example, an eight-card half adder protocol [29], an eight-card protocol for a three-input majority function [42], an eight-card protocol for any three-input function $f : \{0,1\}^3 \to \{0,1\}$ [41], and a $2n+6$-card protocol for any function $f : \{0,1\}^n \to \{0,1\}$ [40] were proposed. All of these are finite runtime protocols while protocols with random cuts [9, 34, 38, 59] are Las-Vegas protocols. As a result of the invention of random bisection cuts, new shuffles have been introduced.

In 2013, Cheung, Hawthorne, and Lee [8] designed a five-card Las-Vegas AND protocol using a new shuffle called an *unequal division shuffle* as follows:



Based on this shuffle, Nishimura, Nishida, Hayashi, Mizuki, and Sone [44] designed a five-card Las-Vegas COPY protocol. In 2015, Koch, Walzer, and Härtel [21] designed a four-card Las-Vegas AND protocol and a five-card finite-runtime AND protocol using new shuffles as follows:

Koch-Walzer-Härtel's AND protocol #1: 

Koch-Walzer-Härtel's AND protocol #2: 

The four-card Las-Vegas AND protocol (#1) uses a shuffle whose probability distribution D is not uniform as follows:

$$\overset{1\ 2\ 3\ 4}{\boxed{?}\boxed{?}\boxed{?}\boxed{?}} \rightarrow \begin{cases} \overset{1\ 2\ 3\ 4}{\boxed{?}\boxed{?}\boxed{?}\boxed{?}} & \text{with probability } 1/3 \\ \overset{3\ 4\ 1\ 2}{\boxed{?}\boxed{?}\boxed{?}\boxed{?}} & \text{with probability } 2/3 \end{cases}$$

Such a shuffle is said to be *non-uniform*. The five-card finite runtime AND protocol (#2) uses a shuffle whose permutation set Π is not closed as follows:

$$\overset{1\ 2\ 3\ 4\ 5}{\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}} \rightarrow \begin{cases} \overset{1\ 2\ 3\ 4\ 5}{\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}} & \text{with probability } 1/3 \\ \overset{2\ 3\ 4\ 5\ 1}{\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}} & \text{with probability } 2/3 \end{cases}$$

Such a shuffle is said to be *non-closed*. The shuffle used in the AND protocol (#2) is in fact non-uniform and non-closed. Nishimura, Nishida, Hayashi, Mizuki, and Sone [43] showed how to perform (a class of) non-uniform/non-closed shuffles using special cases called *sliding covers*. However, these shuffles are considered to be not easy to perform, as sliding covers are not everyday objects. The authors [21] also proved that their protocols are optimal in terms of the number of cards by showing that there is no four-card finite runtime AND protocol. Inspired by these results, Abe, Hayashi, Mizuki, and Sone [2] designed a five-card Las-Vegas AND protocol using a random cut and random bisection cut. Kastner, Koch, Walzer, Miyahara, Hayashi, Mizuki, and Sone [18] and Koch [19] proved the optimality of AND/COPY protocols in various settings: the optimality of Abe et al.'s five-card AND protocol [2] in the case of Las-Vegas and uniform closed shuffles, the optimality of Mizuki-Sone's six-card AND protocol [33] in the case of finite runtime and uniform closed shuffles, Nishimura et al.'s five-card COPY protocol [44] in the case of Las-Vegas, and Mizuki-Sone's six-card COPY protocol [33] in the case of finite runtime.

Known results in a non-committed format. In 1989, den Boer [10] proposed a five-card AND protocol in a non-committed format that is known as the *Five-Card Trick* as follows:

$$\text{den Boer's AND protocol: } \underbrace{\boxed{?}\boxed{?}}_{\overline{x_1}}\ \boxed{?}_{\heartsuit}\ \underbrace{\boxed{?}\boxed{?}}_{x_2} \Rightarrow \begin{cases} \boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit} & \text{Output 0} \\ \boxed{\clubsuit}\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\clubsuit} & \text{Output 1} \end{cases}$$

We note that the output is 0 when the opened symbol is cyclically equivalent to $\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}$ and 1 otherwise. In 2006, Mizuki, Uchiike, and Sone [34] proposed a four-card XOR protocol and an eight-card protocol for the four-input XOR function. In 2012, Mizuki, Kumamoto, and Sone [31] improved the Five-Card Trick by constructing a four-card AND protocol using a random bisection cut. In 2014, Heather, Schneider, and Teague [16] proposed a six-card protocol for the three-input equality function; the same protocol was independently rediscovered by Shinagawa and Mizuki [53]. In 2016, Mizuki [27] proposed a $2n$-card protocol for the n-input AND function.

Discussion: obtaining easy to perform protocols. The efficiency of card-based protocols is measured primarily by the number of cards. However, as illustrated above, using the minimum number of cards does not imply an easy to perform protocol. For example, the four-card AND protocol in a committed format inherently requires either a non-uniform shuffle or a non-closed shuffle [19], both of which are not easy to perform. This shows that some card-efficient constructions inherently require kind of heavy operations which are hard to perform. As another example, a $2n + 6$-card protocol [40] for any function $f : \{0,1\}^n \to \{0,1\}$ requires $O(2^n)$ random bisection cuts. This shows that some card-efficient constructions require a large number of operations. To obtain easy to perform protocols, it is important to focus on a small number of easy to perform operations.

1.1.2 Protocols with private permutations

As in the model of protocols with shuffles, the model of protocols with *private permutations* also focuses on computing a Boolean function $f : \{0,1\}^n \to \{0,1\}$ using a deck of binary cards $\boxed{\clubsuit}\boxed{\heartsuit}$. The main feature of this model is the use of private permutations instead of shuffles; thus, three types of operations – permutations, turnings, and private permutations – can be applied. A private permutation is defined by a permutation π and an index $i \in \{1, 2, \ldots, n\}$ for the input length n, and it is performed by a player with the i-th input bit $x_i \in \{0,1\}$. The player covertly rearranges the order of the sequence according to π if $x_i = 1$ and does nothing otherwise. The other players cannot know which permutation (π or the identity permutation id) is applied. In actual execution, this is performed behind the player's back or under the table so that no players can see the private permutation process.

Known results with private permutations. In 2016, Nakai, Tokushige, Misawa, Iwamoto, and Ohta [36] proposed the model of private permutations. They constructed efficient protocols for the well-known millionaires' problem, which is the problem on determining the richest person among all players. Subsequently, Ono and Manabe [46] constructed an improved protocol for the millionaires' problem. Nakai, Shirouchi, Iwamoto, and Ohta [35] constructed a four-card protocol for a three-input voting function that takes three bits as input and outputs 1 if at least two of three input bits are 1 and 0 otherwise. Following this work, Watanabe, Kuroki, Suzuki, Koga, Iwamoto, and Ohta [62] improved the efficiency by constructing a three-card protocol for the three-input voting function. Although these protocols assume that each input is known by a player, there are several protocols with private permutations that do not hold this assumption. For example, Ono and Manabe [45, 47] designed several such protocols for fundamental functions. Instead of private permutations with input bits, their protocols use private permutations with random bits generated during protocol execution. In particular, the authors introduced a new operation, a *private random bisection cut*, in which a designated player chooses a random bit $r \in \{0,1\}$ and applies a private permutation according to r. Thus,

their protocols can be regarded as alternative implementations of the protocols proposed by Mizuki and Sone [33] using private permutations instead of random bisection cuts.

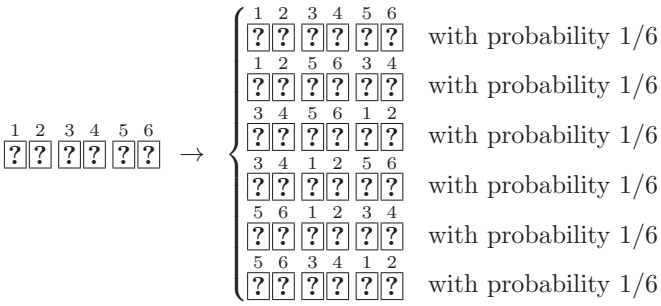
Discussion: protecting protocols against active attacks. Unlike shuffles, no matter how complex a permutation π is, a private permutation with π can be performed physically. (Recall that it is unknown how to physically perform certain shuffles.) However, this produces a new threat. Since a private permutation must be performed so that no player (except the i-th player) sees the operation process, the i-th player may perform malicious activities in his or her private permutations if he or she is malicious. Therefore, to obtain easy to perform and secure protocols based on private permutations, malicious behavior in private permutations must be addressed.

1.1.3 Protocols based on non-standard cards

Although the majority of card-based cryptography uses a deck of cards with two colors $\boxed{\clubsuit}\boxed{\heartsuit}$, there are several works that use non-standard cards. In the following, three types of cards – *number cards* (also known as *playing cards*), *cards with a rotationally symmetric back*, and *polarizing cards* – are introduced.

Known results based on number cards. In 1999, Niemi and Renvall [39] introduced a deck of *number cards* each with a unique color, which is denoted by $\boxed{1}\boxed{2}\boxed{3}\cdots\boxed{\ell}$ when the size of the deck is ℓ. The standard deck of playing cards can be regarded as a deck of 52 number cards. For any two cards $\boxed{i}$ and $\boxed{j}$ with $i < j$, a commitment to $x \in \{0, 1\}$ is defined by two face-down cards of $\boxed{i}\boxed{j}$ if $x = 0$ and $\boxed{j}\boxed{i}$ otherwise. Based on this encoding rule, the authors constructed a five-card Las-Vegas AND protocol, a four-card Las-Vegas XOR protocol, and a six-card Las-Vegas COPY protocol. Mizuki [28] improved Niemi and Renvall's result to finite runtime: an eight-card AND protocol, a four-card XOR protocol, and a six-card COPY protocol. Koch, Schrempp, and Kirsten [20] constructed a four-card Las-Vegas AND protocol and showed that there is no four-card AND protocol with finite runtime. Although these protocols compute fundamental functions, there are several works that compute other functions, such as computation over permutations. For example, Hashimoto, Nuida, Shinagawa, Inamura, and Hanaoka [14] used a deck of number cards to construct a protocol generating a random permutation with no fixed points. Such a permutation is known as *derrangement*, and the original protocol was proposed by Crépeau and Killian [9]. For this type of computation, a *pile-scramble shuffle*, which was introduced by Ishikawa, Chida, and Mizuki [17], is useful. This type of shuffle is a generalized version of a random bisection cut, as it divides $k\ell$ cards into k piles of ℓ cards and permutes them in a completely random manner as follows (the following is an example of the case $k = 3$ and

$\ell = 2$):



A deck of number cards with pile-scramble shuffles (and *pile-shifting shuffles*, a cyclic version of pile-scramble shuffles) makes it possible to construct new types of protocols: a grouping protocol that generates a random permutation with some conditions [15], zero-knowledge proofs for puzzles [6, 7, 11, 13, 22, 25, 48], and a ranking protocol [60]. Some of these protocols use a mixed deck of cards (e.g., ♣♡♣♡♣♡1 2 3 4 5 6). Shinagawa and Mizuki [54] showed an upper bound on the number of cards for computing any function $f : \{0,1\}^n \to \{0,1\}$ in the case of a mixed deck of cards by constructing a generalized version of Nishida et al. [40]'s general-purpose protocol.

Known results based on cards with a rotationally symmetric back. In 2014, Mizuki and Shizuya [32] introduced an entirely different type of card whose back side has a rotationally symmetric pattern (e.g., a white pattern such as) and whose front side has a non-rotationally symmetric pattern (e.g., a pattern with an arrow such as ↓). For a single bit, it is encoded as follows:

$$\boxed{\downarrow} = 0, \quad \boxed{\uparrow} = 1.$$

Using a deck of these cards, it is possible to compute any function using approximately half the number of cards that are required using a deck of binary cards ♣♡.

Known results based on polarizing cards. In 2015, Shinagawa, Mizuki, Schuldt, Nuida, Kanayama, Nishide, Hanaoka, and Okamoto [56] proposed a deck of polarizing cards, which are square-shaped polarizing plates. To encode a bit $x \in \{0,1\}$, two pairs of polarizing cards are encoded as 1 if a stack of them produces a black pattern (i.e., they have the opposite polarizing directions) and 0 otherwise (i.e., they have the same polarizing direction). Using polarizing cards, the authors constructed a four-card COPY protocol, a four-card XOR protocol, and a four-card AND protocol. These protocols have the minimum construction in terms of the number of cards.

Discussion: efficiently computing arithmetic circuits. In conventional secure multiparty computation, arithmetic circuits (i.e., circuits using addition and multiplication over mod m) are well studied, as in the case of Boolean circuits. However, in card-based cryptography, almost all protocols are focused on Boolean circuits, with several exceptions [6, 7, 11, 13, 15, 17, 22, 25, 48, 60]). Although a general protocol for Boolean circuits implies a protocol for arithmetic circuits, it produces inefficiency whereby the numbers of cards and shuffles are increased by a factor of $O(\log m)$. An important open question is how to design a new card for efficiently computing arithmetic circuits.

1.2 Contribution

In this dissertation, we study three models: protocols with shuffles, protocols with private permutations, and protocols with non-standard cards.

Protocols with uniform closed shuffles. In the first model, we study *protocols with uniform closed shuffles* (Chapter 3). In this model, three types of operations – permutations, turnings, and uniform closed shuffles – are allowed. Our contributions are as follows (see also Table 1.1):

- We construct a general protocol for any function $f : \{0,1\}^n \to \{0,1\}$ with a single uniform closed shuffle. The number of cards is $2n + 24q$ where q is the number of AND/COPY gates of a circuit computing f.
- We construct a general protocol with two (extended) pile-scramble shuffles. In the case of two extended pile-scramble shuffles, the number of cards is $2n + 24q + \Delta$. In the case of two standard pile-scramble shuffles, the number of cards is $2n + 24q + \Delta'$ such that $\Delta' > \Delta$. Δ and Δ' are defined in Sections 3.4.3 and 3.4.4.
- We propose a card-based variant of the *garbled circuit technique*, which computes any function with multiple pile-scramble shuffles. The first general protocol is obtained by combining all shuffles in the garbled circuit construction into a single shuffle.
- We propose a *batching technique* that combines multiple parallel pile-scramble shuffles into a single pile-scramble shuffle. The second general protocol is obtained by applying the batching technique to the garbled circuit construction.

Active protocols with private permutations. In the second model, we study *active protocols with private permutations* (Chapter 4). In this model, three types of operations – permutations, turnings, and private permutations – are allowed. We also assume that the initial sequence is a fixed sequence and that the result is provided as a single face-down card. Our contributions are as follows (see also Table 1.2):

Table 1.1: Protocols with uniform closed shuffles

	# of cards	# of uniform closed shuffles
◦ Any function $f : \{0,1\}^n \to \{0,1\}$ with q gates		
Section 3.3	$2n + 24q$	1 (uniform closed)
Section 3.4.3	$2n + 24q + \Delta$	2 (extended pile-scramble)
Section 3.4.4	$2n + 24q + \Delta'$	2 (pile-scramble)

Table 1.2: Protocols with private permutations

	security	# of cards	# of private permutations
◦ Any function $f : \{0,1\}^n \to \{0,1\}$ with depth d			
Section 4.3	active	2^n	n
Section 4.5.1	passive	5	$O(2^d)$
Section 4.5.2	active	$2n + 7$	$O(2^d + n)$
◦ Any symmetric function $f : \{0,1\}^n \to \{0,1\}$			
Section 4.6.1	passive	n	$n + 1$
Section 4.6.2	active	$2n$	$2n + 2$
◦ AND function for n inputs			
Section 4.6.3	active	$n + 1$	n
◦ Equality of two n-bit strings			
Section 4.6.4	active	$n + 1$	$2n$

- We define a new security notion, *active security*, for protocols with private permutations. Informally, a protocol is said to be actively secure if it does not leak any input information even when some players behave maliciously in private permutations. Hereafter, we refer to a protocol with active security as an *active protocol*.
- We construct an active protocol for any function $f : \{0,1\}^n \to \{0,1\}$ with n private permutations and 2^n cards. This is the minimum construction in terms of the number of private permutations.
- We construct an active protocol for any function $f : \{0,1\}^n \to \{0,1\}$ with $2n + 7$ cards and $O(2^n)$ private permutations.
- We construct active protocols for various concrete functions: a protocol for symmetric functions, a protocol for the AND function, and a protocol for equality of two strings.

Protocols based on polygon-shaped cards. In the third model, we study *protocols based on polygon-shaped cards* (Chapter 5). In particular, we design two types of polygon shaped cards, *cyclic cards* and *dihedral cards*. Our contributions are as follows (see also Tables 1.3 and 1.4):

Table 1.3: Protocols based on cyclic cards: n is the number of inputs; m is the modulus; R is a rotation shuffle; BR is a backward rotation shuffle; FLW is a flower shuffle; PRC is a pile random cut.

	# of cards	# of shuffles			
		R	BR	FLW	PRC
◦ Subtraction: $x_1 - x_2 \bmod m$					
Section 5.1.5	2	1	0	0	0
◦ Addition: $x_1 + x_2 \bmod m$					
Section 5.1.6	3	2	0	0	0
Section 5.1.7	2	0	1	0	0
◦ Constant multiplication: $(0, x, 2x, \cdots, (m-1)x)$					
Section 5.1.8	m	0	$2\lceil \log m \rceil$	0	0
◦ Multiplication: $x_1 x_2 \bmod m$					
Section 5.1.9	$m+1$	0	$2\lceil \log m \rceil$	1	0
◦ Oblivious conversion					
Section 5.1.10	$(k+1)m+1$	0	1	0	1
Section 5.1.11	$km+1$	0	0	1	0
◦ Any function: $f : (\mathbb{Z}_m)^n \to \mathbb{Z}_m$					
Section 5.1.12	$m^n + n$	0	0	n	0

- We design a new card called a cyclic card that can treat a multi-valued input $x \in \mathbb{Z}/m\mathbb{Z}$ naturally. Hereafter, we use $\mathbb{Z}_m$ to denote $\mathbb{Z}/m\mathbb{Z}$.
- Based on cyclic cards, we construct efficient protocols over $\mathbb{Z}_m$: a subtraction, addition, constant multiplication, multiplication, *oblivious conversion*, and general protocol.
- We design a new card called a dihedral card that is a variant of a cyclic card. As a cyclic card, it can treat a multi-valued input $x \in \mathbb{Z}_m$ naturally. Moreover, every protocol based on cyclic cards can be easily converted into a protocol based on dihedral cards.
- Based on dihedral cards, we construct efficient protocols over $\mathbb{Z}_m$: an initialization, addition, sign normalization, sign-to-value, carry, equality with zero, equality, and greater than protocol.

1.3 Overview of techniques

1.3.1 Overview of protocols with uniform closed shuffles

Rationale for uniform closed shuffles. A uniform closed shuffle is a shuffle whose permutation set Π is closed and the probability distribution D is a

Table 1.4: Protocols based on dihedral cards: n is the number of inputs; m is the modulus; R is a rotation shuffle; FLP is a flipping shuffle; TR is a two-sided rotation shuffle; $\mathsf{p}(\alpha)$ is a function that outputs 1 if α is true and 0 otherwise.

	# of cards	# of shuffles		
		R	FLP	TR
◦ Initialization				
Section 5.2.5	1	1	0	0
◦ Addition: $x_1 + x_2 \bmod 2m$				
Section 5.2.6	2	1	0	0
◦ Sign normalization: $x \bmod m$				
Section 5.2.7	1	0	0	1
◦ Sign-to-value: $\mathsf{p}(x \geq m)$				
Section 5.2.8	2	1	1	1
◦ Carry: $\mathsf{p}(x_1 + x_2 \geq m)$				
Section 5.2.9	2	2	1	1
◦ Equality with zero: $\mathsf{p}(x = 0)$				
Section 5.2.10	2	1	1	1
◦ Equality: $\mathsf{p}(x_1 = x_2)$				
Section 5.2.11	2	2	1	2
◦ Greater than: $\mathsf{p}(x_1 \geq x_2)$				
Section 5.2.12	2	2	1	1

uniform distribution over Π. This class of shuffles includes random cuts, random bisection cuts, and pile-scramble shuffles. Every uniform closed shuffle is considered easy to perform, as the effect of the shuffle can be easily produced using private permutations as follows. Suppose that there are n players in the execution of a protocol, and the protocol enters a uniform closed shuffle with Π and D. Then, for $1 \leq i \leq n$, the i-th player P_i uniformly and randomly chooses a permutation $\pi_i \in \Pi$ and covertly applies it to the sequence of cards. Due to the uniform closed property, the resulting permutation $\pi = \pi_n \cdots \pi_2 \pi_1$ is distributed uniformly and randomly among Π. Moreover, the above process ensures that no player knows which permutation is chosen unless at least one player is an honest player. Thus, any uniform closed shuffle is considered easy to perform. Among uniform closed shuffles, three types of shuffles – random cuts, random bisection cuts, and pile-scramble shuffles – are considered easier to perform than other uniform closed shuffles since they have fairly simple physical implementations: A random cut can be performed by a Hindu cut [61]; A random bisection cut can be performed using a separator card and rubber band, which is known as a *spinning throw* [61]; A pile scramble shuffle can be performed using envelopes [17].

Protocol with a single uniform closed shuffle. We construct a general protocol with a single uniform closed shuffle. This construction is based on a

card-based variant of the garbled circuit technique. Roughly speaking, the usual garbled circuit technique to securely evaluate a circuit proceeds as follows: (I) represent each gate in the circuit as the truth table of the associated function $\{0,1\}^2 \to \{0,1\}$; (II) randomly permute the four input-output pairs in the truth table, in order to prevent leakage of the output value when the gate is evaluated; (III) randomly encode each of the inputs and outputs in the truth table (in a consistent manner between the output of the previous gate and the corresponding input(s) of the subsequent gate(s)), in order to hide the input values; and (IV) then successively open one true output value among the four in the randomly encoded truth table of each gate, from the bottom to the top. The base protocol in Section 3.2 is a translation of the process described above into a card-based protocol, where the random permutations in (II) and the random encoding in (III) are realized by shuffle operations (the aforementioned consistency in (III) between the gates is ensured by the property of pile-scramble shuffles). See Section 3.2 for details. Based on the garbled circuit construction, we obtain a general-purpose protocol with one shuffle immediately (Section 3.3). This is achieved by aggregating all shuffles in the garbled circuit construction into one shuffle. This strategy is effective since all shuffles in the garbled circuit construction are successively applied.

Protocol with two pile-scramble shuffles. We construct a general protocol with two (extended) pile-scramble shuffles (Table 1.1). This construction is also a slightly modified version of the base protocol in Section 3.2. An important property of the card-based garbled circuit technique is compatibility with the parallel processing of shuffles. Namely, among the four steps in the garbled circuit technique, the random permutations (shuffles) for the truth tables in step (II) can be performed *in parallel* for all gates, and the random encoding (shuffles) of the truth tables in step (III) can also be performed *in parallel* for all input/output bits of the gates. The parallel executability of the shuffles combined with our batching technique achieves a protocol with two pile-scramble shuffles described in Section 3.4.

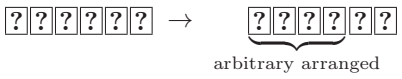
Batching technique. To explain the batching technique, here we provide an example of combining a pile-scramble shuffle of k piles and a pile-scramble shuffle of ℓ piles. The underlying idea is to first apply a pile-scramble shuffle to all $k + \ell$ piles and then divide the resulting piles into the first set of k piles and second set of ℓ piles. Now both the first k piles and second ℓ piles are individually shuffled uniformly at random whenever the shuffle for the whole of $k + \ell$ piles is uniformly random. However, this naive idea is not generally effective when the piles consist of face-down cards and the symbols on the front sides of the cards cannot be revealed. In fact, it is impossible in this case to detect the k piles in the first set among the $k + \ell$ shuffled piles. To overcome this issue, before performing the shuffle, we append several auxiliary face-down cards to the top of each pile, where the auxiliary cards for each of the first k piles (resp. second ℓ piles) encode information that the pile belongs to the first

(resp. second) set of piles. Then, even after the shuffle, by opening the auxiliary cards for each pile only, the piles in the two sets remain distinguishable from each other while the front sides of the original cards remain hidden.

1.3.2 Overview of active protocols with private permutations

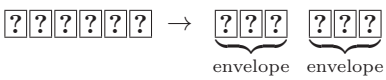
Definition of active attacks. In previous works on private permutations, all players are assumed to be semi-honest; that is, all players always follow protocol specifications. Thus, we must first define an *active attack*, that is, malicious behavior by a malicious player. It is reasonable to assume that in the execution of public operations such as permutations and turnings, it is impossible to act maliciously without detection by other players since all players can see the execution process at all times. Therefore, we assume that malicious actions only accor in private permutations. Suppose that a protocol enters a private permutation and the performing player is malicious. Since the other players cannot see the process of the private permutation, malicious players can do *arbitrary malicious actions*, including an *illegal opening*, in which they forcefully turn over cards and obtain input information, and an *illegal replacement*, in which they forcefully replace some cards with other cards that they themselves prepared. Clearly, even a weaker notion of security cannot be guaranteed unless malicious actions are restricted. Consequently, we first assume that in a private permutation, the least harmful malicious action is to apply a malicious permutation only. This assumption is reasonable when physical objects are available. For example, illegal openings can be prevented by the use of tamper-evident envelopes and illegal replacements can be prevented by the use of a physical unclonable function.

Restricting a set of possible permutations. Next we define the permutations that can be applied in a private permutation. Suppose that a protocol using six cards enters a private permutation $\pi = (1\ 2\ 3\ 4)$ with an index i. Then, the leftmost four cards are passed to the i-th player who performs the private permutation. If the player is malicious, it is reasonable to assume that he or she can apply an arbitrary permutation $\widetilde{\pi}$ with conditioning $\widetilde{\pi}(5) = 5$ and $\widetilde{\pi}(6) = 6$ as follows:



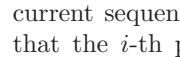
We refer to the model in which any permutation can be applied as a *bare-bone model*. However, the level of maliciousness in the bare-bone model is sometimes too high. Thus, we also define two alternative models – an *envelope model* and a *ring-with-envelope model* – both of which are justified by the use of physical objects. In the envelope model, a number of envelopes are used when a private permutation is executed. For example, if $\pi = (1\ 4)(2\ 5)(3\ 6)$, the first, second,

and third cards are putted into an envelope, while the fourth, fifth, and sixth cards are putted into another envelope as follows:



Then, the two envelopes are passed to a player. The player decides whether to swap them, and then, the envelopes are opened. In this case, due to the physical nature of the envelopes, even a malicious player cannot apply a permutation $\widetilde{\pi} \notin \{\mathsf{id}, \pi\}$. In the ring-with-envelope model, a ring hanging a number of envelopes is used when a private permutation is executed. As in the envelope model, this model restricts the set of possible permutations to a cyclic permutation of envelopes. For example, if $\pi = (1\ 3\ 5)(2\ 4\ 6)$, the first and second cards are putted into the first envelope, the third and fourth cards are putted into the second envelope, and the fifth and sixth cards are putted into the third envelope. Three envelopes are hung with a ring so as not to operate out of cyclic rotations. The ring is then passed to a player. The player applies a cyclic rotation to them, and then, the envelopes are opened as follows:

In this case, due to the physical nature of the envelopes, even a malicious player cannot apply a permutation $\widetilde{\pi} \notin \{\mathsf{id}, \pi, \pi^2\}$. Finally, we classify attacks into two types: an *out-of-range attack* and an *inconsistent attack*. Suppose that a protocol enters a private permutation with a permutation π and index i. An out-of-range attack applies a *possible* permutation $\widetilde{\pi} \notin \{\mathsf{id}, \pi\}$. An inconsistent attack applies a permutation $\pi' \in \{\mathsf{id}, \pi\}$ which is not inconsistent with a previous private permutation with index i; that is, the input bit x_i for choosing π' is not matched with that of a previous private permutation.

Active security. We introduce a new operation called an *abort operation* that terminates protocol execution if the current visible sequence does not match that of an honest execution. For example, suppose that a protocol enters an abort operation with a visible sequence $(?, ?, ?, ?, \clubsuit, \heartsuit)$ and an index i, and the current sequence is  . Then, the protocol terminates by claiming that the i-th player is malicious. We say that a protocol is actively secure whenever attack happens at a private permutation with an index i, a protocol immediately aborts with claiming that the i-th player is malicious.

Active protocol with n private permutations. We construct a protocol for any function $f : \{0,1\}^n \to \{0,1\}$ with n private permutations and 2^n cards. This protocol is actively secure in the envelope model. It is optimal in terms of the number of private permutations, as each input bit must be called by at least one private permutation to compute a non-trivial function. This construction is essentially based on a decision tree.

Active protocol with $2n+7$ cards. We construct a protocol for any function $f : \{0,1\}^n \to \{0,1\}$ with $2n+7$ cards and $O(2^d)$ private permutations, where d is the minimum depth of circuits computing f. This protocol is actively secure in the envelope model. We actually show that any protocol that is secure against semi-honest players can be converted into a protocol with active security by adding $2n+2$ cards. Based on Barrington's theorem, there is a semi-honest protocol with five cards for any function. Thus, we obtain a protocol with $2n+7$ cards by applying the compiler to the five-card protocol.

1.3.3 Overview of protocols based on polygon-shaped cards

Cyclic cards. Let $m \geq 2$ be any positive integer. A cyclic card for modulus m is a card having a back side with $(360/m)^\circ$ rotational symmetry and a front side with no rotational symmetry. For example, a square card with a front side $\boxed{\uparrow}$ and a back side $\boxed{}$ is a cyclic card for modulus $m = 4$ as follows:

$$\boxed{\uparrow} = 0, \quad \boxed{\leftarrow} = 1, \quad \boxed{\downarrow} = 2, \quad \boxed{\rightarrow} = 3.$$

For $x \in \mathbb{Z}_m$, we use $[\![x]\!]$ to denote a face-down card having the value x. We also use $\underline{x}$ to denote a face-up card having the value x. An important property is that every $[\![x]\!]$ has an identical face $\boxed{}$ on the back side. We show that several concrete functions $f : (\mathbb{Z}_m)^n \to \mathbb{Z}_m$ can be efficiently computed by using a deck of cyclic cards. For example, we construct an addition protocol using only two cards as follows:

$$\underbrace{\boxed{}}_{[\![x_1]\!]}\ \underbrace{\boxed{}}_{[\![x_2]\!]} \Rightarrow \underbrace{\boxed{\uparrow}}_{\underline{0}}\ \underbrace{\boxed{}}_{[\![x_1+x_2]\!]}.$$

Since the protocol only uses a single *backward rotation shuffle*, it has the minimum number of cards and shuffles.

Dihedral cards. Although a deck of cyclic cards makes it possible to compute several functions efficiently, it is incapable of efficiently computing some classes of functions. In particular, it has inefficiency to compute *predicates* that output either 0 or 1. An important predicate is the carry of addition, which outputs whether $x_1 + x_2 \geq m$. If a protocol computing the carry of addition is available, it is possible to add any large numbers as follows:

$$\underbrace{\boxed{}}_{[\![x_0]\!]}\ \underbrace{\boxed{}}_{[\![x_1]\!]} \cdots \underbrace{\boxed{}}_{[\![x_{k-1}]\!]}\ \underbrace{\boxed{}}_{[\![y_0]\!]}\ \underbrace{\boxed{}}_{[\![y_1]\!]} \cdots \underbrace{\boxed{}}_{[\![y_{k-1}]\!]} \Rightarrow \underbrace{\boxed{}}_{[\![z_0]\!]}\ \underbrace{\boxed{}}_{[\![z_1]\!]} \cdots \underbrace{\boxed{}}_{[\![z_{k-1}]\!]}\ \underbrace{\boxed{}}_{[\![z_k]\!]},$$

where $x, y \in \{1, 2, \ldots, m^k\}$, $z = x + y \in \{1, 2, \ldots, 2m^k\}$, and $x_i, y_i, z_i \in \mathbb{Z}_m$ is the i-th digit of x, y, z when the base is m. To circumvent the limitation of cyclic cards, we introduce *invisible ink* to the field of card-based cryptography. Text written in invisible ink is not visible but can becomes visible whenever illuminated by a black light. Using invisible ink, we design a new card called a *dihedral card*. A deck of dihedral cards is upward compatible with a deck of

cyclic cards in the sense that any protocol based on cyclic cards can be easily converted into the same protocol based on dihedral cards. In addition, a deck of dihedral cards makes it possible to compute several concrete predicates including a carry predicate $\mathsf{p}(x_1+x_2 \geq m)$, equality with zero predicate $\mathsf{p}(x = 0)$, equality predicate $\mathsf{p}(x_1 = x_2)$, and greater than predicate $\mathsf{p}(x_1 \geq x_2)$.

1.4 Publication overview

Chapter 3 is based on the paper as follows:

- Kazumasa Shinagawa, Koji Nuida. "A Single Shuffle Is Enough for Secure Card-Based Computation of Any Circuit," IACR Cryptology ePrint Archive, 2019/380 [58].

Chapter 4 is based on the papers as follows:

- Kazumasa Shinagawa. "Card-based Cryptographic Protocols Based on Private Transpositions," In 2018 Symposium on Cryptography and Information Security, SCIS 2018, Niigata, Japan, January 23–26, 2018, Proceedings, 2018 (In Japanese) [49].

- Kazumasa Shinagawa. "Deterministic Cryptographic Protocols with Active Security Using a Deck of Cards, Envelopes and Chains," In 2019 Symposium on Cryptography and Information Security, SCIS 2019, Shiga, Japan, January 22–25, 2018, Proceedings, 2019 (In Japanese) [50].

Chapter 5 is based on the papers as follows:

- Kazumasa Shinagawa, Takaaki Mizuki, Jacob C. N. Schuldt, Koji Nuida, Naoki Kanayama, Takashi Nishide, Goichiro Hanaoka, Eiji Okamoto. "Multi-party Computation with Small Shuffle Complexity Using Regular Polygon Cards," In Provable Security - 9th International Conference, ProvSec 2015, Kanazawa, Japan, November 24–26, 2015, Proceedings, pages 127–146, 2015 [55].

- Kazumasa Shinagawa, Takaaki Mizuki, Jacob C. N. Schuldt, Koji Nuida, Naoki Kanayama, Takashi Nishide, Goichiro Hanaoka, Eiji Okamoto. "Card-Based Protocols Using Regular Polygon Cards," IEICE Transaction on Fundamentals of Electronics, Communications and Computer Sciences, vol.100-A, no. 9, pp.1900–1909, 2017 [57].

- Kazumasa Shinagawa, Takaaki Mizuki. "Card-based Protocols Using Triangle Cards," In 9th International Conference on Fun with Algorithms, FUN 2018, June 13–15, 2018, La Maddalena, Italy, pages 31:1–31:13, 2018 [52].

- Kazumasa Shinagawa. "Card-based Cryptography with Invisible Ink," Theory and Applications of Models of Computation - 15th Annual Conference, TAMC 2019, Kitakyushu, Japan, April 13–16, 2019, Proceedings, volume 11436 of Lecture Notes in Computer Science, pages 566–577. Springer, 2019 [51].

Chapter 2

Preliminaries

2.1 Basic notations

Throughout this dissertation, we use the following notations: "$\mathbb{N}$" denotes the set of natural numbers, i.e., $\mathbb{N} = \{1, 2, 3, \ldots\}$. For any natural number $m \in \mathbb{N}$, "$[m]$" denotes the set $[m] = \{1, 2, 3, \ldots, m\}$. For any natural number $m \in \mathbb{N}$, "S_m" denotes the symmetric group of order m, i.e., S_m is the set of all permutations over m symbols. For any real number x, "$\lceil x \rceil$" denotes the smallest integer i such that $i \geq x$. "$\log x$" denotes the logarithm of x with the base two, i.e., $\log_2 x$. "$\emptyset$" denotes an empty set.

2.2 Model of protocols

2.2.1 Deck, sequence, and visible sequence

Deck. In Mizuki-Shizuya model, a *deck* is defined by a finite multiset. For example, $\mathcal{D} = \{\clubsuit, \clubsuit, \clubsuit, \heartsuit, \heartsuit, \heartsuit\}$ denotes a deck consists of six cards: three clubs and three hearts. All backsides are assumed to be "**?**". (Thus, it is required the condition that $\mathcal{D} \cap \{?\} = \emptyset$.) Although it captures some class of decks including decks of binary cards $\boxed{\clubsuit}\boxed{\heartsuit}$ and number cards $\boxed{1}\boxed{2}\boxed{3}$, it is not sufficient for our purpose since it does not capture a deck of "functional" cards. In particular, we introduce decks of such functional cards: *cyclic cards* and *dihedral cards*, both of which allow to change a symbol of a card by rotating or flipping the card.

In our model, we define a *deck* as follows:

Definition 2.1 (Deck)**.** A *deck* $\overline{\mathcal{D}}$ is defined by a five-tuple as follows:

$$\overline{\mathcal{D}} := (\mathcal{C}, \mathcal{T}, \Sigma, \mathsf{vis}, \mathcal{D}),$$

where $\mathcal{C}$ is a finite set called a *card set*, $\mathcal{T} \subset \{t \mid f : \mathcal{C} \to \mathcal{C}\}$ is called a *transformation set*, Σ is a finite set called a *symbol set*, $\mathsf{vis} : \mathcal{C} \to \Sigma$ is a function called a *vision function*, and $\mathcal{D}$ is a finite multiset called a *deck set*, where the

base set is $\mathcal{C}$. We assume that $\mathcal{T}$ always contains the identity function $\mathsf{id} : \mathcal{C} \to \mathcal{C}$. The former four-tuple $(\mathcal{C}, \mathcal{T}, \Sigma, \mathsf{vis})$ is called a *card specification*. ■

Example 1. Consider a deck of cards $\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\heartsuit}$ whose back sides are $\boxed{?}$, which is used by the Five-Card Trick [10]. The deck is described by the following:

- The card set is $\mathcal{C} = \{\clubsuit/?, \heartsuit/?, ?/\clubsuit, ?/\heartsuit\}$;
- The symbol set is $\Sigma = \{\clubsuit, \heartsuit, ?\}$;
- The transformation set is $\mathcal{T} = \{\mathsf{id}, \mathsf{turn}\}$, where the function turn is defined by $\mathsf{turn}(X/Y) = Y/X$ for any $X, Y \in \Sigma$;
- The vision function vis is defined by $\mathsf{vis}(X/Y) = X$ for any $X, Y \in \Sigma$;
- The deck set is $\mathcal{D} = \{\clubsuit/?, \clubsuit/?, \heartsuit/?, \heartsuit/?, \heartsuit/?\} = \{(\clubsuit/?)^2, (\heartsuit/?)^3\}$.

For the card set $\mathcal{C}$, the element "$\clubsuit/?$" (resp. "$\heartsuit/?$") means a face-up card $\boxed{\clubsuit}$ (resp. $\boxed{\heartsuit}$) and the element "$?/\clubsuit$" (resp. "$?/\heartsuit$") means a face-down card $\boxed{\clubsuit}$ whose front side is $\boxed{?}$ (resp. $\boxed{\heartsuit}$). The transformation set has a turning transformation turn. By applying turn to a card, a face-up card is changed to a face-down card (and vice versa). The vision function specifies what information is revealed from a card. From face-up cards "$\clubsuit/?$" and "$\heartsuit/?$", it reveals the symbols "$\clubsuit$" and "$\heartsuit$", on the other hand, from face-down cards "$?/\clubsuit$" and "$?/\heartsuit$", it reveals "$?$" only. This card specification $(\mathcal{C}, \mathcal{T}, \Sigma, \mathsf{vis})$ is called the *binary cards*. Hereafter, we denote the binary cards by $\mathsf{Binary} = (\mathcal{C}^{\mathsf{b}}, \mathcal{T}^{\mathsf{b}}, \Sigma^{\mathsf{b}}, \mathsf{vis}^{\mathsf{b}})$. We will use the binary cards in Chapters 3 and 4. ■

Sequence. We define a *sequence* as follows:

Definition 2.2 (Sequence)**.** Let $\overline{\mathcal{D}} = (\mathcal{C}, \mathcal{T}, \Sigma, \mathsf{vis}, \mathcal{D})$ be a deck. A *sequence* s *in* $\overline{\mathcal{D}}$ is defined as follows:

$$s = (t_1(x_1), t_2(x_2), \ldots, t_{|\mathcal{D}|}(x_{|\mathcal{D}|})),$$

where $t_1, t_2, \ldots, t_{|\mathcal{D}|} \in \mathcal{T}$ and $\mathcal{D} = \{x_1, x_2, \ldots, x_{|\mathcal{D}|}\}$ as a multiset. The set of all sequences in $\overline{\mathcal{D}}$ is denoted by $\mathsf{Seq}^{\overline{\mathcal{D}}}$. ■

Example 2. Let $\overline{\mathcal{D}} = (\mathsf{Binary}, \mathcal{D})$ be the deck in Example 1. An example of a sequence s of $\overline{\mathcal{D}}$ is as follows:

$$s = (?/\clubsuit, ?/\heartsuit, \heartsuit/?, ?/\heartsuit, ?/\clubsuit).$$

This is because s is represented as follows:

$$s = (\mathsf{turn}(\clubsuit/?), \mathsf{turn}(\heartsuit/?), \mathsf{id}(\heartsuit/?), \mathsf{turn}(\heartsuit/?), \mathsf{turn}(\clubsuit/?)).$$

It represents a sequence $\boxed{?}\boxed{?}\boxed{\heartsuit}\boxed{?}\boxed{?}$. ■

Visible sequence. We define a *visible sequence* as follows:

Definition 2.3 (Visible sequence)**.** Let $\overline{\mathcal{D}} = (\mathcal{C}, \mathcal{T}, \Sigma, \mathsf{vis}, \mathcal{D})$ be a deck and let $s = (x_1, x_2, \ldots, x_{|\mathcal{D}|}) \in \mathsf{Seq}^{\overline{\mathcal{D}}}$ be a sequence in $\overline{\mathcal{D}}$. A *visible sequence of s in $\overline{\mathcal{D}}$* is defined as follows:

$$\mathsf{vis}(s) := (\mathsf{vis}(x_1), \mathsf{vis}(x_2), \ldots, \mathsf{vis}(x_{|\mathcal{D}|})).$$

The set of all visible sequences in $\overline{\mathcal{D}}$ is defined as follows:

$$\mathsf{Vis}^{\overline{\mathcal{D}}} = \{\mathsf{vis}(s) \mid s \in \mathsf{Seq}^{\overline{\mathcal{D}}}\}.$$

■

Example 3. Let s be the sequence in Example 2. The visible sequence of s is $\mathsf{vis}(s) = (?, ?, \heartsuit, ?, ?)$. We sometimes write it by $(?^2, \heartsuit, ?^2)$ or $?^2\heartsuit?^2$. ■

2.2.2 Operation

Let $\overline{\mathcal{D}}$ be a deck. Let $s \in \mathsf{Seq}^{\overline{\mathcal{D}}}$ be a sequence in $\overline{\mathcal{D}}$. We consider two types of operations, *conversion* and *opening*, as follows:

- Conversion: It converts s into a new sequence $s' \in \mathsf{Seq}^{\overline{\mathcal{D}}}$. When it is deterministic, it is called a *deterministic operation* (e.g. permutation). When it is randomized, it is called a *probabilistic operation* (e.g. shuffle). When it depends on the input information, it is called an *input-dependent operation* (e.g. private permutation).

- Opening: It reveals some information on s (e.g. sign opening).

Now we define the most standard set of operations (of conversion) for binary cards. Let $\overline{\mathcal{D}} = (\mathsf{Binary}, \mathcal{D})$ be a deck of binary cards such that $|\mathcal{D}| = \ell$ and let $s = (c_1, c_2, \ldots, c_\ell) \in \mathsf{Seq}^{\overline{\mathcal{D}}}$ be a sequence in $\overline{\mathcal{D}}$. We define three sets of operations, *permutation*, *turning*, and *shuffle*, as follows:

Permutation. For $\pi \in S_\ell$, a *permutation operation* (perm, π) generates a new sequence in $\overline{\mathcal{D}}$ as follows:

$$(c_1, c_2, \ldots, c_\ell) \ \rightarrow \ (c_{\pi^{-1}(1)}, c_{\pi^{-1}(2)}, \ldots, c_{\pi^{-1}(\ell)}).$$

That is, the card in the i-th position in s is moved to the $\pi(i)$-th position in the new sequence. The set of permutations Perm_ℓ for sequences of ℓ cards is defined as follows:

$$\mathsf{Perm}_\ell := \{(\mathsf{perm}, \pi) \mid \pi \in S_\ell\}.$$

Turn. For a set of positions $T \subset [\ell]$, a *turning operation* (turn, T) takes s as input and returns a new sequence $s' \in \mathsf{Seq}^{\overline{\mathcal{D}}}$ as follows:

$$(c_1, c_2, \ldots, c_\ell) \ \to \ (c'_1, c'_2, \ldots, c'_\ell),$$

where for $i \in T$, it holds $c'_i = \mathsf{turn}(c_i)$, where this "$\mathsf{turn}$" is a transformation (i.e., $\mathsf{turn} \in \mathcal{T}^{\mathsf{b}}$), and for $i \notin T$, it holds $c'_i = c_i$. The set of turnings Turn_ℓ for sequences of ℓ cards is defined as follows:

$$\mathsf{Turn}_\ell := \{(\mathsf{turn}, T) \mid T \subset [\ell]\}.$$

Shuffle. A *shuffle operation* is defined by a tuple $(\mathsf{shuffle}, \Pi, D)$, where $\Pi \subset S_\ell$ is a subset of permutations and D is a probability distribution on Π. It randomly generates a new sequence $s' \in \mathsf{Seq}^{\mathcal{D}}$ as follows:

$$(c_1, c_2, \ldots, c_\ell) \ \to \ (c_{\pi^{-1}(1)}, c_{\pi^{-1}(2)}, \ldots, c_{\pi^{-1}(\ell)}),$$

where $\pi \in \Pi$ is independently and randomly chosen according to D. The set of shuffles Shuf_ℓ for sequences of ℓ cards is defined as follows:

$$\mathsf{Shuf}_\ell := \{(\mathsf{shuffle}, \Pi, D) \mid \Pi \subset S_\ell, D \text{ is a distribution on } \Pi\}.$$

A shuffle $(\mathsf{shuffle}, \Pi, D)$ is said to be

- *uniform* if D is a uniform distribution on Π;
- *closed* if Π is closed;
- *uniform closed* if it is uniform and closed.

The sets of uniform shuffles, closed shuffles, and uniform closed shuffles for sequences of ℓ cards are similarly defined and denoted by U_ℓ, C_ℓ, and UC_ℓ, respectively. Note that all of three are subsets of Shuf_ℓ and $\mathsf{UC}_\ell = \mathsf{U}_\ell \cap \mathsf{C}_\ell$.

2.2.3 View

Let $\overline{\mathcal{D}}$ be a deck. Let $\mathcal{O}$ be a set of operations. For a sequence $s \in \mathsf{Seq}^{\overline{\mathcal{D}}}$, an operation $\mathsf{op} \in \mathcal{O}$ converts it into a new sequence $s' \in \mathsf{Seq}^{\overline{\mathcal{D}}}$ with revealed information $r \in \{0,1\}^*$ as follows:

$$s \ \to \ s' \quad \text{revealed information } r,$$

where if op is conversion, revealed information is defined by $r = \bot$, and if op is opening, s' is equivalent to s. What is revealed from this process to the players? Before applying op, they observe a visible sequence $\mathsf{vis}(s)$. After applying op, they observe a visible sequence $\mathsf{vis}(s')$ and revealed information r. Thus, all information revealed from the above process is $(\mathsf{vis}(s), \mathsf{vis}(s'), r)$.

Suppose that a list of k operations $\vec{\mathsf{op}} \in \mathcal{O}^k$ is applied to a sequence s_0 as follows:

$$s_0 \ \to \ s_1 \ \to \ s_2 \ \to \ \cdots \ \to \ s_k.$$

Assume that the i-th operation brings revealed information $r_i \in \{0,1\}^*$. Then, all information revealed from the above process is given as follows:

$$(\mathsf{vis}(s_0), r_0) \to (\mathsf{vis}(s_1), r_1) \to (\mathsf{vis}(s_2), r_2) \to \cdots \to (\mathsf{vis}(s_k), r_k),$$

where $r_0 = \bot$ and $r_i = \bot$ if the i-th operation is conversion. This is called a *view of* $\vec{\mathsf{op}}$ *starting with the sequence* s_0. The set of views $\mathsf{View}^{\overline{\mathcal{D}}}$ is defined as follows:

$$\mathsf{View}^{\overline{\mathcal{D}}} = \left(\mathsf{Vis}^{\overline{\mathcal{D}}} \times \{0,1\}^*\right)^*.$$

Example 4. Let $\overline{\mathcal{D}} = (\mathsf{Binary}, \mathcal{D})$ be the deck in Example 1. Let $\mathcal{O}$ be a set of operations $\mathcal{O} = \mathsf{Perm}_5 \cup \mathsf{Turn}_5$. Let $\vec{\mathsf{op}}$ be a list of operations defined as follows:

$$\vec{\mathsf{op}} = \big((\mathsf{perm}, (1\ 2)), (\mathsf{turn}, \{1,2\}), (\mathsf{perm}, (1\ 3))\big).$$

When it is applied to a sequence $s_0 = (?/\clubsuit, ?/\heartsuit, ?/\clubsuit)$ as follows:

$$(?/\clubsuit, ?/\heartsuit, ?/\clubsuit) \ \to\ (?/\heartsuit, ?/\clubsuit, ?/\clubsuit) \ \to\ (\heartsuit/?, \clubsuit/?, ?/\clubsuit) \ \to\ (?/\clubsuit, \clubsuit/?, \heartsuit/?),$$

a view of $\vec{\mathsf{op}}$ starting with the sequence s_0 is given as follows:

$$((?,?,?), \bot) \ \to\ ((?,?,?), \bot) \ \to\ ((\heartsuit, \clubsuit, ?), \bot) \ \to\ ((?, \clubsuit, \heartsuit), \bot).$$

We sometimes omit revealed information it is clear that all operations are conversion as follows:

$$(?,?,?) \ \to\ (?,?,?) \ \to\ (\heartsuit, \clubsuit, ?) \ \to\ (?, \clubsuit, \heartsuit).$$

We also write the above by $?^3 \to ?^3 \to \heartsuit\clubsuit? \to ?\clubsuit\heartsuit$. ■

2.2.4 Protocol

Protocol. We define a *protocol* as follows:

Definition 2.4 (Protocol)**.** A protocol $\mathcal{P}$ is defined by a five-tuple as follows:

$$\mathcal{P} = (n, X, \overline{\mathcal{D}}, \mathcal{O}, A),$$

where

- $n \in \mathbb{N}$ is any natural number called the *number of inputs*;
- X is a finite set called an *input domain*;
- $\overline{\mathcal{D}} = (\mathcal{C}, \mathcal{T}, \Sigma, \mathsf{vis}, \mathcal{D})$ is a *deck*;
- $\mathcal{O}$ is a finite set called an *operation set*;
- $A : \mathsf{View}^{\overline{\mathcal{D}}} \to \mathcal{O} \cup \{\bot\}$ is an *action function*. ■

Execution of a protocol. Let $\mathcal{P} = (n, X, \overline{\mathcal{D}}, \mathcal{O}, A)$ be a protocol. Let $s_0 \in \mathsf{Seq}^{\overline{\mathcal{D}}}$ be a sequence. An execution of $\mathcal{P}$ starting with s_0 proceeds as follows:

1. The initial sequence is set to s_0 as follows:

$$s_0 = \boxed{?}\boxed{?}\boxed{?}\cdots\boxed{?}.$$

   Set $s \leftarrow s_0$ and $v \leftarrow (\mathsf{vis}(s_0), \bot)$, where s is a variable of the current sequence and v is a variable of the entire view of an execution.

2. Compute the action function $A(v) = \alpha$; if $\alpha \neq \bot$, apply the operation α to the sequence s; and obtain a new sequence s' with revealed information $r \in \{0,1\}^*$; Set $s \leftarrow s'$ and append "$\rightarrow (\mathsf{vis}(s'), r)$" to v; Repeat this step until it happens $\alpha = \bot$.

3. If $A(v) = \bot$, terminate the execution.

Example 5. We describe a (slightly modified version of) six-card AND protocol by Mizuki and Sone [33] as follows:

$$(2, \{0,1\}, \overline{\mathcal{D}}, \mathcal{O}, A).$$

The deck $\overline{\mathcal{D}}$ is defined by $\overline{\mathcal{D}} = (\mathsf{Binary}, \{(\clubsuit/?)^3, (\heartsuit/?)^3\})$. The operation set $\mathcal{O}$ is defined by $\mathcal{O} = \mathsf{Perm}_6 \cup \mathsf{Turn}_6 \cup \mathsf{Shuf}_6$. The action function A is defined by:

- $A(v_0) = (\mathsf{perm}, (2\ 4\ 3))$;
- $A(v_1) = (\mathsf{shuffle}, \Pi, D)$ where $\Pi = \{\mathsf{id}, (1\ 4)(2\ 5)(3\ 6)\}$ and D is a uniform distribution over Π;
- $A(v_2) = (\mathsf{perm}, (2\ 4\ 3)^{-1})$;
- $A(v_3) = (\mathsf{turn}, \{1,2\})$;
- $A(v_4) = (\mathsf{perm}, (1\ 2)(3\ 5)(4\ 6))$;
- $A(v) = \bot$ for any $v \notin \{v_0, v_1, v_2, v_3, v_4\}$.

where

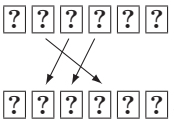
- $v_0 = (?^6, \bot)$;
- $v_1 = (?^6, \bot) \rightarrow (?^6, \bot)$;
- $v_2 = (?^6, \bot) \rightarrow (?^6, \bot) \rightarrow (?^6, \bot)$;
- $v_3 = (?^6, \bot) \rightarrow (?^6, \bot) \rightarrow (?^6, \bot) \rightarrow (?^6, \bot)$;
- $v_4 = (?^6, \bot) \rightarrow (?^6, \bot) \rightarrow (?^6, \bot) \rightarrow (?^6, \bot) \rightarrow (\heartsuit\clubsuit?^4, \bot)$.

We describe an execution of this protocol starting with an initial sequence $s_0 = (\mathsf{com}(x_1), \mathsf{com}(x_2), \mathsf{com}(1))$ as follows:

$$s_0 = \underbrace{?\,?}_{x_1}\ \underbrace{?\,?}_{x_2}\ \underbrace{?\,?}_{1},$$

where the commitment $\mathsf{com}(b)$ ($b \in \{0,1\}$) be two face-down cards whose front sides are ♣♡ if $b = 0$ and ♡♣ otherwise. The protocol proceeds as follows:

1. $(\mathsf{perm}, (2\ 4\ 3))$: Rearrange the order of the sequence as follows:

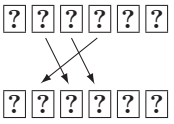


2. $(\mathsf{shuffle}, \Pi, D)$: Apply the shuffle:

$$\overset{1\ 2\ 3\ 4\ 5\ 6}{?\,?\,?\,?\,?\,?} \xrightarrow{(\mathsf{shuffle}, \Pi, U_\Pi)} \begin{cases} \overset{1\ 2\ 3\ 4\ 5\ 6}{?\,?\,?\,?\,?\,?} & \text{with probability } 1/2 \\ \overset{4\ 5\ 6\ 1\ 2\ 3}{?\,?\,?\,?\,?\,?} & \text{with probability } 1/2 \end{cases}$$

   This is a random bisection cut (See also Section 1.1.1).

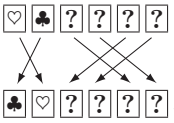
3. $(\mathsf{perm}, (2\ 4\ 3)^{-1})$: Rearrange the order of the sequence as follows:



4. $(\mathsf{turn}, \{1, 2\})$: Turn the leftmost commitment as follows:

   If it is the former case, i.e., the opened symbols are ♣♡, the protocol terminates. Otherwise, it proceeds to the next Step.

5. $(\mathsf{perm}, (1\ 2)(3\ 5)(4\ 6)\})$: Rearrange the order of the sequence as follows:



After Steps 4 and 5, the protocol terminates. Then, the finial sequence is given as follows:

$$♣♡\ \underbrace{?\,?}_{x_1 \wedge x_2}\ \underbrace{?\,?}_{\overline{x_1} \wedge x_2}.$$

Since it contains a commitment to $x_1 \wedge x_2$, it is said to be an AND protocol. ■

2.2.5 Functionality

In order to define the correctness and the security of protocols, we introduce a notion of *functionality*. Informally speaking, a functionality is a pair of sequences parametrized by input variables $\vec{x} \in X^n$. For example, the following is the functionality $\mathcal{F}_{\mathsf{AND}}$ of Mizuki-Sone's AND protocol (See Example 5).

$$\mathcal{F}_{\mathsf{AND}} : \underbrace{?\,?}_{x_1}\ \underbrace{?\,?}_{x_2}\ \underbrace{?\,?}_{1} \Rightarrow \clubsuit\heartsuit\ \underbrace{?\,?}_{x_1 \wedge x_2}\ \underbrace{?\,?}_{\overline{x_1} \wedge x_2}.$$

It is also described as follows:

$$\mathcal{F}_{\mathsf{AND}} : (\mathsf{com}(x_1), \mathsf{com}(x_2), \mathsf{com}(1)) \Rightarrow (\clubsuit\heartsuit, \mathsf{com}(x_1 \wedge x_2), \mathsf{com}(\overline{x_1} \wedge x_2)).$$

When some part of input/output sequences in a functionality are not important, $\perp$ is used. For example, when the AND protocol does not care about the rightmost commitment in the output sequence, it is described as follows:

$$\mathcal{F}'_{\mathsf{AND}} : (\mathsf{com}(x_1), \mathsf{com}(x_2), \mathsf{com}(1)) \Rightarrow (\clubsuit\heartsuit, \mathsf{com}(x_1 \wedge x_2), \perp^2).$$

Sequence with a dummy symbol. Let $\overline{\mathcal{D}} = (\mathcal{C}, \mathcal{T}, \Sigma, \mathsf{vis}, \mathcal{D})$ be a deck with $\mathcal{C} \cap \{\perp\} = \emptyset$, where $\perp$ is a *dummy symbol*. Let $s = (c_1, c_2, \ldots, c_\ell) \in \mathsf{Seq}^{\overline{\mathcal{D}}}$ be a sequence. A sequence $s' = (c'_1, c'_2, \ldots, c'_\ell) \in (\mathcal{C} \cup \{\perp\})^\ell$ is said to be a *dummy sequence of* s if $c'_i \in \{c_i, \perp\}$ for all $i \in [\ell]$. Thus, there exist 2^ℓ dummy sequences of any sequence of ℓ cards. The set of dummy sequences of s is denoted by $\mathsf{Seq}_\perp(s)$. The set of dummy sequences of $\overline{\mathcal{D}}$ is defined by

$$\mathsf{Seq}_\perp^{\overline{\mathcal{D}}} = \bigcup_{s \in \mathsf{Seq}^{\overline{\mathcal{D}}}} \mathsf{Seq}_\perp(s).$$

We say that $s \in \mathsf{Seq}^{\overline{\mathcal{D}}}$ is *matched* with $s' \in \mathsf{Seq}_\perp^{\overline{\mathcal{D}}}$ if $s' \in \mathsf{Seq}_\perp(s)$.

Example 6. For a sequence $s = (c_1, c_2, c_3)$, $\mathsf{Seq}_\perp(s)$ is given as follows:

$$\begin{aligned}\mathsf{Seq}_\perp(s) = \{&(c_1, c_2, c_3), (\perp, c_2, c_3), (c_1, \perp, c_3), (c_1, c_2, \perp),\\ &(\perp, \perp, c_3), (c_1, \perp, \perp), (\perp, c_2, \perp), (\perp, \perp, \perp)\}.\end{aligned}$$

For a sequence $s' = (c_1, c_2, c'_3)$ with $c'_3 \neq c_3$, s' is matched with $(c_1, c_2, \perp)$. ■

Variable sequence. Let $\overline{\mathcal{D}}$ be a deck, X be an input domain, and n be the number of inputs. A *variable sequence* s *over* $\mathsf{Seq}^{\overline{\mathcal{D}}}$ is defined by a function $s : X^n \to \mathsf{Seq}^{\overline{\mathcal{D}}}$. A *variable dummy sequence* s *over* $\mathsf{Seq}_\perp^{\overline{\mathcal{D}}}$ is defined by a function $s : X^n \to \mathsf{Seq}_\perp^{\overline{\mathcal{D}}}$.

Example 7. An input sequence $s(x)$ of Mizuki-Sone's AND protocol is a variable sequence $s : \{0,1\}^2 \to \mathsf{Seq}^{\overline{\mathcal{D}}}$ defined as follows:

$$s(x) = \begin{cases} (?/\clubsuit, ?/\heartsuit, ?/\clubsuit, ?/\heartsuit, ?/\clubsuit, ?/\heartsuit) & \text{if } x = (0,0) \\ (?/\clubsuit, ?/\heartsuit, ?/\heartsuit, ?/\clubsuit, ?/\clubsuit, ?/\heartsuit) & \text{if } x = (0,1) \\ (?/\heartsuit, ?/\clubsuit, ?/\clubsuit, ?/\heartsuit, ?/\clubsuit, ?/\heartsuit) & \text{if } x = (1,0) \\ (?/\heartsuit, ?/\clubsuit, ?/\heartsuit, ?/\clubsuit, ?/\clubsuit, ?/\heartsuit) & \text{otherwise.} \end{cases}$$

An output sequence $s'(x)$ of Mizuki-Sone's AND protocol is a variable dummy sequence $s' : \{0,1\}^2 \to \mathsf{Seq}^{\overline{\mathcal{D}}}_{\perp}$ defined as follows:

$$s'(x) = \begin{cases} (\clubsuit/?, \heartsuit/?, ?/\heartsuit, ?/\clubsuit, \perp^2) & \text{if } x = (1,1) \\ (\clubsuit/?, \heartsuit/?, ?/\clubsuit, ?/\heartsuit, \perp^2) & \text{otherwise.} \end{cases}$$

■

Functionality. A functionality is defined as follows:

Definition 2.5 (Functionality)**.** Let $\overline{\mathcal{D}}$ be a deck, X be an input domain, and n be the number of inputs. A *functionality* $\mathcal{F}$ is defined by a pair:

$$\mathcal{F} = (s_{\mathsf{in}}, s_{\mathsf{out}}),$$

where $s_{\mathsf{in}} : X^n \to \mathsf{Seq}^{\overline{\mathcal{D}}}$ is a variable sequence over $\mathsf{Seq}^{\overline{\mathcal{D}}}$ and $s_{\mathsf{out}} : X^n \to \mathsf{Seq}^{\overline{\mathcal{D}}}_{\perp}$ is a variable dummy sequence over $\mathsf{Seq}^{\overline{\mathcal{D}}}_{\perp}$. ■

2.2.6 Correctness

Correctness. The correctness of protocols is defined as follows:

Definition 2.6 (Correctness)**.** Let $\mathcal{P} = (n, X, \overline{\mathcal{D}}, \mathcal{O}, A)$ be a protocol. Let $\mathcal{F} = (s_{\mathsf{in}}, s_{\mathsf{out}})$ be a functionality. We say that $\mathcal{P}$ *correctly realizes* $\mathcal{F}$ if for any input $\vec{x} \in X^n$, any execution of $\mathcal{P}$ starting with $s_{\mathsf{in}}(\vec{x})$ terminates with a sequence s that is matched with $s_{\mathsf{out}}(\vec{x})$. ■

The correctness of protocols in a committed format is defined as follows:

Definition 2.7 (Correctness in a committed format)**.** Let $\mathcal{P} = (n, X, \overline{\mathcal{D}}, \mathcal{O}, A)$ be a protocol for $\overline{\mathcal{D}} = (\mathcal{C}, \mathcal{T}, \Sigma, \mathsf{vis}, \mathcal{D})$. Let $\mathcal{F} = (s_{\mathsf{in}}, s_{\mathsf{out}})$ be a functionality. Let $f : X^n \to X$ be a function. Let $\mathsf{com} : X \to \mathsf{Seq}^{\overline{\mathcal{D}}'}$ be a commitment, where $\overline{\mathcal{D}}' = (\mathcal{C}, \mathcal{T}, \Sigma, \mathsf{vis}, \mathcal{D}')$ such that $\mathcal{D}$ contains n copies of $\mathcal{D}'$. We say that $\mathcal{P}$ *correctly computes* f if it satisfies the following:

- $\mathcal{P}$ correctly realizes $\mathcal{F}$;
- $s_{\mathsf{in}} = (\mathsf{com}(x_1), \mathsf{com}(x_2), \cdots, \mathsf{com}(x_n), s)$ where s is a (possibly empty) fixed sequence;
- s_{out} contains $\mathsf{com}(f(x_1, x_2, \ldots, x_n))$ on a fixed position. ■

2.2.7 Security

The probability distribution of a view. Let $\mathcal{P} = (n, X, \overline{\mathcal{D}}, \mathcal{O}, A)$ be a protocol. Let $s_0 \in \mathsf{Seq}^{\overline{\mathcal{D}}}$ be a sequence and let $x \in X^n$ be an input. The *probability distribution of a view of $\mathcal{P}$ with input x and starting with sequence s_0* is denoted by $\mathsf{view}_{\mathcal{P}}(s_0, x)$, where randomness comes from probability operations (e.g., shuffles).

Security. The security of protocols is defined as follows:

Definition 2.8 (Security)**.** Let $\mathcal{P} = (n, X, \overline{\mathcal{D}}, \mathcal{O}, A)$ be a protocol. Let $\mathcal{F} = (s_{\mathsf{in}}, s_{\mathsf{out}})$ be a functionality. We say that $\mathcal{P}$ *securely realizes* $\mathcal{F}$ if for every $x, x' \in X^n$, it holds $\mathsf{view}_{\mathcal{P}}(s_{\mathsf{in}}(x), x) = \mathsf{view}_{\mathcal{P}}(s_{\mathsf{in}}(x'), x')$. ■

Example 8. Let us prove that the protocol given in Example 5 securely realizes the functionality $\mathcal{F}_{\mathsf{AND}} = (s_{\mathsf{in}}, s_{\mathsf{out}})$ defined as follows:

$$\mathcal{F}_{\mathsf{AND}} : (\mathsf{com}(x_1), \mathsf{com}(x_2), \mathsf{com}(1)) \Rightarrow (\clubsuit\heartsuit, \mathsf{com}(x_1 \wedge x_2), \mathsf{com}(\overline{x_1} \wedge x_2)).$$

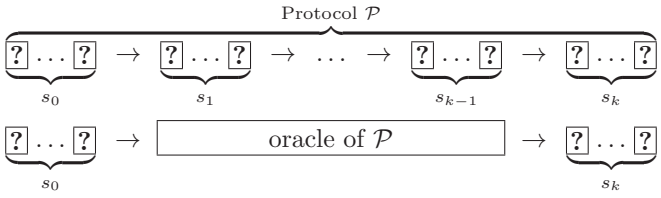
Let $x \in \{0,1\}^2$ be any input. The probability distribution of a view of the protocol starting with the sequence $s_{\mathsf{in}}(x) = (\mathsf{com}(x_1), \mathsf{com}(x_2), \mathsf{com}(1))$ is given as follows:

$$\mathsf{view}(s_{\mathsf{in}}(x), x) = \begin{cases} v \to (\clubsuit\heartsuit\mathbf{?}^4, \bot) & \text{with probability } 1/2 \\ v \to (\heartsuit\clubsuit\mathbf{?}^4, \bot) \to (\clubsuit\heartsuit\mathbf{?}^4, \bot) & \text{with probability } 1/2 \end{cases}$$

where $v = (\mathbf{?}^6, \bot) \to (\mathbf{?}^6, \bot) \to (\mathbf{?}^6, \bot) \to (\mathbf{?}^6, \bot)$. Due to the random bisection cut, the above probability distribution $\mathsf{view}(s_{\mathsf{in}}(x), x)$ is the same for any $x \in \{0,1\}^2$. Therefore, it securely realizes the functionality. ■

2.2.8 Composition of protocols

Oracle operation. Let $\mathcal{P} = (n, X, \overline{\mathcal{D}}, \mathcal{O}, A)$ be a protocol. An *oracle of $\mathcal{P}$* is a "magical box" that executes the protocol $\mathcal{P}$ in a single step: it takes a sequence $s_0 \in \mathsf{Seq}^{\overline{\mathcal{D}}}$ as an input and outputs a final sequence of $\mathcal{P}$ when the initial sequence is s_0 as follows:



Formally, an *oracle operation for a protocol $\mathcal{P}$* is defined as follows:

$$(\mathsf{oracle}, \mathcal{P}, T),$$

where $T \subset [\ell]$ is a subset of positions such that $|T|$ is the number of cards of $\mathcal{P}$. (We assume that the number of cards of $\mathcal{P}$ is equal to or less than ℓ.) The set of oracle operations with $\mathcal{P}$ is denoted as follows:

$$\mathsf{Oracle}_\ell[\mathcal{P}] = \{(\mathsf{oracle}, \mathcal{P}, T) \mid T \subset [\ell]\}.$$

For protocols $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k$, we define the set of oracle operations as follows:

$$\mathsf{Oracle}_\ell[\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k] = \mathsf{Oracle}_\ell[\mathcal{P}_1] \cup \mathsf{Oracle}_\ell[\mathcal{P}_2] \cup \cdots \cup \mathsf{Oracle}_\ell[\mathcal{P}_k].$$

We define an *oracle-respecting protocol* as follows:

Definition 2.9 (Oracle-respecting protocol)**.** Let $\mathcal{F}_{\mathsf{sub}} = (s_{\mathsf{in}}, s_{\mathsf{out}})$ be a functionality using ℓ_{sub} cards. Let $\mathcal{P}_{\mathsf{sub}} = (n_{\mathsf{sub}}, X_{\mathsf{sub}}, \overline{\mathcal{D}}_{\mathsf{sub}}, \mathcal{O}_{\mathsf{sub}}, A_{\mathsf{sub}})$ be a protocol using ℓ_{sub} cards. Let $\mathcal{P} = (n, X, \overline{\mathcal{D}}, \mathcal{O}, A)$ be a protocol using ℓ cards ($\ell \geq \ell_{\mathsf{sub}}$). We say that $\mathcal{P}$ is *oracle-respecting for* $\mathcal{P}_{\mathsf{sub}}$ *and* $\mathcal{F}_{\mathsf{sub}}$ if it satisfies as follows:

- $\mathsf{Oracle}_\ell[\mathcal{P}_{\mathsf{sub}}] \subset \mathcal{O}$;
- For any input $x \in \{0,1\}^n$, whenever $\mathcal{P}$ enters an operation $(\mathsf{oracle}, \mathcal{P}_{\mathsf{sub}}, T)$, the cards on positions T in the current sequence is always equivalent to $s_{\mathsf{in}}(x')$ for some input $x' \in X_{\mathsf{sub}}$. Here, the input x' for $\mathcal{P}_{\mathsf{sub}}$ can be varied for each call of the oracle for $\mathcal{P}_{\mathsf{sub}}$. ■

Example 9. Let $\mathcal{P}_{\mathsf{AND2}}$ be a two-bit AND protocol defined as follows:

$$\mathcal{P}_{\mathsf{AND2}} = (2, \{0,1\}, (\mathsf{Binary}, \{(\clubsuit/\mathbf{?})^3, (\heartsuit/\mathbf{?})^3\}), \mathsf{Perm}_6 \cup \mathsf{Turn}_6 \cup \mathsf{Shuf}_6, A),$$

that correctly and securely realizes a functionality $\mathcal{F}_{\mathsf{AND2}}$ as follows:

$$\mathcal{F}_{\mathsf{AND2}} : \underbrace{\boxed{?}\boxed{?}}_{x_1}\underbrace{\boxed{?}\boxed{?}}_{x_2}\underbrace{\boxed{?}\boxed{?}}_{1} \Rightarrow \underbrace{\boxed{?}\boxed{?}}_{\bot\ \bot}\underbrace{\boxed{?}\boxed{?}}_{x_1 \wedge x_2}\underbrace{\boxed{?}\boxed{?}}_{1}$$

This is obtained from Mizuki and Sone's AND protocol in Example 5 with a small modification. By using the oracle of $\mathcal{P}_{\mathsf{AND2}}$, we construct an eight-card three-bit AND protocol $\mathcal{P}_{\mathsf{AND3}}$ defined as follows:

$$\mathcal{P}_{\mathsf{AND3}} = (3, \{0,1\}, (\mathsf{Binary}, \{(\clubsuit/\mathbf{?})^4, (\heartsuit/\mathbf{?})^4\}), \mathsf{Oracle}_8[\mathcal{P}_{\mathsf{AND2}}], A').$$

that realizes a functionality $\mathcal{F}_{\mathsf{AND3}} = (s_{\mathsf{in}}, s_{\mathsf{out}})$ as follows:

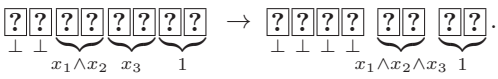
$$\mathcal{F}_{\mathsf{AND3}} : \underbrace{\boxed{?}\boxed{?}}_{x_1}\underbrace{\boxed{?}\boxed{?}}_{x_2}\underbrace{\boxed{?}\boxed{?}}_{x_3}\underbrace{\boxed{?}\boxed{?}}_{1} \Rightarrow \underbrace{\boxed{?}\boxed{?}\boxed{?}\boxed{?}}_{\bot\ \bot\ \bot\ \bot}\underbrace{\boxed{?}\boxed{?}}_{x_1 \wedge x_2 \wedge x_3}\underbrace{\boxed{?}\boxed{?}}_{1}.$$

It proceeds as follows:

1. $(\mathsf{oracle}, \mathcal{P}_{\mathsf{AND2}}, \{1,2,3,4,7,8\})$: Apply the two-bit AND protocol for cards on $\{1,2,3,4,7,8\}$ as follows:

$$\underbrace{\boxed{?}\boxed{?}}_{x_1}\underbrace{\boxed{?}\boxed{?}}_{x_2}\underbrace{\boxed{?}\boxed{?}}_{x_3}\underbrace{\boxed{?}\boxed{?}}_{1} \rightarrow \underbrace{\boxed{?}\boxed{?}}_{\bot\ \bot}\underbrace{\boxed{?}\boxed{?}}_{x_1 \wedge x_2}\underbrace{\boxed{?}\boxed{?}}_{x_3}\underbrace{\boxed{?}\boxed{?}}_{1}.$$

2. $(\mathsf{oracle}, \mathcal{P}_{\mathsf{AND2}}, \{3,4,5,6,7,8\})$: Apply the two-bit AND protocol for cards on $\{3,4,5,6,7,8\}$ as follows:



We can observe that the protocol $\mathcal{P}_{\mathsf{AND3}}$ is oracle-respecting for $\mathcal{P}_{\mathsf{AND2}}$ and $\mathcal{F}_{\mathsf{AND2}}$: the first condition in Definition 2.9 is satisfied since the operation set of $\mathcal{P}_{\mathsf{AND3}}$ is $\mathsf{Oracle}_8[\mathcal{P}_{\mathsf{AND2}}]$; and, the second condition in Definition 2.9 is satisfied since for each call of the oracle $\mathcal{P}_{\mathsf{AND2}}$, the cards on positions T in the sequence is equivalent to $s_{\mathsf{in}}(x')$ for some $x' \in \{0,1\}^2$. ■

Proposition 2.1 (Composition theorem)**.** Let $\mathcal{P}_i = (n_i, X_i, \overline{\mathcal{D}}_i, \mathcal{O}_i, A_i)$ $(i \in [k])$ be a protocol that correctly and securely realizes a functionality $\mathcal{F}_i$. Let $\mathcal{P} = (n, X, \overline{\mathcal{D}}, \mathcal{O} \cup \mathsf{Oracle}_\ell[\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_k], A)$ be a protocol that is oracle-respecting for $\mathcal{P}_i$ and $\mathcal{F}_i$, and $\mathcal{O}$ is upward compatible with $\mathcal{O}_i$ for every $i \in [k]$. If $\mathcal{P}$ correctly and securely realizes a functionality $\mathcal{F}$, then there exists a protocol $\mathcal{P}' = (n, X, \overline{\mathcal{D}}, \mathcal{O}, A)$ that correctly and securely realizes $\mathcal{F}$. ■

Proof. The protocol $\mathcal{P}'$ is obtained from the protocol $\mathcal{P}$ by replacing all oracle calls of $\mathcal{P}_i$ with the protocols $\mathcal{P}_i$ for all $i \in [k]$. We can observe that the final sequence of $\mathcal{P}$ and that of $\mathcal{P}'$ are the same since $\mathcal{P}$ is oracle-respecting. Thus, $\mathcal{P}'$ correctly realizes $\mathcal{F}$. We can also observe that a view of $\mathcal{P}'$ is obtained from a view of $\mathcal{P}$ by replacing all oracle calls of $\mathcal{P}_i$ with a view of $\mathcal{P}_i$ for all $i \in [k]$. Since $\mathcal{P}$ and $\mathcal{P}_i$ securely realize $\mathcal{F}$ and $\mathcal{F}_i$, respectively, for all $i \in [k]$. Thus, $\mathcal{P}'$ also securely realizes $\mathcal{F}$. ■

2.3 Terminologies

2.3.1 Circuit

In this paper, we use the following formulation for circuits given in [5]. A *circuit* C is defined as a six-tuple $C = (n, m, q, L, R, G)$. Here, $n \geq 1$ is the number of input bits, $m \geq 1$ is the number of output bits, $q \geq 1$ is the number of gates. We assume that each gate has two incoming wires and one outgoing wire, and an outgoing wire that is not an output wire of the protocol may then branch and go into several gates as the incoming wires. Accordingly, the outgoing wire of a gate and the corresponding incoming wire(s) of the subsequent gate(s) are identified with each other. We also allow a case where the two incoming wires of a gate come from the same previous gate, in order to realize by a gate a single-input function such as the NOT function.

Now we associate indices to the input bits, gates, wires, and the output bits as follows: $\mathsf{Inputs} = \{1, \ldots, n\}$, $\mathsf{Gates} = \{n+1, \ldots, n+q\}$, $\mathsf{Wires} = \{1, \ldots, n+q\}$, $\mathsf{Outputs} = \{n+q-m+1, \ldots, n+q\}$. Every wire $w \in \mathsf{Wires}$ is either an input wire or an outgoing wire of some gate, which has the same index as the wire itself. Then, $L, R : \mathsf{Gates} \to \mathsf{Wires} \setminus \mathsf{Outputs}$ are functions that map a gate to its

left (respectively, right) incoming wire. Moreover, for each $w \in \mathsf{Wires} \setminus \mathsf{Outputs}$, we write $L^{-1}(w), R^{-1}(w)$ to denote the set of the gates g satisfying $L(g) = w$ (respectively, $R(g) = w$). Finally, $G : \mathsf{Gates} \times \{0,1\}^2 \to \{0,1\}$ is a function that determines the functionality of each gate; given $g \in \mathsf{Gates}$ and $b_1, b_2 \in \{0,1\}$, we often write $G_g(b_1, b_2) = G(g, (b_1, b_2)) \in \{0,1\}$ to simplify the description. We require $L(g) \leq R(g) < g$ for all $g \in$ Gates.

Example 10. We consider a function $f : \{0,1\}^3 \to \{0,1\}^2$ given by $f(x_1, x_2, x_3) = ((x_1 \wedge x_2) \oplus x_3, (x_1 \wedge x_2) \vee x_3)$. A circuit for f can be defined by $n = 3$, $m = 2$, $q = 3$, $G_4(b_1, b_2) = b_1 \wedge b_2$, $G_5(b_1, b_2) = b_2 \oplus b_1$, $G_6(b_1, b_2) = b_2 \vee b_1$, $L(4) = 1$, $R(4) = 2$, $L(5) = 3$, $R(5) = 4$, $L(6) = 3$, and $R(6) = 4$.

2.3.2 Branching program

A *branching program* is a finite list of *instructions* which is defined by a three-tuple $\langle p, \pi_0, \pi_1 \rangle$ for an index $p \in [n]$ and two permutations $\pi_0, \pi_1 \in S_w$. The number of instructions is called the *length* and w is called the *width*.

Let B be a branching program of length ℓ and of width w, where the j-th instruction is $\langle p_j, \pi_0^{(j)}, \pi_1^{(j)} \rangle$ for $j \in [\ell]$. We define an *executed permutation of B with an input $x \in \{0,1\}^n$*, denoted by $\mathsf{ep}_{B,x}$, by the following permutation:

$$\mathsf{ep}_{B,x} = \pi^{(\ell)}_{x_{p_\ell}} \pi^{(\ell-1)}_{x_{p_{\ell-1}}} \cdots \pi^{(2)}_{x_{p_2}} \pi^{(1)}_{x_{p_1}}.$$

We say that B computes a function $f : \{0,1\}^n \to \{0,1\}$ if $\mathsf{ep}_{B,x}(1) = 1$ if and only if $f(x) = 1$.

Lemma 2.1 (Barrington's Theorem [4])**.** Let $f : \{0,1\}^n \to \{0,1\}$ be a function which can be computed by a depth-d circuit. Then, there exists a branching program of size 5 and length at most 4^d, which computes f.

Chapter 3

Protocols with Uniform Closed Shuffles

3.1 Notations

In this chapter, we use notations as follows.

Deck and commitment. We assume that a deck is $\overline{\mathcal{D}} = (\mathsf{Binary}, \mathcal{D})$, i.e., the card specification of $\overline{\mathcal{D}}$ is $\mathsf{Binary} = (\mathcal{C}^{\mathsf{b}}, \mathcal{T}^{\mathsf{b}}, \Sigma^{\mathsf{b}}, \mathsf{vis}^{\mathsf{b}})$ (see Example 1 in Section 2.2.1). Based on binary cards, we use a commitment com defined as follows:

$$\mathsf{com}(x) = \begin{cases} (?/\clubsuit, ?/\heartsuit) & \text{if } x = 0 \\ (?/\heartsuit, ?/\clubsuit) & \text{if } x = 1 \end{cases}$$

For a commitment to $x \in \{0, 1\}$, we use a graphical notation as follows:

$$\underbrace{\boxed{?}\boxed{?}}_{x}.$$

Operations. Let $\ell = |\mathcal{D}|$ be the number of cards. We assume that the set of operations is $\mathcal{O}^{\mathsf{b}}_{\ell}$ defined as follows:

$$\mathcal{O}^{\mathsf{b}}_{\ell} = \mathsf{Perm}_{\ell} \cup \mathsf{Turn}_{\ell} \cup \mathsf{UC}_{\ell}.$$

In other words, we assume that a protocol does not use a non-uniform and/or non-closed shuffle.

Pile-scramble shuffle. A *pile-scramble shuffle for ℓ' piles of k cards* is a uniform closed shuffle where ℓ' sets of k cards are rearranged according to a completely random permutation $\pi \in S_{\ell'}$. Let $T_1, T_2, \ldots, T_{\ell'} \subset [\ell]$ be disjoint sets with $T_i = \{t_{i,1}, t_{i,2}, \ldots, t_{i,k}\}$ for $t_{i,1} < t_{i,2} < \ldots < t_{i,k}$. For a permutation

$\pi \in S_{\ell'}$ over ℓ' words, a *pile permutation* $\mathsf{pile}_\ell[T_1, T_2, \ldots, T_{\ell'}; \pi] \in S_\ell$ is defined by a permutation over ℓ words as follows:

$$\mathsf{pile}_\ell[T_1, T_2, \ldots, T_{\ell'}; \pi](x) = \begin{cases} t_{\pi(i),j} & \text{if } x = t_{i,j} \text{ for some } i \in [\ell'], j \in [k] \\ x & \text{otherwise.} \end{cases}$$

Formally, the pile-scramble shuffle is denoted as $(\mathsf{shuffle}, \Pi, D)$ where D is a uniform distribution of Π and the permutation set Π is defined as follows:

$$\Pi = \{\mathsf{pile}_\ell[T_1, T_2, \ldots, T_{\ell'}; \pi] \mid \pi \in S_{\ell'}\}.$$

Hereafter, we use to denote it as follows:

$$(\mathsf{pileShuffle}, T_1, T_2, \ldots, T_k).$$

3.2 Base protocol

Preliminaries. Let $f : \{0,1\}^n \to \{0,1\}$ be a function and let $C = (n, 1, q, L, R, G)$ be a circuit computing f. For each gate $g \in \mathsf{Gates}$, let $t_g \in \{0,1\}^{12}$ be the string representing the truth table of g as follows:

$$t_g = \begin{array}{|c|c|c|} \hline 0 & 0 & G_g(0,0) \\ 0 & 1 & G_g(0,1) \\ 1 & 0 & G_g(1,0) \\ 1 & 1 & G_g(1,1) \\ \hline \end{array}$$

It can also be represented by a single-line bit string as follows:

$$t_g = (0, 0, G_g(0,0), 0, 1, G_g(0,1), 1, 0, G_g(1,0), 1, 1, G_g(1,1)).$$

We call the former by a *table expression of* g and the latter by a *string expression of* g. Using additional cards, a truth table t_g can be encoded as a sequence of face-down cards as follows:

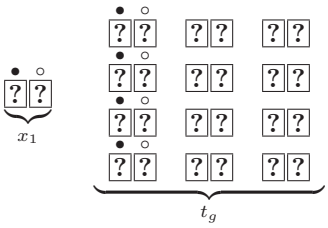
$$\underbrace{?\,?}_{0}\ \underbrace{?\,?}_{0}\ \underbrace{?\,?}_{G_g(0,0)}\ \underbrace{?\,?}_{0}\ \underbrace{?\,?}_{1}\ \underbrace{?\,?}_{G_g(0,1)}\ \underbrace{?\,?}_{1}\ \underbrace{?\,?}_{0}\ \underbrace{?\,?}_{G_g(1,0)}\ \underbrace{?\,?}_{1}\ \underbrace{?\,?}_{1}\ \underbrace{?\,?}_{G_g(1,1)}$$

This encoding of 24 cards is denoted by $\mathsf{com}(t_g)$. The initial sequence Γ^x for an input $x = (x_1, \cdots, x_n)$ is the concatenation of $\mathsf{com}(x_1), \cdots, \mathsf{com}(x_n)$ and $\mathsf{com}(t_{n+1}), \cdots, \mathsf{com}(t_{n+q})$ as follows:

$$\Gamma^x = \underbrace{?\,?}_{x_1}\ \underbrace{?\,?}_{x_2}\ \cdots\ \underbrace{?\,?}_{x_n}\ \overbrace{\underbrace{?\,?\ \cdots\ ?\,?}_{t_{n+1}}}^{24\text{ cards}}\ \cdots\ \overbrace{\underbrace{?\,?\ \cdots\ ?\,?}_{t_{n+q}}}^{24\text{ cards}}.$$

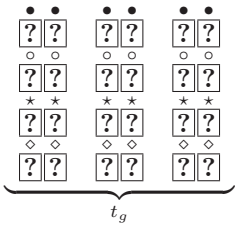
Thus, the number of cards in Γ^x is $2n + 24q$. We note that the commitments of the truth tables $\mathsf{com}(t_{n+1}), \cdots, \mathsf{com}(t_{n+q})$ can be put in front of all parties since all truth tables are publicly known.

In the base protocol, two types of shuffles are used. The first type of shuffles is random bisection cut, which is applied for each wire except the output wire. This results in two cases: nothing happens or all values associated to the wire are flipped. For example, suppose that a truth table of t_g satisfies $L(g) = 1$, i.e., the left input wire of t_g is connected to the input bit x_1, and there is no other gate $g' \in \mathsf{Gates} \setminus \{g\}$ such that g' is connected to x_1. Then, a random bisection cut is a random swapping between the $\bullet$-group and the $\circ$-group as follows:



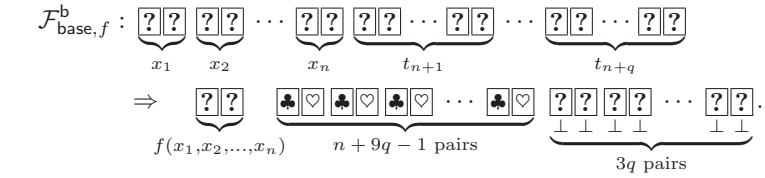
After the random bisection cut, all values associated to the wire are flipped with probability 1/2 and unchanged with probability 1/2. For any wire $w \in \mathsf{Wires}$, we denote by $P^{(w)}_{\mathsf{left}}$ (resp. $P^{(w)}_{\mathsf{right}}$) a subset of $\{1, 2, \cdots, 2n + 24q\}$ representing all positions of the left (resp. right) card associated to w. The random bisection cut in the above example is written by $(\mathsf{pileShuffle}, P^{(w)}_{\mathsf{left}}, P^{(w)}_{\mathsf{right}})$. (Note that a random bisection cut is a special case of pile-scramble shuffles.)

The second type of shuffles is pile-scramble shuffle, which is applied for each gate. This randomly permutes the order of four rows in the truth table. For example, for a truth table t_g, it is a random permutation among four rows, the $\bullet$-row, the $\circ$-row, the $\star$-row and the $\diamond$-row, as follows:



After the pile-scramble shuffle, one of 24 rearrangements is chosen uniformly and randomly. For any gates $g \in \mathsf{Gates}$, we denote by $P^{(g)}_i$ $(1 \le i \le 4)$ a subset of $\{1, 2, \cdots, 2n + 24q\}$ representing all positions of the i-th row in the truth table. The pile-scramble shuffle in the above example is written by $(\mathsf{pileShuffle}, P^{(g)}_1, P^{(g)}_2, P^{(g)}_3, P^{(g)}_4)$.

Functionality. A functionality $\mathcal{F}^{\mathsf{b}}_{\mathsf{base},f}$ is defined as follows:



Note that each t_i $(n+1 \le i \le n+q)$ consists of 12 pairs of $\boxed{\clubsuit}\boxed{\heartsuit}$. Thus, the number of cards is $2n + 24q$.

Protocol. The base protocol $\mathcal{P}^{\mathsf{b}}_{\mathsf{base},f}$ is defined as follows:

$$\mathcal{P}^{\mathsf{b}}_{\mathsf{base},f} = (n, \{0,1\}, (\mathsf{Binary}, \{\clubsuit^{\ell/2}, \heartsuit^{\ell/2}\}), \mathcal{O}^{\mathsf{b}}_{\ell}, A),$$

where $\ell = 2n + 24q$. It consists of the *garbling stage* and the *evaluation stage*. It proceeds as follows.

Garbling: Given a sequence $s_0 \in \mathsf{Seq}^{\overline{\mathcal{D}}}$, it proceeds as follows:

1. For every $w \in \mathsf{Wires} \setminus \mathsf{Outputs}$, apply $(\mathsf{pileShuffle}, P^{(w)}_{\mathsf{left}}, P^{(w)}_{\mathsf{right}})$.
2. For every $g \in \mathsf{Gates}$, apply $(\mathsf{pileShuffle}, P^{(g)}_1, P^{(g)}_2, P^{(g)}_3, P^{(g)}_4)$.

Evaluation: Given a final sequence of the garbling stage, it proceeds as follows:

1. For every $i \in [n]$, apply $(\mathsf{turn}, \{2i-1, 2i\})$, i.e., open the i-th input commitment. Let $x'_i \in \{0,1\}$ be the opened value.
2. For every gate $g \in \mathsf{Gates}$ (in order from $n+1$ to $n+q$), apply the following:
   (a) Open the leftmost and center commitments in the table expression of g, i.e., open eight commitments associated to the input wires of g. Let $l_i \in \{0,1\}$ (resp. $r_i \in \{0,1\}$) be the i-th row in the leftmost (resp. center) column.
   (b) Let $x'_{L(g)}, x'_{R(g)}$ be values associated to the left and right input wires, supposed to be defined in previous steps. Let $k_g \in \{1,2,3,4\}$ be an index such that $(l_{k_g}, r_{k_g}) = (x'_{L(g)}, x'_{R(g)})$.
   (c) If it is not the output gate, i.e., $g \neq n+q$, open the commitment in the k_g-th row of the rightmost column. Let $x'_g \in \{0,1\}$ be the opened value.
3. The output commitment is the commitment in the k_{n+q}-th row of the rightmost column in the output gate. Rearrange the order of the sequence so that the output commitment is moved to the leftmost, all opened cards are moved to the next leftmost in order $\boxed{\clubsuit}\boxed{\heartsuit}$, and other cards are moved to the rightmost.

Proof of correctness. We show the correctness of the base protocol $\mathcal{P}^{\mathsf{b}}_{\mathsf{base},f}$. In Step 1 of the garbling stage, for each wire $w \in \mathsf{Wires} \setminus \mathsf{Outputs}$, a pile-scramble shuffle is applied over two positions $P^{(w)}_{\mathsf{left}}$ and $P^{(w)}_{\mathsf{right}}$. Recall that the position $P^{(w)}_{\mathsf{left}}$ (resp. $P^{(w)}_{\mathsf{right}}$) designates the first (resp. second) cards of the commitments corresponding to the wire w. Thus, it is equivalent to masking the values of the commitments by an independently and uniformly random value $r_w \in \{0,1\}$. Therefore, after applying it, each row $(a, b, G_g(a,b))$ corresponding to the gate g turns into $(a \oplus r_{L(g)}, b \oplus r_{R(g)}, G_g(a,b) \oplus r_g)$. Since all values associated with the wire w are masked by the same random value r_w, it preserves the functionality of the truth table. In Step 2 of the garbling stage, for each gate $g \in \mathsf{Gates}$, a pile-scramble shuffle is applied over the four sets of positions $P^{(g)}_1, P^{(g)}_2, P^{(g)}_3$, and $P^{(g)}_4$. It just permutes the four rows in the truth table t_g. Thus, it also preserves the functionality of the truth table. Therefore, the output commitment is surely a commitment to the output value $f(x)$. We conclude that the base protocol $\mathcal{P}^{\mathsf{b}}_{\mathsf{base},f}$ correctly realizes the functionality $\mathcal{F}^{\mathsf{b}}_{\mathsf{base},f}$. ■

Proof of security. Let $\mathcal{P} = \mathcal{P}^{\mathsf{b}}_{\mathsf{base},f}$. Let $h = |\mathsf{Wires} \setminus \mathsf{Outputs}|$ and $q = |\mathsf{Gates}|$. Recall that the former h shuffles in Step 1 of the garbling stage are random bisection cuts and the latter q shuffles in Step 2 of the garbling stage are pile-scramble shuffles of four piles. Since a permutation in a random bisection cut is chosen by a random bit $r \in \{0,1\}$ and a permutation in a pile-scramble shuffle of four piles is chosen by a random permutation $\pi \in S_4$, a view of $\mathcal{P}$ starting with Γ_x is completely determined by $r_1, r_2, \ldots, r_h \in \{0,1\}$ and $\pi_1, \pi_2, \ldots, \pi_q \in S_4$. We denote it as follows:

$$v_{\mathcal{P}}(\Gamma_x; r_1, r_2, \ldots, r_h, \pi_1, \pi_2, \ldots, \pi_q) \in \mathsf{View}^{\overline{\mathcal{P}}}.$$

Define the set of all *possible* views $V_{\mathcal{P}} \subset \mathsf{View}^{\overline{\mathcal{P}}}$ as follows:

$$V_{\mathcal{P}} = \{v_{\mathcal{P}}(\Gamma_x; r_1, r_2, \ldots, r_h, \pi_1, \pi_2, \ldots, \pi_q) \mid x \in \{0,1\}^n, r_i \in \{0,1\}, \pi_j \in S_4\}.$$

We claim that for any $v \in V_{\mathcal{P}}$ and any input $x^* \in \{0,1\}^n$, there exist *unique* bits r^*_i $(1 \le i \le h)$ and *unique* permutations $\pi^*_i \in S_4$ $(1 \le i \le q)$ such that

$$v = v_{\mathcal{P}}(\Gamma_{x^*}; r^*_1, r^*_2, \ldots, r^*_h, \pi^*_1, \pi^*_2, \ldots, \pi^*_q).$$

Given $v = v_{\mathcal{P}}(\Gamma_x; r_1, \ldots, r_h, \pi_1, \ldots, \pi_q) \in V_{\mathcal{P}}$ and $x^* \in \{0,1\}^n$, the bits $r^*_i \in \{0,1\}$ and the permutations $\pi^*_i \in S_4$ are uniquely determined as follows:

1. We first define $r^*_i \in \{0,1\}$ $(1 \le i \le h)$. Since the input $x^* \in \{0,1\}^n$ is fixed, each value of the wire $i \in \mathsf{Wires}$ is also fixed. The bit r^*_i is set to the value of the wire i.

2. We next define $\pi^*_i \in S_4$ $(1 \le i \le q)$. Let $g = i + n \in \mathsf{Gates}$ be the gate and let $a, b \in \mathsf{Wires}$ be the left and right input wires of the gate i. Let $\tau_a, \tau_b \in S_4$ be the permutations defined as $\tau_a = (1\ 3)(2\ 4)$ and $\tau_b = (1\ 2)(3\ 4)$. We can observe that applying τ_a (resp. τ_b) to the initial

truth table $(00, 01, 10, 11)$ is equivalent to the bit flipping of the left (reps. right) row. For example, $\tau_a(00, 01, 10, 11) = (10, 11, 00, 01)$ is equivalent to the bit flipping of the left row. Thus, it is necessary to satisfy an equation of permutations as follows:

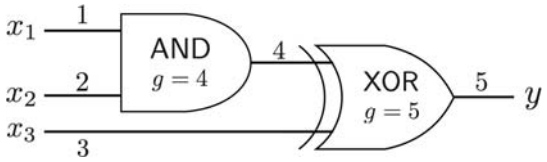
$$\pi_i^*(\tau_b)^{r_b^*}(\tau_a)^{r_a^*} = \pi_i(\tau_b)^{r_b}(\tau_a)^{r_a}.$$

We have to set the permutation π_i^* as follows:

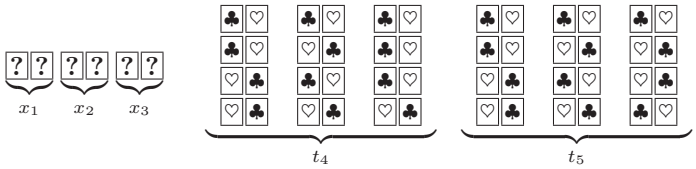
$$\pi_i^* = \pi_i(\tau_b)^{r_b \oplus r_b^*}(\tau_a)^{r_a \oplus r_a^*}.$$

Thus, for any input $x \in \{0,1\}^n$, the probability distribution $\mathsf{view}_{\mathcal{P}}(\Gamma_x, x)$ is a uniform distribution of $V_{\mathcal{P}}$. This implies $\mathsf{view}_{\mathcal{P}}(\Gamma_x, x) = \mathsf{view}_{\mathcal{P}}(\Gamma_{x'}, x')$ for any $x, x' \in \{0,1\}^n$. Therefore, $\mathcal{P}$ securely realizes $\mathcal{F}_{\mathsf{base},f}^{\mathsf{b}}$. ■

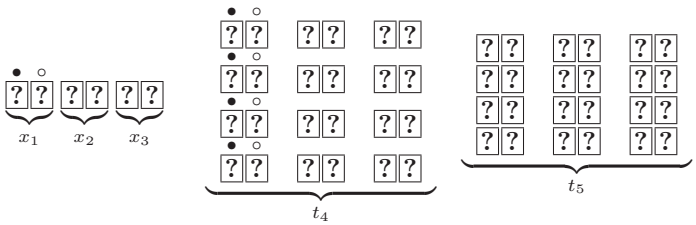
Example 11. We give an example of a protocol execution for the following circuit.



The initial sequence Γ^x is arranged as follows:

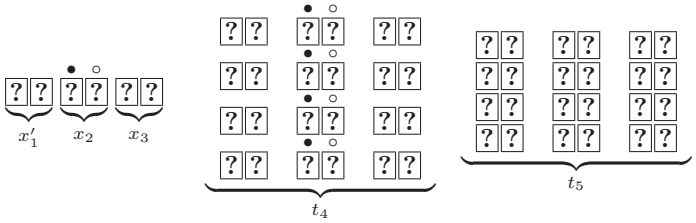


In Step 1 of the garbling stage, a pile-scramble shuffle is applied for each wire w except the output wire. For $w = 1$, a random bisection cut is applied for the •-group and the ∘-group as follows:

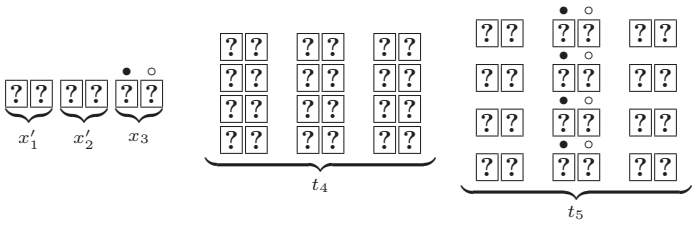


For $w = 2$, a random bisection cut is applied for the •-group and the ∘-group

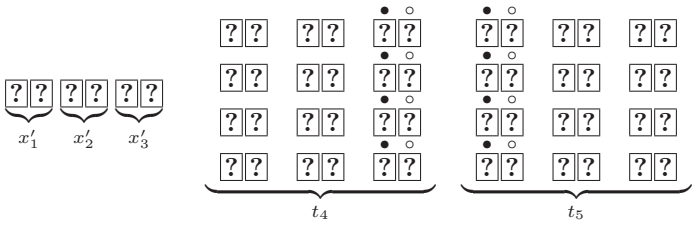
as follows:



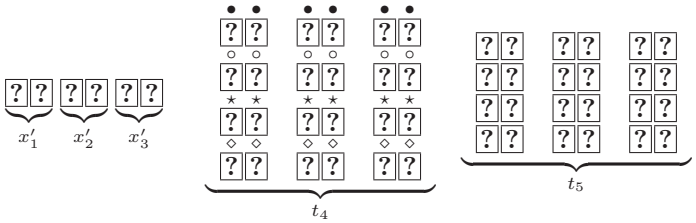
For $w = 3$, a random bisection cut is applied for the $\bullet$-group and the $\circ$-group as follows:



For $w = 4$, a random bisection cut is applied for the $\bullet$-group and the $\circ$-group as follows:

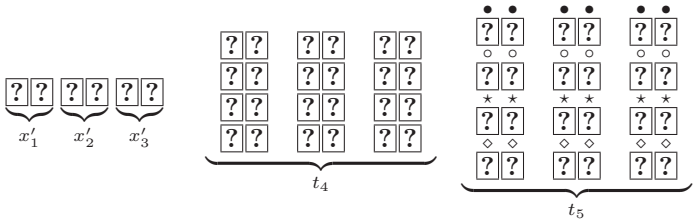


In Step 2 of the garbling stage, a pile-scramble shuffle is applied for every $g \in \mathsf{Gates}$. For $g = 4$ (corresponding to the AND gate), a pile-scramble shuffle is applied for the $\bullet$-row, the $\circ$-row, the $\star$-row and the $\diamond$-row as follows:

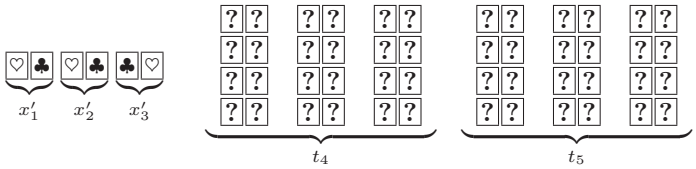


For $g = 5$ (corresponding to the XOR gate), a pile-scramble shuffle is applied

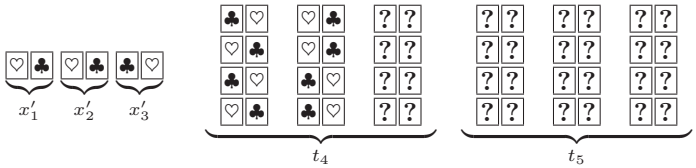
for the $\bullet$-row, the $\circ$-row, the $\star$-row and the $\diamond$-row as follows:



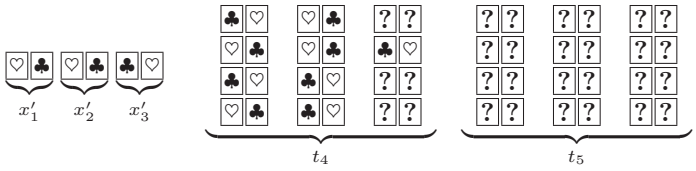
In Step 1 of the evaluation stage, all (randomized) input commitments are opened as follows:



In Step 2 of the evaluation stage, a set of randomized commitments are opened for each gate. For $g = 4$ (corresponding to the AND gate), the leftmost and center columns are opened as follows:

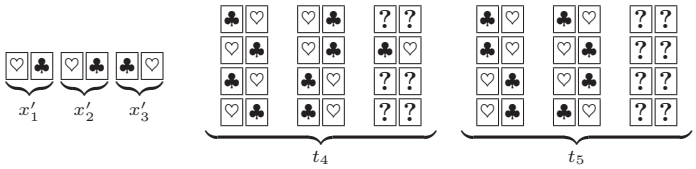


Since $(x'_1, x'_2) = (1, 1)$, the index k_4 is defined $k_4 = 2$. Then, the second row of the rightmost column is opened as follows:



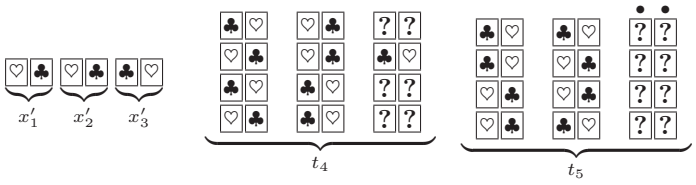
The opened value is $x'_4 = 0$.

For $g = 5$ (corresponding to the XOR gate), the leftmost and center columns are opened as follows:



Since $(x'_4, x'_3) = (0, 0)$, the index k_5 is defined $k_5 = 1$.

The output commitment (pointed by $\bullet$) is the first row of the rightmost column in the output gate as follows:



Finally, the order of the sequence is rearranged so that the output commitment is the leftmost, the opened cards are the next leftmost, and other cards are the rightmost as follows:

This completes the protocol execution. ■

3.3 Protocol with a single uniform closed shuffle

In general, when a protocol applies two (or more) shuffles consecutively, they are combined into a single shuffle. For example, when a protocol applies two shuffles $(\mathsf{shuffle}, \Pi_1, D_1)$ and $(\mathsf{shuffle}, \Pi_2, D_2)$ consecutively, they are combined into a new shuffle $(\mathsf{shuffle}, \Pi, D)$ such that Π is defined by $\Pi = \{\pi_2 \circ \pi_1 \mid \pi_1 \in \Pi_1, \pi_2 \in \Pi_2\}$ and a distribution D is defined as follows:

1. It randomly chooses $\pi_1 \in \Pi_1$ and $\pi_2 \in \Pi_2$ according to the distributions D_1 and D_2, respectively.
2. It outputs $\pi = \pi_2\pi_1 \in \Pi$.

Thus, by combining all shuffles in the garbling stage, we obtain a general protocol with a single shuffle $\mathcal{P}^{\mathsf{b}}_{\mathsf{single},f}$. The protocol $\mathcal{P}^{\mathsf{b}}_{\mathsf{single},f}$ proceeds as follows:

1. Apply a combined shuffle of all shuffles in the garbling stage of the base protocol to the sequence.
2. Perform the evaluation stage the same as in the base protocol.

We can easily observe that $\mathcal{P}^{\mathsf{b}}_{\mathsf{single},f}$ correctly and securely realizes $\mathcal{P}^{\mathsf{b}}_{\mathsf{base},f}$. The remaining task is to show that the combined shuffle used in $\mathcal{P}^{\mathsf{b}}_{\mathsf{single},f}$ is a uniform closed shuffle. Note that, in general, a combined shuffle is not necessarily a uniform closed shuffle even if all underlying shuffles are uniform closed shuffles. We give two examples of the case that the combining technique does not preserve the property of underlying shuffles.

Example 12. We give an example of two closed shuffles but the combined shuffle of them is not closed. Let $(\mathsf{shuffle}, \{\mathsf{id}, (1\ 2)\})$ be the first shuffle and let $(\mathsf{shuffle}, \{\mathsf{id}, (2\ 3)\})$ be the second shuffle. Then, the combined shuffle of them is a non-closed shuffle $(\mathsf{shuffle}, \{\mathsf{id}, (1\ 2), (2\ 3), (1\ 3\ 2)\})$ since the permutation set does not have $(1\ 3\ 2)(1\ 3\ 2) = (1\ 2\ 3)$.

Example 13. We give an example of two uniform shuffles but the combined shuffle of them is not uniform. Let $(\mathsf{shuffle}, \{\mathsf{id}, (1\ 2), (3\ 4)\})$ be the first shuffle and let $(\mathsf{shuffle}, \{\mathsf{id}, (1\ 2)(3\ 4)\})$ be the second shuffle. Then, the combined shuffle of them is $(\mathsf{shuffle}, \{\mathsf{id}, (1\ 2), (3\ 4), (1\ 2)(3\ 4)\})$, which is not uniform.

Nevertheless, the combined shuffle used in $\mathcal{P}^{\mathsf{b}}_{\mathsf{single},f}$ is a uniform closed shuffle. We prove this fact as follows.

Proposition 3.1. A shuffle used in $\mathcal{P}^{\mathsf{b}}_{\mathsf{single},f}$ is a uniform closed shuffle.

Proof. For $w \in \mathsf{Wires} \setminus \mathsf{Outputs}$, let $S_{1,w}$ be the shuffle $(\mathsf{pileShuffle}, P^{(w)}_{\mathsf{left}}, P^{(w)}_{\mathsf{right}})$ in Step 1 of the garbling stage, and for $g \in \mathsf{Gates}$, let $S_{2,g}$ be the shuffle $(\mathsf{pileShuffle}, P^{(g)}_1, P^{(g)}_2, P^{(g)}_3, P^{(g)}_4)$ in Step 2 of the garbling stage. The statement is that the shuffle S^* which is combined all the shuffles $\{S_{1,w}\}_w$ and $\{S_{2,g}\}_g$ is a uniform closed shuffle. We first show that S^* is a closed shuffle. As already seen in the proof of the correctness of the base protocol, each shuffle $S_{1,w}$ is equivalent to masking the wire value by a random value $r_w \in \{0,1\}$ and each shuffle $S_{2,g}$ is equivalent to shuffling four rows of the truth table of g. Thus, they are *commutative*, i.e., exchanging the order of any two shuffles provides the same combined shuffle. Therefore, S^* is a closed shuffle. As already seen in the proof of the security of the base protocol, given the input $x \in \{0,1\}^n$, a distribution of a view $\mathsf{view}_{\mathcal{P}}(\Gamma_x, x)$ is a uniform distribution of all possible views $V_{\mathcal{P}}$, S^* is also a uniform shuffle. Thus, S^* is a uniform closed shuffle. ■

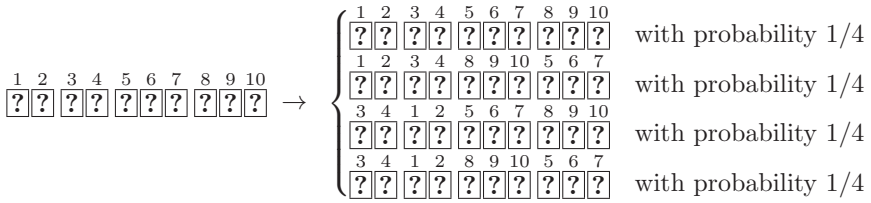
3.4 Protocol with two pile-scramble shuffles

Among uniform closed shuffles, three types of shuffles – a random cut, a random bisection cut, and a pile-scramble shuffle – are considered the most easiest to perform physically. In this section, we construct a general protocol with two *extended pile-scramble shuffles* using $2n + 24q + \Delta$ cards where Δ is defined in later. An extended pile-scramble shuffle is a uniform closed shuffle that can be performed easily like a standard pile-scramble shuffle. We also construct a general protocol with two standard pile-scramble shuffles using $2n + 24q + \Delta'$ cards where $\Delta' > \Delta$ is defined in later.

3.4.1 Extended pile-scramble shuffle

An *extended pile-scramble shuffle* is a shuffle that is similar to a pile-scramble shuffle except that each pile can have a different number of cards. By applying it to a sequence, all piles having the same number of cards are completely

rearranged. For example, an extended pile-scramble shuffle for four piles $T_1 = \{1, 2\}, T_2 = \{3, 4\}, T_3 = \{5, 6, 7\}$, and $T_5 = \{8, 9, 10\}$ is given as follows:

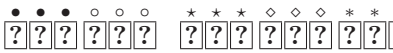


An extended pile-scramble shuffle for $T_1, T_2, \ldots, T_k$ is also denoted as follows:

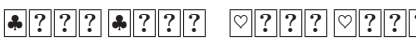
$$(\mathsf{pileShuffle}, T_1, T_2, \ldots, T_k).$$

3.4.2 Batching multiple pile-scramble shuffles

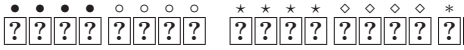
Basic idea. We say that a number of shuffles are *parallel* if there is no card that is touched by two (or more) shuffles. Consider the case that two pile-scramble shuffles are parallel: one shuffle is between "•" and "○", and the other shuffle is among "⋆", "◇", and "∗" as follows:



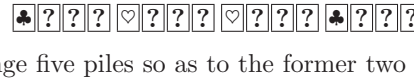
The *batching technique* enables us to combine them into one pile-scramble shuffle by using additional cards. First, two clubs and three hearts are inserted in the sequence as follows:



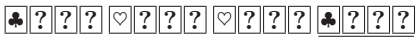
Then, the inserted cards are turned to be face-down cards. Then, a pile-scramble shuffle among "•", "○", '⋆", "◇", and "∗" is applied as follows.



After applying it, open the first cards of all piles as follows:



Finally, rearrange five piles so as to the former two piles have ♣ and the latter three piles have ♡. In this case, the fourth pile (the underlined pile in the following) is moved to the front of the second pile without changing the order of cards in each pile.

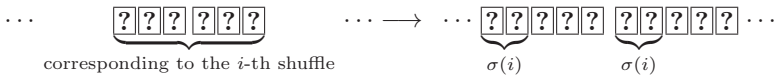


By ignoring opened cards (two ♣ and three ♡), the result sequence in the following is equivalent (as probability distribution) to the result sequence obtained by applying two pile-scramble shuffle sequentially.

This is the core idea of our batching technique.

Batching technique using an extended pile-scramble shuffle. Suppose that we wish to perform N pile-scramble shuffles that are parallel: the i-th pile-scramble shuffle is among ℓ_i piles of n_i cards. Thus, the i-th shuffle treats $\ell_i \cdot n_i$ cards and there are $\sum_{i=1}^{N} \ell_i \cdot n_i$ cards in total. Let $\sigma : [N] \to \{\clubsuit, \heartsuit\}^{\lceil \log_2 N \rceil}$ be an arbitrary injective function. The batching technique proceeds as follows.

1. (**Indexing**) For every $i \in [N]$, insert $\lceil \log_2 N \rceil$ cards representing $\sigma(i)$ to each pile in the i-th shuffle as follows.



   In total, $\sum_{i=1}^{N} \ell_i \cdot \lceil \log_2 N \rceil$ cards are inserted.

2. (**Shuffle**) Perform an extended pile-scramble shuffle among all piles. The number of piles is $\sum_{i=1}^{N} \ell_i$.

3. (**Turning**) Turn the indexes of all piles. In total, $\sum_{i=1}^{N} \ell_i \cdot \lceil \log_2 N \rceil$ cards are turned.

4. (**Rearrangement**) Rearrange all the cards so as to the first ℓ_1 piles are those having the index $\sigma(1)$, the next ℓ_2 piles are those having the index $\sigma(2)$, and so on. Finally, the inserted cards in the Indexing step are removed. (They can be used in future steps as free cards.)

The number of additional cards Δ_0 is:

$$\Delta_0 = \sum_{i=1}^{N} \ell_i \cdot \lceil \log_2 N \rceil.$$

We define a weight function $w_{\clubsuit,k} : \{\clubsuit, \heartsuit\}^k \to \{0, 1, \ldots, k\}$ (resp. $w_{\heartsuit,k}$) that takes a string of k length and outputs the number of $\clubsuit$ (resp. $\heartsuit$). The number of $\boxed{\clubsuit}$ in Δ_0 cards, denoted by $\Delta_0^{\clubsuit}$, is computed as follows:

$$\Delta_0^{\clubsuit} = \sum_{i=1}^{N} w_{\clubsuit,k}(\sigma(i)).$$

The number of $\boxed{\heartsuit}$ in Δ_0 cards, denoted by $\Delta_0^{\heartsuit}$, is $\Delta_0^{\heartsuit} = \Delta_0 - \Delta_0^{\clubsuit}$. The following proposition is useful when $\Delta_0^{\clubsuit} = \Delta_0^{\heartsuit}$ is needed.

Proposition 3.2. For any $N \in \mathbb{N}$, there exists an injective function $\sigma : [N] \to \{\clubsuit, \heartsuit\}^{\lceil \log_2 N \rceil}$ satisfying the following condition:

$$\sum_{i=1}^{N} w_{\clubsuit,k}(\sigma(i)) = \sum_{i=1}^{N} w_{\heartsuit,k}(\sigma(i)).$$

■

Proof. Since the value $\sum_{i=1}^{N} w_{\clubsuit,k}(\sigma(i))$ is determined by the range $R_\sigma = \{\sigma(i) \mid i \in [N]\}$, it is sufficient to find a subset $R \subset \{\clubsuit, \heartsuit\}^{\lceil \log_2 N \rceil}$ such that $\sum_{x \in R} w_{\clubsuit,k}(x) = \sum_{x \in R} w_{\heartsuit,k}(x)$.

If $N = 2^k$ for some $k \in \mathbb{N}$, σ is a bijective function. The range of σ is $R_\sigma = \{\clubsuit, \heartsuit\}^{\lceil \log_2 N \rceil}$. Thus, the condition $\sum_{x \in R_\sigma} w_{\clubsuit,k}(x) = \sum_{x \in R_\sigma} w_{\heartsuit,k}(x)$ is trivially satisfied.

Suppose that $2^{\lceil \log_2 N \rceil} - N = \delta > 0$. Let $R \subset \{\clubsuit, \heartsuit\}^{\lceil \log_2 N \rceil}$ be a variable that is set to $\{\clubsuit, \heartsuit\}^{\lceil \log_2 N \rceil}$ as an initial value. If δ is an even number $\delta = 2\delta'$ for $\delta' \in \mathbb{N}$, remove δ' pairs of $x, x' \in \{\clubsuit, \heartsuit\}^{\lceil \log_2 N \rceil}$ such that $w_{\clubsuit,k}(x) = w_{\heartsuit,k}(x')$. Then, R satisfies $|R| = N$ and $\sum_{x \in R} w_{\clubsuit,k}(x) = \sum_{x \in R} w_{\heartsuit,k}(x)$. If δ is an odd number $\delta = 2\delta' + 1$ for $\delta' \in \mathbb{N}$, remove δ' pairs of $x, x' \in \{\clubsuit, \heartsuit\}^{\lceil \log_2 N \rceil}$ as above and remove one element $x'' \in \{\clubsuit, \heartsuit\}^{\lceil \log_2 N \rceil}$ such that $w_{\clubsuit,k}(x'') = w_{\heartsuit,k}(x'')$. Then, R satisfies $|R| = N$ and $\sum_{x \in R} w_{\clubsuit,k}(x) = \sum_{x \in R} w_{\heartsuit,k}(x)$.

Therefore, we have a desired injective function σ for any $N \in \mathbb{N}$. ■

Batching technique using a standard pile-scramble shuffle. By appending a number of dummy cards, an extended pile-scramble shuffle can be converted into a standard pile-scramble shuffle. Let $n_{\max} = \max(n_1, n_2, \ldots, n_N)$. For the i-th pile-scramble shuffle, $(n_{\max} - n_i)$ dummy cards are appended. The number of dummy cards Δ_1 is:

$$\Delta_1 = \sum_{i=1}^{N} \ell_i \cdot (n_{\max} - n_i).$$

The total number of additional cards of the batching technique using a standard pile-scramble shuffle is:

$$\Delta_0 + \Delta_1 = \sum_{i=1}^{N} \ell_i \cdot (\lceil \log_2 N \rceil + n_{\max} - n_i)\,.$$

3.4.3 Protocol with two extended pile-scramble shuffles

A general protocol with two extended pile-scramble shuffles $\mathcal{P}^{\mathsf{b}}_{\mathsf{two1},f}$ is obtained by applying the batching technique to the base protocol. It proceeds as follows:

1. Apply the batching technique (using an extended pile-scramble shuffle) to Step 1 of the garbling stage in the base protocol.
2. Apply the batching technique (using a standard pile-scramble shuffle) to Step 2 of the garbling stage in the base protocol.
3. Perform the evaluation stage in the base protocol.

Note that the batching technique in Step 2 is of a standard pile-scramble shuffle since all original shuffles in Step 2 have the same number of cards. The number of additional cards Δ is given as follows:

$$\Delta = \max(2(n+q-1)\lceil \log_2(n+q-1) \rceil, 4q\lceil \log_2 q \rceil).$$

The reason why we take a max is all additional cards in the batching technique can be reused in the next steps. From Proposition 3.2, we can have σ such that the numbers of $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$ are the same.

3.4.4 Protocol with two standard pile-scramble shuffles

By appending a number of dummy cards, we obtain a general protocol with two standard pile-scramble shuffles $\mathcal{P}^{\mathsf{b}}_{\mathsf{two2},f}$ as follows:

1. Apply the batching technique (using a standard pile-scramble shuffle) to Step 1 of the garbling stage in the base protocol.
2. Apply the batching technique (using a standard pile-scramble shuffle) to Step 2 of the garbling stage in the base protocol.
3. Perform the evaluation stage in the base protocol.

We compute the number of additional cards $\Delta' \in \mathbb{N}$ as follows. Let $\delta_0 = 2(n+q-1)\lceil \log_2(n+q-1) \rceil$ and let $\delta_1 = 4q\lceil \log_2 q \rceil$. We assume that the numbers of $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$ are balanced in these cards from Proposition 3.2. The number of dummy cards δ_2 used in Step 1 is given as follows:

$$\delta_2 = \sum_{w=1}^{n+q-1} 2 \cdot (n_{\max} - |P^{(w)}_{\mathsf{left}}|),$$

where $n_{\max} = \max\{|P^{(w)}_{\mathsf{left}}| \mid 1 \leq w \leq n+q-1\}$. Unlike the δ_0 cards and δ_1 cards, the δ_2 cards are not necessarily balanced. Let $\alpha = \delta_2/2$. If α is an even number, they are $\Delta'_1/2$ $\boxed{\clubsuit}$s and $\Delta'_1/2$ $\boxed{\heartsuit}$s, i.e., they are balanced. If α is an odd number, they are $(\Delta'_1/2+1)$ $\boxed{\clubsuit}$s and $(\Delta'_1/2-1)$ $\boxed{\heartsuit}$s, i.e., they are not balanced. From this observation, we obtain the total number of addition cards Δ' as follows:

$$\Delta' = \begin{cases} \delta_1 & \text{if } \delta_1 > \delta_0 + \delta_2 \\ \delta_0 + \delta_2 + 1 & \text{if } \delta_1 = \delta_0 + \delta_2 \\ \delta_0 + \delta_2 & \text{otherwise.} \end{cases}$$

The most technical case is the middle case. In the middle case, although Step 1 uses $(\delta_0 + \delta_2)$ cards, the number of pairs of $\boxed{\clubsuit}\boxed{\heartsuit}$ is $(\delta_0 + \delta_2)/2 - 1$. Thus, to have $(\delta_0 + \delta_2)$ pairs of $\boxed{\clubsuit}\boxed{\heartsuit}$, one $\boxed{\heartsuit}$ must be added.

Chapter 4

Protocols with Private Permutations

4.1 Introducing private permutations

4.1.1 Private permutations

Let n be the number of inputs and ℓ be the number of cards. A *private permutation* is defined by $(\mathsf{privatePerm}, i, \pi)$, where $i \in [n]$ is an index of the input bit and $\pi \in S_\ell$ is a permutation. For a sequence s of ℓ cards and an input $x \in \{0,1\}^n$, π is applied to s if $x_i = 1$, otherwise the identity permutation id is applied to s as follows:

$$s \ \rightarrow \ \begin{cases} \pi(s) & \text{if } x_i = 1 \\ s & \text{otherwise.} \end{cases}$$

Applying a permutation is covertly done by the player having the input bit x_i so that other players cannot know which permutation is chosen. The set of private permutations is defined by:

$$\mathsf{PrivatePerm}_{n,\ell} := \{(\mathsf{privatePerm}, i, \pi) \mid i \in [n], \pi \in S_\ell\}.$$

4.1.2 Notations

In this chapter, we use notations as follows.

Deck. We assume that a card specification is that of binary cards Binary. That is, all decks $\overline{\mathcal{D}}$ used in this chapter are of type $\overline{\mathcal{D}} = (\mathsf{Binary}, \mathcal{D})$.

Cards. We regard $\clubsuit = 0$ and $\heartsuit = 1$ and use $\{\clubsuit, \heartsuit\}$ and $\{0,1\}$ interchangeably. If a card is $\mathbf{?}/\clubsuit$ if $x = 0$ and $\mathbf{?}/\heartsuit$ otherwise, we denote it as follows:

$$\underset{x}{\boxed{\mathbf{?}}}.$$

Operations. We assume that the set of operations is $\mathcal{O}^{\mathsf{p}}_{n,\ell}$ as follows:

$$\mathcal{O}^{\mathsf{p}}_{n,\ell} = \mathsf{Perm}_\ell \cup \mathsf{Turn}_\ell \cup \mathsf{PrivatePerm}_{n,\ell},$$

where n is the number of inputs and ℓ is the number of cards.

Action function. Every action function A in this chapter can be represented by a list of pairs of a visible sequence and an operation as follows:

$$((v_0, \mathsf{op}_1), (v_1, \mathsf{op}_2), \ldots, (v_{k-1}, \mathsf{op}_k)) \in (\mathsf{Vis}^{\overline{\mathcal{P}}} \times \mathcal{O}^{\mathsf{p}}_{n,\ell})^k,$$

(Recall that $\mathsf{Vis}^{\overline{\mathcal{P}}}$ is the set of visible sequences. See Definition 2.3.) For each round i ($1 \le i \le k$), the action function A outputs $\perp$ if the visible sequence of the current sequence is *not* matched with v_{i-1}, otherwise op_i. In the round $k+1$ (i.e. just after applying op_k), A always outputs $\perp$.

View. We omit revealed information in a view since all operations in $\mathcal{O}^{\mathsf{p}}_{n,\ell}$ do not produce revealed information. Thus, for a protocol $\mathcal{P}$, an initial sequence $s_0 \in \mathsf{Seq}^{\overline{\mathcal{P}}}$, and an input $x \in \{0,1\}^n$, a view $\mathsf{view}_{\mathcal{P}}(s_0, x)$ is represented by a list of visible sequences as follows:

$$\mathsf{view}_{\mathcal{P}}(s_0, x) = (v_0 \to v_1 \to v_2 \to \cdots \to v_{k'}),$$

where $v_{k'}$ is the final visible sequence in the sense that the action function A outputs $\perp$. (If A has k operations, then $k' \le k$.)

4.1.3 Existing protocols in our model

Example 14. A three-card AND protocol proposed by Marcedone, Wen, and Shi [24] is given by $(2, \{0,1\}, (\mathsf{Binary}, \{(\clubsuit/?)^2, \heartsuit/?\}), \mathcal{O}^{\mathsf{p}}_{2,3}, A)$. The action function A is defined by $A = ((v_1, \mathsf{op}_1), (v_2, \mathsf{op}_2))$ as follows:

1. $v_1 = (?, ?, ?)$ and $\mathsf{op}_1 = (\mathsf{privatePerm}, 1, (2\ 3))$.
2. $v_2 = (?, ?, ?)$ and $\mathsf{op}_2 = (\mathsf{privatePerm}, 2, (1\ 2))$.

The functionality is defined as follows:

$$\underset{\clubsuit\,\clubsuit\,\heartsuit}{[?][?][?]} \to \underset{x_1 \wedge x_2}{[?]}\ \underset{\perp\ \perp}{[?][?]}.$$

The correctness is verified by the following diagram:

(x_1, x_2)	Sequence	Output
$(0,0)$	$\clubsuit\clubsuit\heartsuit \xrightarrow{\mathsf{op}_1} \clubsuit\clubsuit\heartsuit \xrightarrow{\mathsf{op}_2} \clubsuit\clubsuit\heartsuit$	0
$(0,1)$	$\clubsuit\clubsuit\heartsuit \xrightarrow{\mathsf{op}_1} \clubsuit\clubsuit\heartsuit \xrightarrow{\mathsf{op}_2} \clubsuit\clubsuit\heartsuit$	0
$(1,0)$	$\clubsuit\clubsuit\heartsuit \xrightarrow{\mathsf{op}_1} \clubsuit\heartsuit\clubsuit \xrightarrow{\mathsf{op}_2} \clubsuit\heartsuit\clubsuit$	0
$(1,1)$	$\clubsuit\clubsuit\heartsuit \xrightarrow{\mathsf{op}_1} \clubsuit\heartsuit\clubsuit \xrightarrow{\mathsf{op}_2} \heartsuit\clubsuit\clubsuit$	1

The security is trivial since the view is the same for all input $(x_1, x_2) \in \{0,1\}^2$ as follows:

$$\mathsf{view}(s_0, (x_1, x_2)) = \big((?, ?, ?) \to (?, ?, ?) \to (?, ?, ?)\big),$$

where $s_0 = (?/\clubsuit, ?/\clubsuit, ?/\heartsuit)$. ■

Example 15. A (slightly modified version of) four-card majority voting protocol proposed by Nakai, Shirouchi, Iwamoto, and Ohta [35] is given by:

$$(3, \{0,1\}, (\mathsf{Binary}, \{(\clubsuit/?)^2, (\heartsuit/?)^2\}, \mathcal{O}^{\mathsf{p}}_{3,4}, A).$$

The action function A is defined by $A = ((v_1, \mathsf{op}_1), (v_2, \mathsf{op}_2), (v_3, \mathsf{op}_3))$ as follows:

1. $v_1 = (?, ?, ?, ?)$ and $\mathsf{op}_1 = (\mathsf{privatePerm}, 1, (3\ 4))$.
2. $v_2 = (?, ?, ?, ?)$ and $\mathsf{op}_2 = (\mathsf{privatePerm}, 2, (1\ 2\ 3))$.
3. $v_3 = (?, ?, ?, ?)$ and $\mathsf{op}_3 = (\mathsf{privatePerm}, 3, (1\ 3))$.

The functionality is defined as follows:

$$\underset{\clubsuit\ \heartsuit\ \clubsuit\ \heartsuit}{\boxed{?}\boxed{?}\boxed{?}\boxed{?}} \to \underset{f(x_1,x_2,x_3)}{\boxed{?}} \underset{\perp\ \perp\ \perp}{\boxed{?}\boxed{?}\boxed{?}},$$

where $f(x_1, x_2, x_3) = \mathsf{p}(x_1 + x_2 + x_3 \geq 2)$.

The correctness is verified by the following diagram:

(x_1, x_2, x_3)	Sequence	Output
$(0,0,0)$	$\clubsuit\heartsuit\clubsuit\heartsuit \xrightarrow{\mathsf{op}_1} \clubsuit\heartsuit\clubsuit\heartsuit \xrightarrow{\mathsf{op}_2} \clubsuit\heartsuit\clubsuit\heartsuit \xrightarrow{\mathsf{op}_3} \clubsuit\heartsuit\clubsuit\heartsuit$	0
$(0,0,1)$	$\clubsuit\heartsuit\clubsuit\heartsuit \xrightarrow{\mathsf{op}_1} \clubsuit\heartsuit\clubsuit\heartsuit \xrightarrow{\mathsf{op}_2} \clubsuit\heartsuit\clubsuit\heartsuit \xrightarrow{\mathsf{op}_3} \clubsuit\heartsuit\clubsuit\heartsuit$	0
$(0,1,0)$	$\clubsuit\heartsuit\clubsuit\heartsuit \xrightarrow{\mathsf{op}_1} \clubsuit\heartsuit\clubsuit\heartsuit \xrightarrow{\mathsf{op}_2} \clubsuit\clubsuit\heartsuit\heartsuit \xrightarrow{\mathsf{op}_3} \clubsuit\clubsuit\heartsuit\heartsuit$	0
$(0,1,1)$	$\clubsuit\heartsuit\clubsuit\heartsuit \xrightarrow{\mathsf{op}_1} \clubsuit\heartsuit\clubsuit\heartsuit \xrightarrow{\mathsf{op}_2} \clubsuit\clubsuit\heartsuit\heartsuit \xrightarrow{\mathsf{op}_3} \heartsuit\clubsuit\clubsuit\heartsuit$	1
$(1,0,0)$	$\clubsuit\heartsuit\clubsuit\heartsuit \xrightarrow{\mathsf{op}_1} \clubsuit\heartsuit\heartsuit\clubsuit \xrightarrow{\mathsf{op}_2} \clubsuit\heartsuit\heartsuit\clubsuit \xrightarrow{\mathsf{op}_3} \clubsuit\heartsuit\heartsuit\clubsuit$	0
$(1,0,1)$	$\clubsuit\heartsuit\clubsuit\heartsuit \xrightarrow{\mathsf{op}_1} \clubsuit\heartsuit\heartsuit\clubsuit \xrightarrow{\mathsf{op}_2} \clubsuit\heartsuit\heartsuit\clubsuit \xrightarrow{\mathsf{op}_3} \heartsuit\heartsuit\clubsuit\clubsuit$	1
$(1,1,0)$	$\clubsuit\heartsuit\clubsuit\heartsuit \xrightarrow{\mathsf{op}_1} \clubsuit\heartsuit\heartsuit\clubsuit \xrightarrow{\mathsf{op}_2} \heartsuit\clubsuit\heartsuit\clubsuit \xrightarrow{\mathsf{op}_3} \heartsuit\clubsuit\heartsuit\clubsuit$	1
$(1,1,1)$	$\clubsuit\heartsuit\clubsuit\heartsuit \xrightarrow{\mathsf{op}_1} \clubsuit\heartsuit\heartsuit\clubsuit \xrightarrow{\mathsf{op}_2} \heartsuit\clubsuit\heartsuit\clubsuit \xrightarrow{\mathsf{op}_3} \heartsuit\clubsuit\heartsuit\clubsuit$	1

The security is trivial since the view is the same for all input $(x_1, x_2, x_3) \in \{0,1\}^3$ as follows:

$$\mathsf{view}(s_0, (x_1, x_2, x_3)) = \big((?, ?, ?, ?) \to (?, ?, ?, ?) \to (?, ?, ?, ?) \to (?, ?, ?, ?)\big),$$

where $s_0 = (?/\clubsuit, ?/\heartsuit, ?/\clubsuit, ?/\heartsuit)$. ■

4.2 Active security

Although security is defined in Definition 2.8, it is unsatisfied for protocols with private permutations. This is because unlike public operations, such as permutations, turnings, and shuffles, a private permutation is done in a physically isolated location so that other players cannot know which permutation is chosen. This situation produces a new threat; it is easy to behave maliciously in a private permutation. To prevent a malicious attack, we propose *active security*. As mentioned in Section 1.3.2, a protocol is said to be actively secure if whenever a permutation is chosen maliciously in a private permutation, the protocol does not reveal any secret information and the malicious player is always detected. In Section 4.2.1, we define a set of executed permutations for a potentially malicious player. In Section 4.2.2, we define active security.

4.2.1 Executed permutation list

Let $\Pi = (n, \{0,1\}, \overline{\mathcal{D}}, \mathcal{O}^{\mathsf{P}}_{n,\ell}, A)$ be a protocol where A has k private permutations. Note that the length of A can be larger than k when it has an operation other than a private permutation. We define an *executed permutation list* as a list of k permutations applied in the protocol execution. This definition is introduced to capture malicious behavior during protocol execution. Before defining it formally, we first present an example of an executed permutation list.

Example 16. Let Π be the four-card majority voting protocol [35] described in Example 15. An example of executed permutation list is $\mathcal{E}_1 = (\mathsf{id}, (1\ 2\ 3), (1\ 3))$, which corresponds to an input $x = (0,1,1)$. $\mathcal{E}_2 = (\mathsf{id}, (1\ 2\ 3), \mathsf{id})$ is another example, which corresponds to an input $x = (0,1,0)$. ■

To define a malicious execution, we must determine the permutations that can be applied maliciously in a private permutation. We define a *permutation model* by a specification whose permutations can be applied in a private permutation. We define three permutation models: a *bare-bone model*, an *envelope model* and an *envelope-with-ring* model.

Bare-bone model: For a private permutation $\mathsf{op} = (\mathsf{privatePerm}, i, \pi)$, a (potentially malicious) player can choose a permutation from a set of permutations $\mathbf{\Pi}_\pi$ that is defined as follows:

$$\mathbf{\Pi}_\pi := \{\pi' \mid \pi' \in S_\ell, \mathsf{fix}(\pi') = \mathsf{fix}(\pi)\},$$

where fix is a function that takes a permutation $\pi \in S_\ell$ and outputs a set of all fixed points of π. In this model, no physical additional objects such as envelopes are used. Thus, a player performing op is given the j-th card for all $j \notin \mathsf{fix}(\pi)$, and applies any permutation to the sequence. The reason that the player is not given cards in positions $\mathsf{fix}(\pi)$ is that these cards are not moved regardless of the input bit x_i.

Envelope model: We say that a permutation π is an *extended transposition* if its transposition decomposition is

$$\pi = (i_1\, i_2)(i_3\, i_4)\cdots(i_{2k-1}\, i_{2k}),$$

where all i_j's are different. We also denote it as follows:

$$\pi = \mathsf{et}((i_1, i_3, \ldots, i_{2k-1}) \leftrightarrow (i_2, i_4, \ldots, i_{2k})).$$

In this model, for a private permutation $\mathsf{op} = (\mathsf{privatePerm}, i, \pi)$ of an extended transposition π, a set of permutations $\mathbf{\Pi}_\pi$ is defined as $\mathbf{\Pi}_\pi = \{\mathsf{id}, \pi\}$; that is, even a malicious player applies either id or π. This is justified by the use of envelopes. Suppose that two lockable envelopes are available that contain a number of cards. When a protocol enters op, all players do the following: k cards in positions $(i_1, i_3, \cdots, i_{2k-1})$ are putted into an envelope while other cards in positions $(i_2, i_4, \cdots, i_{2k})$ are putted into another envelope. Then, the two envelopes are given to the i-th player. Finally, the player swaps if $x_i = 1$ and does nothing otherwise. Here, a private permutation that is not an extended transposition is the same as that of the bare-bone model.

Ring-with-envelope model: We say that a permutation π is an *extended cycle of length* m if its cycle decomposition is k tuples of cycles of length m for some k, m. For example, $\pi = (1\,2\,3) \circ (4\,5\,6)$ is an extended cycle of length $m = 3$. An extended transposition is the same notion as an extended cycle of length $m = 2$. In this model, for a private permutation $\mathsf{op} = (\mathsf{privatePerm}, i, \pi)$ of an extended cycle π of length m, a set of permutations $\mathbf{\Pi}_\pi$ is defined as $\mathbf{\Pi}_\pi = \{\mathsf{id}, \pi, \pi^2, \cdots, \pi^{m-1}\}$. This is justified by the use of a special ring containing a number of envelopes, which we call a *ring with envelope*. As mentioned in Section 1.3.2, a ring with envelopes contains a number of identical lockable envelopes that are hung from the ring and the ring and envelopes can be opened and closed when all players agree. Each envelope can be cyclically moved along with the ring. For example, a ring with envelopes containing four envelopes (A, B, C, D) can be in four possible states $(A, B, C, D), (B, C, D, A), (C, D, A, B), (D, A, B, C)$. When a protocol enters op, each card is putted into one of m envelopes. Then, the ring with m envelopes is given to the i-th player. Finally, the player cyclically rotates the envelopes. Here, a private permutation that is not an extended cycle is the same as that of the bare-bone model.

Now we define an executed permutation list $\mathcal{E}$.

Definition 4.1 (Executed permutation list)**.** Let Π be a protocol whose i-th private permutation is $(\mathsf{privatePerm}, j_i, \pi_i)$. We say that a list of k permutations $(\pi'_1, \pi'_2, \cdots, \pi'_k) \in (S_\ell)^k$ is an *executed permutation list* in a permutation model if each permutation π'_i is an element of the set $\mathbf{\Pi}_{\pi_i}$ in the permutation model. ■

Next we define an honest execution and a malicious execution.

Definition 4.2 (Honest execution and first malicious point)**.** Let Π be a protocol whose i-th private permutation is $(\mathsf{privatePerm}, j_i, \pi_i)$ and let $\mathcal{E}$ be an executed permutation list $\mathcal{E} = (\pi'_1, \pi'_2, \cdots, \pi'_k)$. We say that a subsequence $\mathcal{E}'_i = (\pi'_1, \pi'_2, \cdots, \pi'_i)$ for $1 \leq i \leq k$ is *honest* if there exists an input $x \in \{0,1\}$ such that for all i, each permutation π'_i is consistent with the input bit x_{j_i}. We say that an executed permutation list is an *honest execution* if it is honest. It is called a *malicious execution* if it is not honest. For an executed permutation list $\mathcal{E}$, the *first malicious point* i is defined as the minimum number i such that the prefix of $\mathcal{E}$ of length $i-1$ is honest but that of length i is not honest. If such an i does not exist (i.e., $\mathcal{E}$ is an honest execution), it is defined as $k+1$. ■

Example 17. We use the same example as Example 16. An executed permutation list $\mathcal{E}_1 = (\mathsf{id}, (1\ 2\ 3), (1\ 3))$ is an honest execution since it is consistent with the input $(0,1,1)$. $\mathcal{E}_2 = (\mathsf{id}, (1\ 2\ 3), \mathsf{id})$ is also an honest execution since it is consistent with the input $(0,1,0)$. $\mathcal{E}_3 = (\mathsf{id}, (1\ 2), (1\ 3))$ is a malicious execution since the second permutation is neither $(1\ 2\ 3)$ nor id. The first malicious point of $\mathcal{E}_3$ is 2. ■

4.2.2 Active security

Although we have already defined a view $\mathsf{view}_{\mathcal{P}}(s_0, x)$, it does not capture an execution when some players behave maliciously. Due to this, we re-define a view as follows:

Definition 4.3 (View)**.** Let $\mathcal{P} = (n, \{0,1\}, \overline{\mathcal{D}}, \mathcal{O}^{\mathsf{p}}_{n,\ell}, A)$ be a protocol. Let $s_0 \in \mathsf{Seq}^{\overline{\mathcal{D}}}$ be a sequence. Let $\mathcal{E}$ be an executed permutation list. A *view with* s_0 *and* $\mathcal{E}$, denoted by $\mathsf{view}_{\mathcal{P}}(s_0, \mathcal{E})$, is defined by a list of visible sequences when private permutations are done according to $\mathcal{E}$. ■

Based on the new definition of view, the security in Definition 2.8 (in this chapter, we call this notion *passive security*) is re-defined as follows:

Definition 4.4 (Passive security)**.** Let $\mathcal{P} = (n, \{0,1\}, \overline{\mathcal{D}}, \mathcal{O}^{\mathsf{p}}_{n,\ell}, A)$ be a protocol. Let F be a functionality whose initial sequence is $s_0 \in \mathsf{Seq}^{\overline{\mathcal{D}}}$. We say that $\mathcal{P}$ *securely realizes* F *with passive security* if for any honest executions $\mathcal{E}, \mathcal{E}'$ of $\mathcal{P}$, it holds that $\mathsf{view}_{\mathcal{P}}(s_0, \mathcal{E}) = \mathsf{view}_{\mathcal{P}}(s_0, \mathcal{E}')$. ■

The active security is defined as follows:

Definition 4.5 (Active security)**.** Let $\mathcal{P} = (n, \{0,1\}, \overline{\mathcal{D}}, \mathcal{O}^{\mathsf{p}}_{n,\ell}, A)$ be a protocol. Let $\mathcal{F}$ be a functionality whose initial sequence is $s_0 \in \mathsf{Seq}^{\overline{\mathcal{D}}}$. We say that $\mathcal{P}$ *securely realizes* $\mathcal{F}$ *with actively secure in a permutation model* if for any execution permutation lists $\mathcal{E}, \mathcal{E}'$ in the permutation model, it holds that $\mathsf{view}_{\mathcal{P}}(s_0, \mathcal{E}) = \mathsf{view}_{\mathcal{P}}(s_0, \mathcal{E}')$ if and only if the first malicious points of $\mathcal{E}, \mathcal{E}'$ are the same. ■

4.3 One-round protocol in the envelope model

In this section, we construct a general protocol called *one-round protocol* that requires exactly one private permutation for each input bit. To compute a function $f : \{0,1\}^n \to \{0,1\}$, the total number of private permutations is n. It has the minimum number of private permutations since any protocol for a non-trivial function requires at least one private permutation for each input bit. The construction is based on a truth table. The number of cards is 2^n which equals to the number of lines in the truth table. Since all permutations used in the protocol are extended transpositions, it is *actively secure in the envelope model*.

Functionality. Let $f : \{0,1\}^n \to \{0,1\}$ be any function. The functionality $\mathcal{F}^{\mathsf{p}}_{\mathsf{round},f} = (s_{\mathsf{in}}, s_{\mathsf{out}}(x))$ is defined as follows:

$$s_{\mathsf{in}} = (f(0\cdots00), f(0\cdots01), f(0\cdots10), f(0\cdots11), \ldots, f(1\cdots11)).$$
$$s_{\mathsf{out}} = (f(x_1\cdots x_{n-1}x_n), \bot^{2^n-1}).$$

Protocol. One round protocol $\mathcal{P}^{\mathsf{p}}_{\mathsf{round},f}$ is defined as follows:

$$\mathcal{P}^{\mathsf{p}}_{\mathsf{round},f} = (n, \{0,1\}, (\mathsf{Binary}, \{(\clubsuit/\mathbf{?})^k, (\heartsuit/\mathbf{?})^{2^n-k}\}, \mathcal{O}^{\mathsf{p}}_{n,2^n}, A),$$

where $a := |\{x \mid x \in \{0,1\}^n, f(x) = 0\}|$. The action function

$$A = ((v_0, \mathsf{op}_1), (v_1, \mathsf{op}_2), \ldots, (v_{n-1}, \mathsf{op}_n))$$

is defined as follows. For $i \in [n]$, the visible sequence is defined as $v_{i-1} = \mathbf{?}^{2^n}$ and the operation is defined as follows:

$$\mathsf{op}_i = (\mathsf{privatePerm}, i, \mathsf{et}((1,2,\ldots,2^{n-i}) \leftrightarrow (2^{n-i}+1, 2^{n-i}+2, \ldots, 2^{n-i+1})).$$

Note that this permutation is an extended transposition that swaps the first 2^{n-i} cards and the next 2^{n-i} cards. When $i = 1$, it swaps the former half and the latter half of the sequence. When $i = n$, it swaps the first card and the second card.

Correctness. The initial sequence s_0 is given as follows:

$$s_0 = \big(f(0\cdots00), f(0\cdots01), f(0\cdots10), f(0\cdots11), \ldots, f(1\cdots11)\big).$$

At the first round, the former half and the latter half of the sequence are swapped if $x_1 = 1$. Thus, the sequence just after applying op_1 is as follows:

$$\big(f(x_1\cdots00), f(x_1\cdots01), f(x_1\cdots10), f(x_1\cdots11), \cdots\big).$$

Repeating this process, the sequence just after applying op_i is as follows:

$$\big(f(x_1x_2\cdots x_i0\cdots00), f(x_1x_2\cdots x_i0\cdots01), f(x_1x_2\cdots x_i0\cdots10), \cdots\big).$$

Thus, the first card of the final sequence is $f(x_1\cdots x_{n-1}x_n)$. Therefore, the $\mathcal{P}^{\mathsf{p}}_{\mathsf{round},f}$ correctly realizes the functionality $\mathcal{F}^{\mathsf{p}}_{\mathsf{round},f}$.

Security. The passive security of $\mathcal{P}^{\mathsf{p}}_{\mathsf{round},f}$ is trivial since all cards are always face-down. Since every private permutation is of an extended transposition, any out-of-range attack does not happen. Since every input bit is only called one time, any inconsistent attack does not happen. Thus, $\mathcal{P}^{\mathsf{p}}_{\mathsf{round},f}$ securely realizes $\mathcal{F}^{\mathsf{p}}_{\mathsf{round},f}$ with active security in the envelope model.

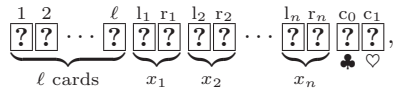
4.4 Protocol compiler in the envelope model

In this section we design a protocol compiler that converts a passively secure protocol computing a function $f : \{0,1\}^n \to \{0,1\}$ into an actively secure protocol computing the same function in the envelope model, while the number of cards and the number of private permutations are increased.

4.4.1 Commit-and-prove technique

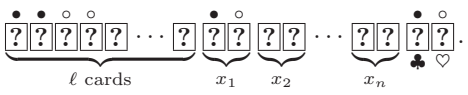
Let $\mathcal{P}$ be a passively secure protocol where every private permutation is of an extended transposition. To enhance the security of $\mathcal{P}$ from passive to active, it is enough to prevent inconsistent attacks. In the subsection, we introduce a *commit-and-prove technique* which prevents inconsistent attacks.

Assume that the number of cards of $\mathcal{P}$ is ℓ. Then, a compiled protocol $\mathcal{P}'$ to be defined requires $\ell + 2n + 2$ cards: $2n$ cards are formed n commitments to $x_1, x_2, \cdots, x_n \in \{0,1\}$ as follows:



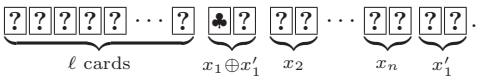
where all l_i, r_i and c_i represent positions. When $\mathcal{P}$ enters a private transposition $(\mathsf{privatePerm}, 1, (1,2), (3,4))$, $\mathcal{P}'$ performs as follows:

1. $(\mathsf{privatePerm}, 1, \mathsf{et}((1,2,\mathrm{l}_1,\mathrm{c}_0) \leftrightarrow (3,4,\mathrm{r}_1,\mathrm{c}_1)))$: The "•" group and the "○" group are swapped if $x_1 = 1$ as follows:

   Let $x'_1 \in \{0,1\}$ be the chosen bit in the private transposition. If it is honestly executed, it holds $x'_1 = x_1$.

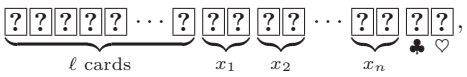
2. $(\mathsf{turn}, \{\mathrm{l}_1\})$: Open the left card of the 1st commitment. The opened symbol is ♣ if and only if it holds $x'_1 = x_1$.

3. $(\mathsf{turn}, \{\mathrm{l}_1\})$: Before entering the turning operation, check that the current visible sequence is matched with $\mathbf{?}^{\ell}\clubsuit\mathbf{?}^{2n+1}$. If it is not the case, the action

function A outputs $\perp$ and an execution is terminated. Otherwise, close the opened card.

4. $(\mathsf{perm}, (\mathrm{l}_1\,\mathrm{c}_0) \circ (\mathrm{r}_1\,\mathrm{c}_1))$: Now we know that the last two cards consist of a commitment to x_1. This and the 1st commitment are exchanged. Then, we obtain the following sequence:



i.e., for the former ℓ cards, $(\mathsf{privatePerm}, 1, (1,2), (3,4))$ is honestly executed, and for the latter $2n+2$ cards, it is the same sequence before applying the step 1.

4.4.2 Protocol compiler

Original functionality. A functionality $\mathcal{F}$ is defined as $\mathcal{F} = (s_{\mathsf{in}}(x), s_{\mathsf{out}}(x))$.

Original protocol. A protocol $\mathcal{P}$ is defined as follows:

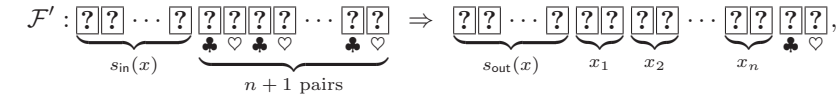
$$\mathcal{P} = (n, \{0,1\}, (\mathsf{Binary}, \mathcal{D}), \mathcal{O}^{\mathsf{P}}_{n,\ell}, A).$$

Here, $\ell = |\mathcal{D}|$. The action function is defined as:

$$A = \big((v_0, \mathsf{op}_1), (v_1, \mathsf{op}_2), \ldots, (v_{k-1}, \mathsf{op}_k).$$

We assume that $\mathcal{P}$ correctly and passively realizes the functionality $\mathcal{F}$. We also assume that every private permutation of $\mathcal{P}$ is an extended transposition. Note that this assumption is reasonable since without loss of generality every passively secure protocol can be transformed into that satisfying this assumption by decomposing a permutation into a number of transpositions.

Compiled functionality. A functionality $\mathcal{F}'$ is defined as follows:

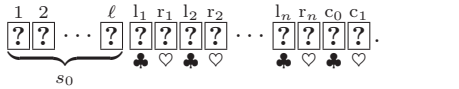


where $s_{\mathsf{in}}(x)$ is the input sequence of $\mathcal{F}$.

Compiled protocol. A protocol $\mathcal{P}'$ is defined as follows:

$$\mathcal{P}' = (n, \{0,1\}, (\mathsf{Binary}, \mathcal{D} \cup \{(\clubsuit/?)^{n+1}, (\heartsuit/?)^{n+1}\}), \mathcal{O}^{\mathsf{P}}_{n,\ell+m}, A'),$$

where $m = 2n+2$. For ease of explanation, we use notations as follows:



That is, $\mathrm{l}_i = \ell + 2i - 1$ and $\mathrm{r}_i = \ell + 2i$ for $i \in [n]$, and $\mathrm{c}_j = \ell + 2n + 1 + j$ for $j \in \{0, 1\}$. The action function A' is defined as follows:

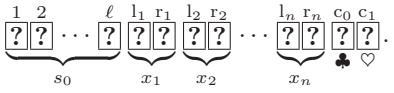
$$\begin{aligned} A' = \big(& \underbrace{(v'_{0,1}, \mathsf{op}'_{0,1}), (v'_{0,2}, \mathsf{op}'_{0,2}), \dots, (v'_{0,n}, \mathsf{op}'_{0,n})}_{\text{Commit stage}}, \\ & \underbrace{(v'_{1,1}, \mathsf{op}'_{1,1}), (v'_{1,2}, \mathsf{op}'_{1,2}), \dots, (v'_{1,4}, \mathsf{op}'_{1,4})}_{\text{corresponding to } (v_0, \mathsf{op}_1)}, \\ & \underbrace{(v'_{2,1}, \mathsf{op}'_{2,1}), (v'_{2,2}, \mathsf{op}'_{2,2}), \dots, (v'_{2,4}, \mathsf{op}'_{2,4})}_{\text{corresponding to } (v_1, \mathsf{op}_2)}, \dots \\ & \dots, \underbrace{(v'_{i,1}, \mathsf{op}'_{i,1}), (v'_{i,2}, \mathsf{op}'_{i,2}), \dots, (v'_{i,4}, \mathsf{op}'_{i,4})}_{\text{corresponding to } (v_{i-1}, \mathsf{op}_i)}, \dots \\ & \dots, \underbrace{(v'_{k,1}, \mathsf{op}'_{k,1}), (v'_{k,2}, \mathsf{op}'_{k,2}), \dots, (v'_{k,4}, \mathsf{op}'_{k,4})}_{\text{corresponding to } (v_{k-1}, \mathsf{op}_k)} . \end{aligned}$$

The first n steps are called the *commit stage* while the remaining steps are called the *evaluation stage*.

The commit stage proceeds as follows: For $1 \le i \le n$, a pair of a visible sequence and an operation $(v'_{0,i}, \mathsf{op}'_{0,i})$ is defined as follows:

$$(v'_{0,i}, \mathsf{op}'_{0,i}) = \big((v_{i-1}, \widetilde{?^{\ell}}), (\mathsf{privatePerm}, i, \mathsf{et}((\mathrm{l}_i) \leftrightarrow (\mathrm{r}_i)))\big).$$

After the i-th step, a pair of cards on position $(\mathrm{l}_i, \mathrm{r}_i)$ consists of a commitment to x_i. Thus, after the commit stage, we obtain the sequence as follows:



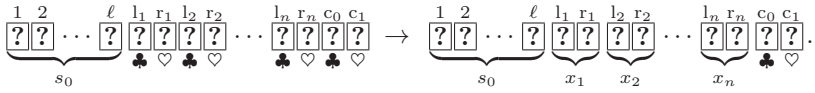
The evaluation stage proceeds as follows: Let $1 \le i \le k$. When op_i is either a permutation or a turning, four pairs $(v'_{i,j}, \mathsf{op}'_{i,j})$ for $j \in [4]$ are defined as follows:

- $(v'_{i,1}, \mathsf{op}'_{i,1}) = \big((v_{i-1}, ?^m), \mathsf{op}_i\big)$.
- $(v'_{i,2}, \mathsf{op}'_{i,2}) = \big((v_i, ?^m), (\mathsf{perm}, \mathsf{id})\big)$.
- $(v'_{i,3}, \mathsf{op}'_{i,3}) = \big((v_i, ?^m), (\mathsf{perm}, \mathsf{id})\big)$.
- $(v'_{i,4}, \mathsf{op}'_{i,4}) = \big((v_i, ?^m), (\mathsf{perm}, \mathsf{id})\big)$.

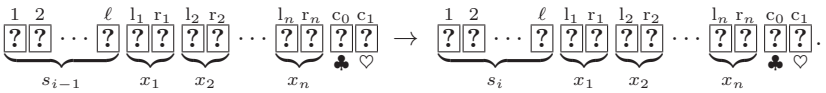
Here, $\mathsf{op}_i \in \mathcal{O}^{\mathsf{p}}_{n,\ell}$ is converted into $\mathsf{op}'_{i,1} \in \mathcal{O}^{\mathsf{p}}_{n,\ell+m}$ in a canonical way. That is, every $(\mathsf{perm}, \pi \in S_\ell)$ is converted into $(\mathsf{perm}, \pi' \in S_{\ell+m})$ such that $\pi'(x) = \pi(x)$ for all $1 \le x \le \ell$ and $\pi'(x) = x$ for all $\ell+1 \le x \le \ell+m$, and every $(\mathsf{turn}, T \subset [\ell])$ is converted into $(\mathsf{turn}, T' \subset [\ell + m])$ with $T' = T$. When op_i is a private permutation $\mathsf{op}_i = (\mathsf{privatePerm}, \iota, \mathsf{et}(T_0 \leftrightarrow T_1))$, four pairs $(v'_{i,j}, \mathsf{op}'_{i,j})$ for $j \in [4]$ are defined as follows:

- $(v'_{i,1}, \mathsf{op}'_{i,1}) = \big((v_{i-1}, \mathtt{?}^m), (\mathsf{privatePerm}, \iota, \mathsf{et}((T_0, \mathrm{l}_i, \mathrm{c}_0) \leftrightarrow (T_1, \mathrm{r}_i, \mathrm{c}_1)))\big)$.
- $(v'_{i,2}, \mathsf{op}'_{i,2}) = \big((v_i, \mathtt{?}^m), (\mathsf{turn}, \{\mathrm{l}_i\})\big)$.
- $(v'_{i,3}, \mathsf{op}'_{i,3}) = \big((v_i, \mathtt{?}^{2i-2}\clubsuit\mathtt{?}^{2n-2i+3}), (\mathsf{turn}, \{\mathrm{l}_i\})\big)$.
- $(v'_{i,4}, \mathsf{op}'_{i,4}) = \big((v_i, \mathtt{?}^m), (\mathsf{perm}, (\mathrm{l}_i\, \mathrm{c}_0) \circ (\mathrm{r}_i\, \mathrm{c}_1))\big)$.

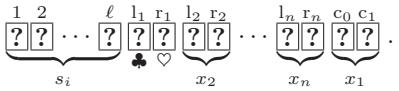
Correctness. After the commit stage, the initial sequence is changed to the sequence as follows:



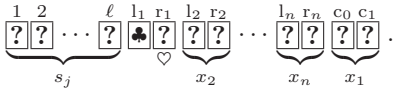
To prove the correctness, it is sufficient to show that in the evaluation stage, each step of $\mathcal{P}$ is correctly simulated as follows:



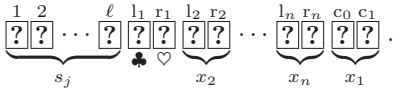
Since it is trivial when op_i is either a permutation or a turning, we consider the case when op_i is a private permutation $\mathsf{op}_i = (\mathsf{privatePerm}, \iota, \mathsf{et}(T_0 \leftrightarrow T_1))$. Without loss of generality, we can assume $i = 1$. After applying the private permutation $\mathsf{op}'_{i,1} = (\mathsf{privatePerm}, \iota, \mathsf{et}((T_0, \mathrm{l}_i, \mathrm{c}_0) \leftrightarrow (T_1, \mathrm{r}_i, \mathrm{c}_1)))$, the current sequence is as follows:



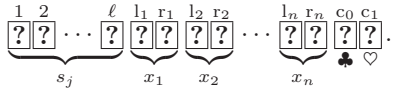
After applying $\mathsf{op}'_{i,2} = (\mathsf{turn}, \{\mathrm{l}_1\})$, the current sequence is as follows:



Note that the visible sequence of the above sequence is matched with $v'_{i,3} = (v_i, \mathtt{?}^{2i-2}\clubsuit\mathtt{?}^{2n-2i+3})$. After applying $\mathsf{op}'_{i,3} = (\mathsf{turn}, \{\mathrm{l}_1\})$, the current sequence is as follows:



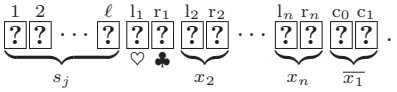
After applying $\mathsf{op}'_{i,4} = (\mathsf{perm}, \mathsf{et}((\mathrm{l}_1, \mathrm{r}_1) \leftrightarrow (\mathrm{c}_0, \mathrm{c}_1)))$, the current sequence is as follows:



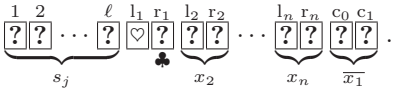
Thus, $\mathcal{P}'$ correctly realizes the functionality $\mathcal{F}'$.

Active security in the envelope model. We show that if $\mathcal{P}$ securely realizes $\mathcal{F}$ with passive security, $\mathcal{P}'$ securely realizes $\mathcal{F}' = (s'_{\mathsf{in}}, s'_{\mathsf{out}})$ with active security in the envelope model. We can observe that out-of-range attacks do not happen since every private permutation is an extended transposition. Thus, it is sufficient to show that any inconsistent attack in the evaluation stage can be immediately detected by the successive turning operation.

Now suppose that the first player acts maliciously in the private permutation $(\mathsf{privatePerm}, 1, \mathsf{et}((T_0, \mathrm{l}_1, \mathrm{c}_0) \leftrightarrow (T_1, \mathrm{r}_1, \mathrm{c}_1)))$. Then, the current sequence is as follows:



After applying $(\mathsf{turn}, \{\mathrm{l}_1\})$, the current sequence is as follows:



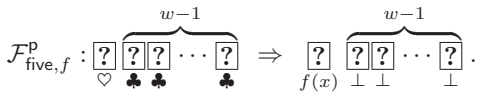
Thus, the protocol terminates since the action function A' outputs $\perp$. Since the view depends only the first malicious points of the execution only, for any execution permutation lists $\mathcal{E}, \mathcal{E}'$ in the envelope model, it holds that $\mathsf{view}_{\mathcal{P}'}(s'_{\mathsf{in}}, \mathcal{E}) = \mathsf{view}_{\mathcal{P}'}(s'_{\mathsf{in}}, \mathcal{E}')$ if and only if the first malicious points of $\mathcal{E}, \mathcal{E}'$ are the same. Thus, $\mathcal{P}'$ securely realizes $\mathcal{F}'$ with active security in the envelope model.

4.5 Card-efficient protocol in the envelope model

4.5.1 Passive protocol based on branching program

Let B be any branching program of width w and length k, which computes a function $f : \{0,1\}^n \to \{0,1\}$. We design a protocol with passive security using w cards and k private permutations.

Functionality. A functionality $\mathcal{F}^{\mathsf{p}}_{\mathsf{five},f}$ is defined by:



Protocol. A protocol $\mathcal{P}^{\mathsf{p}}_{\mathsf{five},f}$ is defined by:

$$\mathcal{P}^{\mathsf{p}}_{\mathsf{five},f} = (n, \{0,1\}, (\mathsf{Binary}, \{(\clubsuit/\mathbf{?})^{w-1}, \heartsuit/\mathbf{?}\}, \mathcal{O}^{\mathsf{p}}_{n,w}, A).$$

The action function A is as follows:

$$A = ((v, \mathsf{op}_1), (v, \mathsf{op}_2), (v, \mathsf{op}_3), \ldots, (v, \mathsf{op}_{2k})),$$

where $v = \mathbf{?}^w$ and $\mathsf{op}_{2i-1}, \mathsf{op}_{2i}$ $(1 \le i \le k)$ are defined by: for the i-th operation $\langle j, \pi_0, \pi_1 \rangle$ of B,

- $\mathsf{op}_{2i-1} = (\mathsf{privatePerm}, j, (\pi_0)^{-1} \circ \pi_1)$;
- $\mathsf{op}_{2i} = (\mathsf{perm}, \pi_0)$.

Correctness. If $x_i = 0$, the applied permutation is π_0. If $x_i = 1$, the applied permutation is $\pi_0 \circ ((\pi_0)^{-1} \circ \pi_1) = \pi_1$. Thus, the above protocol simulates the branching program B. Therefore, the above protocol correctly realizes the functionality so as B computes f.

Passive security. This is trivial since all sequences are always face-down.

4.5.2 Active protocol using $2n + 7$ cards

Theorem 4.1. Let $f : \{0,1\}^n \to \{0,1\}$ be any function. There exists a protocol using $2n + 7$ cards that correctly and securely realizes a functionality $\mathcal{F}$ with active security defined as follows:

$$\mathcal{F} : \underbrace{?\,?\,?\,?\,?}_{\heartsuit\,\clubsuit\,\clubsuit\,\clubsuit\,\clubsuit}\ \underbrace{?\,?\,?\,?\,\cdots\,?\,?}_{\clubsuit\,\heartsuit\,\clubsuit\,\heartsuit\ \cdots\ \clubsuit\,\heartsuit}\ \Rightarrow\ \underbrace{?}_{f(x)}\ \underbrace{?\,?\,?\,?}_{\perp\,\perp\,\perp\,\perp}\ \underbrace{?\,?}_{x_1}\ \underbrace{?\,?}_{x_2}\ \cdots\ \underbrace{?\,?}_{x_n}\ \underbrace{?\,?}_{\clubsuit\,\heartsuit}.$$

The deck of the protocol is $\overline{\mathcal{D}} = (\mathsf{Binary}, \{(\clubsuit/?)^{n+5}, (\heartsuit/?)^{n+1}\})$. ■

Proof. From Barrington's Theorem (Lemma 2.1), for any function $f : \{0,1\}^n \to \{0,1\}$, there exists a branching program B of width 5 and length at most 4^d, where d is the depth of a circuit computing f. Thus, we have a protocol with passive security using 5 cards. By applying our protocol compiler, we obtain a protocol with active security using $2n + 7$ cards.

4.6 Efficient protocols for specific functions

4.6.1 Passive protocol for symmetric functions

Let $w(x) \in \{0, 1, \ldots, n\}$ be the Hamming weight of $x \in \{0,1\}^n$, that is, for $x = x_1 x_2 \cdots x_n \in \{0,1\}^n$, $w(x) = \sum_{i=1}^n x_i$. We say that a function $f : \{0,1\}^n \to \{0,1\}$ is *symmetric* if $f(x) = f(x')$ for any $x, x' \in \{0,1\}^n$ such that $w(x) = w(x')$. We construct two protocols for computing any symmetric function. The first protocol $\mathcal{P}^{\mathsf{p}}_{\mathsf{sym1},f}$ is passively secure and the second protocol $\mathcal{P}^{\mathsf{p}}_{\mathsf{sym2},f}$ is actively secure in the ring-with-envelope model.

Functionality $\mathcal{F}^{\mathsf{p}}_{\mathsf{sym1},f}$. A functionality $\mathcal{F}^{\mathsf{p}}_{\mathsf{sym1},f}$ is defined as follows:

$$\mathcal{F}^{\mathsf{p}}_{\mathsf{sym1},f} = \underbrace{?\,?\,\cdots\,?}_{z_0\ z_1\ \cdots\ z_n}\ \Rightarrow\ \underbrace{?}_{f(x)}\ \overbrace{\underbrace{?\,?\,\cdots\,?}_{\perp\,\perp\ \cdots\ \perp}}^{n},$$

where z_i is the output of $f(x)$ with $w(x) = i$.

Protocol $\mathcal{P}^{\mathsf{p}}_{\mathsf{sym1},f}$. A protocol $\mathcal{P}^{\mathsf{p}}_{\mathsf{sym1},f}$ is defined by:

$$\mathcal{P}^{\mathsf{p}}_{\mathsf{sym1},f} = (n, \{0,1\}, (\mathsf{Binary}, \{(\clubsuit/\mathbf{?})^a, (\heartsuit/\mathbf{?})^{n+1-a}\}, \mathcal{O}^{\mathsf{p}}_{n,n+1}, A),$$

where $a := |\{x \mid x \in \{0,1\}^n, f(x) = 0\}|$. The action function A is as follows:

$$A = ((v, \mathsf{op}_1), (v, \mathsf{op}_2), (v, \mathsf{op}_3), \ldots, (v, \mathsf{op}_n)),$$

where $v = \mathbf{?}^{n+1}$ and op_i $(1 \le i \le n)$ is defined as follows:

$$\mathsf{op}_i = (\mathsf{privatePerm}, i, \sigma),$$

for $\sigma = (n+1 \ \cdots \ 3\ 2\ 1)$.

Correctness of $\mathcal{P}^{\mathsf{p}}_{\mathsf{sym1},f}$. When $w(x) = i$, a permutation σ^i is totally applied to the initial sequence. Thus, the first card of the final sequence is z_i which is the output value $f(x)$. Therefore, $\mathcal{P}^{\mathsf{p}}_{\mathsf{sym1},f}$ correctly realizes the functionality $\mathcal{F}^{\mathsf{p}}_{\mathsf{sym1},f}$.

Passive security of $\mathcal{P}^{\mathsf{p}}_{\mathsf{sym1},f}$. It is trivial since all sequences are always face-down.

4.6.2 Active protocol for symmetric functions

Functionality $\mathcal{F}^{\mathsf{p}}_{\mathsf{sym2},f}$. A functionality $\mathcal{F}^{\mathsf{p}}_{\mathsf{sym2},f}$ is defined as follows:

$$\mathcal{F}^{\mathsf{p}}_{\mathsf{sym2},f} = \underset{z_0}{\boxed{?}}\underset{z_1}{\boxed{?}} \cdots \underset{z_n}{\boxed{?}}\overbrace{\underset{\clubsuit}{\boxed{?}}\underset{\clubsuit}{\boxed{?}} \cdots \underset{\clubsuit}{\boxed{?}}}^{n}\underset{\heartsuit}{\boxed{?}} \Rightarrow \underset{f(x)}{\boxed{?}}\overbrace{\underset{\perp}{\boxed{?}}\underset{\perp}{\boxed{?}} \cdots \underset{\perp}{\boxed{?}}}^{n}\overbrace{\underset{\clubsuit}{\boxed{?}}\underset{\clubsuit}{\boxed{?}} \cdots \underset{\clubsuit}{\boxed{?}}}^{n}\underset{\heartsuit}{\boxed{?}}.$$

Protocol $\mathcal{P}^{\mathsf{p}}_{\mathsf{sym2},f}$. A protocol $\mathcal{P}^{\mathsf{p}}_{\mathsf{sym2},f}$ is defined as follows:

$$\mathcal{P}^{\mathsf{p}}_{\mathsf{sym2},f} = (n, \{0,1\}, (\mathsf{Binary}, \{(\clubsuit/\mathbf{?})^{a+n}, (\heartsuit/\mathbf{?})^{n+2-a}\}, \mathcal{O}^{\mathsf{p}}_{n,2n+2}, A),$$

where $a := |\{x \mid x \in \{0,1\}^n, f(x) = 0\}|$. For ease of explanation, we use notations as follows:

$$\overset{1}{\underset{z_0}{\boxed{?}}}\overset{2}{\underset{z_1}{\boxed{?}}}\overset{3}{\underset{z_2}{\boxed{?}}}\overset{4}{\underset{z_3}{\boxed{?}}} \cdots \overset{n+1}{\underset{z_n}{\boxed{?}}}\overset{p_0}{\underset{\clubsuit}{\boxed{?}}}\overset{p_1}{\underset{\clubsuit}{\boxed{?}}}\overset{p_2}{\underset{\clubsuit}{\boxed{?}}}\overset{p_3}{\underset{\clubsuit}{\boxed{?}}} \cdots \overset{p_n}{\underset{\heartsuit}{\boxed{?}}},$$

where $p_i = n + 2 + i$ denotes a position. The action function A is as follows:

$$A = ((v, \mathsf{op}_1), (v, \mathsf{op}_2), (v, \mathsf{op}_3), \ldots, (v, \mathsf{op}_{4n})),$$

where v_{4i-j} and op_{4i-j} $(1 \le i \le n$ and $1 \le j \le 4)$ is defined as follows:

- $(v_{4i-3}, \mathsf{op}_{4i-3}) = \big(\mathbf{?}^{2n+2}, (\mathsf{privatePerm}, i, (p_0\ p_n))\big)$;
- $(v_{4i-2}, \mathsf{op}_{4i-2}) = \big(\mathbf{?}^{2n+2}, (\mathsf{privatePerm}, i, \sigma\tau)\big)$ for $\sigma = (n+1 \ \cdots \ 3\ 2\ 1)$ and $\tau = (p_n \ \cdots \ p_2\ p_1\ p_0)$;
- $(v_{4i-1}, \mathsf{op}_{4i-1}) = \big(\mathbf{?}^{2n+2}, (\mathsf{turn}, \{p_n\})\big)$;
- $(v_{4i}, \mathsf{op}_{4i}) = \big(\mathbf{?}^{2n+1}\heartsuit, (\mathsf{turn}, \{p_n\})\big)$.

Correctness. Since the leftmost $n+1$ cards are always equivalent to those of $\mathcal{P}^{\mathsf{p}}_{\mathsf{sym1},f}$, the leftmost card in the final sequence is $f(x) = z_i$ with $w(x) = i$. In addition, after the operation op_{4i} $(1 \le i \le n)$, the rightmost $n+1$ cards are equivalent to as follows:

$$\overbrace{\boxed{?}\boxed{?}\cdots\boxed{?}}^{n}\boxed{?}.$$
$$\clubsuit\ \clubsuit\qquad\clubsuit\ \heartsuit$$

Therefore, $\mathcal{P}^{\mathsf{p}}_{\mathsf{sym2},f}$ correctly realizes the functionality $\mathcal{F}^{\mathsf{p}}_{\mathsf{sym2},f}$.

Active security in the ring-with-envelope model. Let $\mathcal{P} = \mathcal{P}^{\mathsf{p}}_{\mathsf{sym2},f}$. Let $\pi_1 = (p_0\ p_n)$ and $\pi_2 = \sigma\tau$. Any executed permutation list $\mathcal{E}$ in the ring-with-envelope model is written by the following:

$$\mathcal{E} = (\pi_1^{b_1}, \pi_2^{c_1}, \pi_1^{b_2}, \pi_2^{c_2}, \ldots, \pi_1^{b_n}, \pi_2^{c_n}),$$

where $b_i \in \{0,1\}$ and $0 \le c_i \le n$ for all $i \in [n]$. When $\mathcal{E}$ is an honest execution, $b_i = c_i$ for all $i \in [n]$ and the view $\mathsf{view}_{\mathcal{P}}(s_0, \mathcal{E})$ is a sequence as follows:

$$\mathsf{view}_{\mathcal{P}}(s_0, \mathcal{E}) = \big((?^{2n+2} \to ?^{2n+2} \to ?^{2n+2} \to ?^{2n+1}\heartsuit)^n \to ?^{2n+2}\big).$$

When $\mathcal{E}$ is a malicious execution whose malicious point is $1 \le j \le 2n$, the view $\mathsf{view}_{\mathcal{P}}(s_0, \mathcal{E})$ is a sequence as follows:

$$\begin{aligned}\mathsf{view}_{\mathcal{P}}(s_0, \mathcal{E}) = \big((?^{2n+2} \to ?^{2n+2} \to ?^{2n+2} \to ?^{2n+1}\heartsuit)^{j/2-1} \\ \to ?^{2n+2} \to ?^{2n+2} \to ?^{2n+2} \to ?^{2n+1}\clubsuit\big).\end{aligned}$$

Note that j is an even number since every private permutation of π_1 is the first call of the input bit. Thus, it holds $\mathsf{view}_{\mathcal{P}}(\mathcal{E}) = \mathsf{view}_{\mathcal{P}}(\mathcal{E}')$ for any executed permutation lists $\mathcal{E}, \mathcal{E}'$ whose malicious points are the same. Therefore, $\mathcal{P}$ securely realizes $\mathcal{F}^{\mathsf{p}}_{\mathsf{sym2},f}$ with active security in the ring-with-envelope model.

4.6.3 Active protocol for the AND function

Functionality. A functionality $\mathcal{F}^{\mathsf{p}}_{\mathsf{AND}}$ is defined as follows:

$$\mathcal{F}^{\mathsf{p}}_{\mathsf{AND}} : \overbrace{\boxed{?}\boxed{?}\cdots\boxed{?}}^{n}\boxed{?} \Rightarrow \underset{f(x)}{\boxed{?}}\ \overbrace{\boxed{?}\boxed{?}\cdots\boxed{?}}^{n},$$
$$\clubsuit\ \clubsuit\qquad\clubsuit\ \heartsuit\qquad\qquad\bot\ \bot\qquad\bot$$

Protocol. A protocol $\mathcal{P}^{\mathsf{p}}_{\mathsf{AND}}$ is defined as follows:

$$\mathcal{P}^{\mathsf{p}}_{\mathsf{AND}} = (n, \{0,1\}, (\mathsf{Binary}, \{(\clubsuit/?)^n, \heartsuit/?\}, \mathcal{O}^{\mathsf{p}}_{n,n+1}, A).$$

The action function A is as follows:

$$A = ((v, \mathsf{op}_1), (v, \mathsf{op}_2), (v, \mathsf{op}_3), \ldots, (v, \mathsf{op}_n)),$$

where $v = ?^{n+1}$ and op_i $(1 \le i \le n)$ is defined as follows:

$$\mathsf{op}_i = (\mathsf{privatePerm}, i, (n+1-i\ \ n+2-i)).$$

Correctness. After applying op_1 to the initial sequence $\clubsuit^n\heartsuit$, the rightmost card and the next rightmost card are swapped if $x_1 = 1$ as follows:

$$\clubsuit^n\heartsuit \to \begin{cases} \clubsuit^{n-1}\heartsuit\clubsuit & \text{if } x_1 = 1 \\ \clubsuit^{n-1}\clubsuit\heartsuit & \text{otherwise.} \end{cases}$$

Thus, if $x_1 = 1$, the position of $\heartsuit$ is moved to the left. We can observe that the rightmost $\heartsuit$ in the initial sequence is moved to the leftmost in the final sequence if and only if $x_1 = x_2 = \cdots = x_n = 1$. Thus, the rightmost card is $\heartsuit$ if and only if $x_1 \wedge x_2 \wedge \cdots \wedge x_n = 1$. Therefore, $\mathcal{P}^{\mathsf{p}}_{\mathsf{AND}}$ correctly realizes the functionality $\mathcal{F}^{\mathsf{p}}_{\mathsf{AND}}$.

Active security in the bare-bone model. Since every private permutation is a transposition, out-of-range attacks do not happen. In addition, every input bit is called at once, inconsistent attacks do not happen. Thus, $\mathcal{P}^{\mathsf{p}}_{\mathsf{AND}}$ securely realizes $\mathcal{F}^{\mathsf{p}}_{\mathsf{AND}}$ with active security in the bare-bone model.

4.6.4 Active protocol for equality of two strings

Functionality. Let $x, y \in \{0,1\}^n$. A functionality $\mathcal{F}^{\mathsf{p}}_{\mathsf{EQ}}$ is defined as follows:

$$\mathcal{F}^{\mathsf{p}}_{\mathsf{EQ}} : \underset{\heartsuit}{\boxed{?}}\overbrace{\underset{\clubsuit}{\boxed{?}}\underset{\clubsuit}{\boxed{?}}\cdots\underset{\clubsuit}{\boxed{?}}}^{n} \Rightarrow \underset{\mathsf{p}(x=y)}{\boxed{?}}\overbrace{\underset{\bot}{\boxed{?}}\underset{\bot}{\boxed{?}}\cdots\underset{\bot}{\boxed{?}}}^{n}.$$

Protocol. A protocol $\mathcal{P}^{\mathsf{p}}_{\mathsf{EQ}}$ is defined as follows:

$$\mathcal{P}^{\mathsf{p}}_{\mathsf{EQ}} = (2n, \{0,1\}, (\mathsf{Binary}, \{(\clubsuit/?)^n, \heartsuit/?\}, \mathcal{O}^{\mathsf{p}}_{2n,n+1}, A).$$

(We refer the former n bits as $x \in \{0,1\}^n$ and the latter n bits as $y \in \{0,1\}^n$.) The action function A is as follows:

$$A = ((v, \mathsf{op}_1), (v, \mathsf{op}_2), (v, \mathsf{op}_3), \ldots, (v, \mathsf{op}_{2n})),$$

where $v = ?^{n+1}$ and $(\mathsf{op}_{2i-1}, \mathsf{op}_{2i})$ $(1 \le i \le n)$ is defined as follows:

- $\mathsf{op}_{2i-1} = (\mathsf{privatePerm}, i, (1 \;\; i+1))$;
- $\mathsf{op}_{2i} = (\mathsf{privatePerm}, i+n, (1 \;\; i+1))$.

Correctness. After applying op_{2i-1} and op_{2i}, the first card and the i-th card are swapped if and only if $x_i \neq y_i$. We can observe that the leftmost card in the final sequence is $\heartsuit$ if and only if $(x_1 = y_1) \wedge (x_2 = y_2) \wedge \cdots \wedge (x_n = y_n)$. Therefore, $\mathcal{P}^{\mathsf{p}}_{\mathsf{EQ}}$ correctly realizes the functionality $\mathcal{F}^{\mathsf{p}}_{\mathsf{EQ}}$.

Active security in the bare-bone model. Since every private permutation is a transposition, out-of-range attacks do not happen. In addition, every input bit is called at once, inconsistent attacks do not happen. Thus, $\mathcal{P}^{\mathsf{p}}_{\mathsf{EQ}}$ securely realizes $\mathcal{F}^{\mathsf{p}}_{\mathsf{EQ}}$ with active security in the bare-bone model.

Chapter 5

Protocols Based on Polygon-Shaped Cards

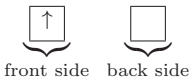
5.1 Protocols based on cyclic cards

5.1.1 Cyclic cards

Let $m \geq 2$ be any integer. A *cyclic card of modulus m* is a card as follows:

- One side (referred as *back side*) has a $(360/m)^\circ$ rotational symmetry;
- The other side (referred as *front side*) has no rotational symmetry.

For example, a cyclic card of modulus 4 is implemented as follows:



The advantage of cyclic cards is that a single card can have a non-binary value $x \in \mathbb{Z}_m$ as follows:

$$\boxed{\uparrow} = 0,\ \boxed{\leftarrow} = 1,\ \boxed{\downarrow} = 2,\ \boxed{\rightarrow} = 3.$$

A card specification of cyclic cards. For $x \in \mathbb{Z}_m$, we denote a face-up card having x by $\underline{x}$ and a face-down card having x by $[\![x]\!]$. The card set of cyclic cards of modulus m, denoted by $\mathcal{C}_m^{\mathsf{c}}$, is defined as follows:

$$\mathcal{C}_m^{\mathsf{c}} = \{\underline{0}, \underline{1}, \ldots, \underline{m-1}, [\![0]\!], [\![1]\!], \ldots, [\![m-1]\!]\}.$$

For a card $c \in \mathcal{C}_m^{\mathsf{c}}$, we define two types of transformations: *rotation* and *turning*. For any $j \in \mathbb{Z}_m$, a *rotation operation with a degree j* is defined as follows:

$$\mathsf{rot}^j(c) = \begin{cases} \underline{i+j} & \text{if } c = \underline{i} \text{ for some } i \in \mathbb{Z}_m \\ [\![i-j]\!] & \text{if } c = [\![i]\!] \text{ for some } i \in \mathbb{Z}_m \end{cases}$$

Physically, this is a rotation with $(360/m)^\circ$. Note that a face-down card $[\![i]\!]$ is transformed into a face-down card $[\![i-j]\!]$ since a rotation of face-down cards is a backward rotation of face-up cards. A *turning operation* is defined as follows:

$$\mathsf{turn}(c) = \begin{cases} [\![i]\!] & \text{if } c = \underline{i} \text{ for some } i \in \mathbb{Z}_m \\ \underline{i} & \text{if } c = [\![i]\!] \text{ for some } i \in \mathbb{Z}_m \end{cases}$$

The transformation set of cyclic cards of modulus m, denoted by $\mathcal{T}^{\mathsf{c}}_m$, is defined as follows:

$$\mathcal{T}^{\mathsf{c}}_m = \{\mathsf{id}, \mathsf{rot}, \mathsf{rot}^2, \ldots, \mathsf{rot}^{m-1}, \mathsf{turn}\}.$$

The symbol set of cyclic cards of modulus m, denoted by Σ^{c}_m, is defined as follows:

$$\Sigma^{\mathsf{c}}_m = \{0, 1, 2, \cdots, m-1, \mathsf{?}\}.$$

The vision function $\mathsf{vis}^{\mathsf{c}}_m : \mathcal{C}^{\mathsf{c}}_m \to \Sigma^{\mathsf{c}}_m$ of cyclic cards of modulus m is defined as follows:

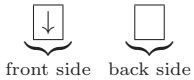
$$\mathsf{vis}^{\mathsf{c}}_m(c) = \begin{cases} i & \text{if } c = \underline{i} \text{ for } 0 \le i \le m-1 \\ \mathsf{?} & \text{otherwise.} \end{cases}$$

A card specification of cyclic cards of modulus m, denoted by Cyclic_m, is defined as follows:

$$\mathsf{Cyclic}_m = (\mathcal{C}^{\mathsf{c}}_m, \mathcal{T}^{\mathsf{c}}_m, \Sigma^{\mathsf{c}}_m, \mathsf{vis}^{\mathsf{c}}_m).$$

Commitment. A *commitment to* $x \in \mathbb{Z}_m$ is defined by $\mathsf{com}(x) = [\![x]\!]$.

Existing cards with a 180° **rotationally symmetric back.** Mizuki and Shizuya [32] proposed a card with a 180° rotationally symmetric back as follows:



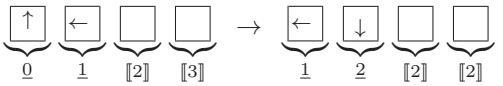
It can be regarded as a cyclic card with modulus 2. Thus, a deck of cyclic cards is considered to be a natural generalization of Mizuki and Shizuya's cards.

5.1.2 Operations for cyclic cards

Rotation. For $T \subset [\ell]$ and $a \in \mathbb{Z}_m$, a *rotation operation* is defined as follows:

$$(\mathsf{rot}, T, a).$$

For a sequence $s = (c_1, c_2, \ldots, c_\ell) \in \mathsf{Seq}^{\overline{\mathcal{D}}}$, by applying a rotation operation (rot, T, a), it is transformed into a new sequence $s' = (c'_1, c'_2, \ldots, c'_\ell) \in \mathsf{Seq}^{\overline{\mathcal{D}}}$ such that $c'_i = \mathsf{rot}^a(c_i)$ for all $i \in T$ and $c'_i = c_i$ for all $i \notin T$. For example, for a sequence $s = (\underline{0}, \underline{1}, [\![2]\!], [\![3]\!])$ with modulus $m = 4$, a rotation operation $(\mathsf{rot}, \{1, 2, 4\}, 1)$ transforms it into a new sequence $s' = (\underline{1}, \underline{2}, [\![2]\!], [\![2]\!])$ as follows:



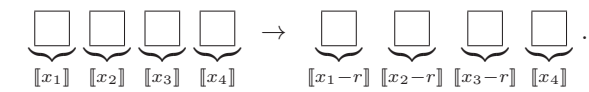
The set of rotations $\mathsf{Rot}_{m,\ell}$ is defined as follows:

$$\mathsf{Rot}_{m,\ell} = \{(\mathsf{rot}, T, a) \mid T \subset [\ell], a \in \mathbb{Z}_m\}.$$

Rotation shuffle. For $T \subset [\ell]$, a *rotation shuffle* is defined as follows:

$$(\mathsf{rotshuf}, T).$$

For all $i \in T$, the i-th card in the sequence is rotated with a degree $r \in \mathbb{Z}_m$, here r is uniformly and randomly chosen from $\mathbb{Z}_m$ and this r is common for all $i \in T$. Other cards are unchanged. For example, for a sequence $(\llbracket x_1 \rrbracket, \llbracket x_2 \rrbracket, \llbracket x_3 \rrbracket, \llbracket x_4 \rrbracket)$ with modulus $m = 4$, a rotation shuffle $(\mathsf{rotshuf}, \{1, 2, 3\})$ generates a sequence $(\llbracket x_1 - r \rrbracket, \llbracket x_2 - r \rrbracket, \llbracket x_3 - r \rrbracket, \llbracket x_4 \rrbracket)$ for a random $r \in \mathbb{Z}/4\mathbb{Z}$ as follows:



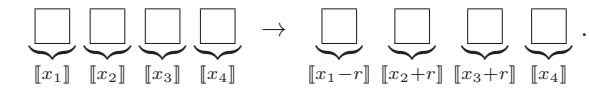
The set of rotation shuffles is defined as follows:

$$\mathsf{RotShuf}_{m,\ell} = \{(\mathsf{rotshuf}, T) \mid T \subset [\ell]\}.$$

Backward rotation shuffle. For $T_1, T_2 \subset [\ell]$ such that $T_1 \cap T_2 = \emptyset$, a *backward rotation shuffle* is defined as follows:

$$(\mathsf{backrot}, T_1, T_2).$$

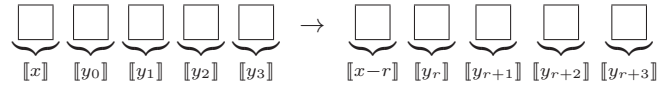
By applying this operation, each card in T_1 is rotated with a degree $r \in \mathbb{Z}_m$ while each card in T_2 is rotated with a degree $-r \in \mathbb{Z}_m$. Other cards are unchanged. For example, for a sequence $(\llbracket x_1 \rrbracket, \llbracket x_2 \rrbracket, \llbracket x_3 \rrbracket, \llbracket x_4 \rrbracket)$, a backward rotation shuffle $(\mathsf{backrot}, \{1\}, \{2, 3\})$ generates a sequence $(\llbracket x_1 - r \rrbracket, \llbracket x_2 + r \rrbracket, \llbracket x_3 + r \rrbracket, \llbracket x_4 \rrbracket)$ for a random $r \in \mathbb{Z}_m$ as follows:



The set of backward rotation shuffles is defined as follows:

$$\mathsf{BackRot}_{m,\ell} = \{(\mathsf{backrot}, T) \mid T \subset [\ell]\}.$$

Flower shuffle. A *flower shuffle* is a random rotation with a cyclic permutation. In the case of $m = 4$, it takes a sequence of five cards $(\llbracket x \rrbracket, \llbracket y_0 \rrbracket, \llbracket y_1 \rrbracket, \llbracket y_2 \rrbracket, \llbracket y_3 \rrbracket)$ and outputs $(\llbracket x - r \rrbracket, \llbracket y_r \rrbracket, \llbracket y_{r+1} \rrbracket, \llbracket y_{r+2} \rrbracket, \llbracket y_{r+3} \rrbracket)$ as follows:



That is, the leftmost card $\llbracket x \rrbracket$ is rotationally randomized and the other four cards $(\llbracket y_0 \rrbracket, \llbracket y_1 \rrbracket, \llbracket y_2 \rrbracket, \llbracket y_3 \rrbracket)$ are cyclically and randomly shifted. $\llbracket x \rrbracket$ is called the

center of the flower shuffle and $[\![y_i]\!]$ is called the i-th petal of the flower shuffle. The center and a petal can be take multiple cards. For example, a flower shuffle takes $([\![x]\!], [\![x']\!], [\![y_0]\!], [\![y'_0]\!], [\![y_1]\!], [\![y'_1]\!], [\![y_2]\!], [\![y'_2]\!], [\![y_3]\!], [\![y'_3]\!])$ as input and outputs

$$([\![x-r]\!], [\![x'-r]\!], [\![y_r]\!], [\![y'_r]\!], [\![y_{r+1}]\!], [\![y'_{r+1}]\!], [\![y_{r+2}]\!], [\![y'_{r+2}]\!], [\![y_{r+3}]\!], [\![y'_{r+3}]\!]).$$

In this case, the center consists of two cards and each petal consists of two cards. Formally, it is defined as follows:

$$(\mathsf{flower}, C, T_0, T_1, \cdots, T_{m-1}),$$

where $C, T_0, T_1, \cdots, T_{m-1} \subset [\ell]$ are pairwise disjoint subsets of positions such that $|T_0| = |T_1| = \cdots = |T_{m-1}| = k$ for some integer k. Here, T_i is an ordered list $T_i = (t_{i,1}, t_{i,2}, \cdots, t_{i,k})$. By this operation, each card in C is rotated with a degree r while each card $t_{i,j}$, which is the j-th card in T_i, is moved to the position of the card $t_{i+r,j}$, which is the j-th card in T_{i+r}. Other cards are unchanged. The set of backward rotation shuffles is defined as follows:

$$\mathsf{Flower}_{m,\ell} = \{(\mathsf{flower}, C, T_0, T_1, \cdots, T_{m-1}) \mid C, T_0, T_1, \cdots, T_{m-1} \subset [\ell]\}.$$

5.1.3 Notations

Hereafter, we use notations as follows.

Operations. The set of permutations, turnings, and shuffles for cyclic cards of modulus m is similarly defined as that for binary cards, and denoted by $\mathsf{Perm}_{m,\ell}$, $\mathsf{Turn}_{m,\ell}$, and $\mathsf{Shuf}_{m,\ell}$, respectively, where ℓ is the number of cards. We assume that the set of operations is $\mathcal{O}^{\mathsf{c}}_{m,\ell}$ defined as follows:

$$\mathcal{O}^{\mathsf{c}}_{m,\ell} = \mathsf{Perm}_{m,\ell} \cup \mathsf{Turn}_{m,\ell} \cup \mathsf{Shuf}_{m,\ell} \cup \mathsf{Rot}_{m,\ell} \cup \mathsf{RotShuf}_{m,\ell} \cup \mathsf{BackRot}_{m,\ell} \cup \mathsf{Flower}_{m,\ell}.$$

View. We omit revealed information in a view since all operations in $\mathcal{O}^{\mathsf{c}}_{m,\ell}$ never produce revealed information.

5.1.4 Outline of protocols

Our protocols based on cyclic cards in this chapter are as follows:

- Subtraction protocol (Section 5.1.5): It takes two cards $([\![x_1]\!], [\![x_2]\!])$ as input and outputs a card $[\![x_2 - x_1]\!]$.
- Addition protocol #1 (Section 5.1.6): It takes three cards $([\![x_1]\!], [\![x_2]\!], [\![0]\!])$ as input and outputs a card $[\![x_1 + x_2]\!]$. It calls the subtraction protocol two times.
- Addition protocol #2 (Section 5.1.7): It takes two cards $([\![x_1]\!], [\![x_2]\!])$ as input and outputs a card $[\![x_1 + x_2]\!]$.

- Constant multiplication protocol (Section 5.1.8): It takes a card $[\![x]\!]$ and $m-1$ copies of $[\![0]\!]$ as input and outputs $([\![ix]\!])_{0\le i\le m-1}$. It calls an extended addition protocol (Section 5.1.7) multiple times.
- Multiplication protocol (Section 5.1.9): It takes two cards $([\![x_1]\!], [\![x_2]\!])$ and $m-1$ copies of $[\![0]\!]$ as input and outputs a card $[\![x_1x_2]\!]$. It calls the constant multiplication protocol once.
- Oblivious conversion #1 (Section 5.1.10): It takes a card $[\![x]\!]$, m copies of $[\![0]\!]$, and m sequences $(E_i)_{0\le i\le m-1}$ and outputs a sequence $[\![E_x]\!]$. It calls a copy protocol (Section 5.1.7) once.
- Oblivious conversion #2 (Section 5.1.11): It takes a card $[\![x]\!]$ and m sequences $(E_i)_{0\le i\le m-1}$ and outputs a sequence $[\![E_x]\!]$.
- General protocol (Section 5.1.12): It takes n cards $([\![x_1]\!], [\![x_2]\!], \ldots, [\![x_n]\!])$ and outputs a card $[\![f(x_1, x_2, \ldots, x_n)]\!]$. It calls the oblivious conversion #2 (Section 5.1.11) n times.

5.1.5 Subtraction protocol

Functionality. A functionality $\mathcal{F}^{\mathsf{c}}_{\mathsf{sub}}$ is defined as follows:

$$\mathcal{F}^{\mathsf{c}}_{\mathsf{sub}} : \underbrace{\square}_{[\![x_1]\!]}\underbrace{\square}_{[\![x_2]\!]} \Rightarrow \underbrace{\boxed{\uparrow}}_{\underline{0}}\underbrace{\square}_{[\![x_2-x_1]\!]}.$$

Protocol. A subtraction protocol $\mathcal{P}^{\mathsf{c}}_{\mathsf{sub}}$ is defined as follows:

$$\mathcal{P}^{\mathsf{c}}_{\mathsf{sub}} = (2, \mathbb{Z}_m, (\mathsf{Cyclic}_m, \{\underline{0}, \underline{0}\}), \mathcal{O}^{\mathsf{c}}_{m,2}, A).$$

It proceeds as follows:

1. $(\mathsf{rotshuf}, \{1, 2\})$: Apply a rotation shuffle to them:

$$\underbrace{\square}_{[\![x_1]\!]}\underbrace{\square}_{[\![x_2]\!]} \to \underbrace{\square}_{[\![x'_1]\!]}\underbrace{\square}_{[\![x'_2]\!]},$$

   where $x'_i := x_i - r$ $(i \in \{1, 2\})$ for a random $r \in \mathbb{Z}_m$.

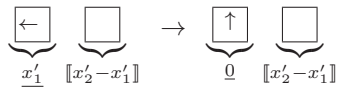
2. $(\mathsf{turn}, \{1\})$: Turn the left card $[\![x'_1]\!]$:

$$\underbrace{\square}_{[\![x'_1]\!]}\underbrace{\square}_{[\![x'_2]\!]} \to \underbrace{\boxed{\leftarrow}}_{\underline{x'_1}}\underbrace{\square}_{[\![x'_2]\!]}.$$

3. $(\mathsf{rot}, \{2\}, x'_1)$: Rotate the right card with a degree x'_1:

$$\underbrace{\boxed{\leftarrow}}_{\underline{x'_1}}\underbrace{\square}_{[\![x'_2]\!]} \to \underbrace{\boxed{\leftarrow}}_{\underline{x'_1}}\underbrace{\square}_{[\![x'_2-x'_1]\!]}.$$

4. $(\mathsf{rot}, \{1\}, -x'_1)$: Rotate the left card with a degree $-x'_1$:



The protocol terminates.

Correctness. Since $x'_1 = x_1 - r$ and $x'_2 = x_2 - r$, the right card in the final sequence is $[\![x'_2 - x'_1]\!] = [\![(x_2 - r) - (x_1 - r)]\!] = [\![x_2 - x_1]\!]$. Thus, the subtraction protocol $\mathcal{P}^{\mathsf{c}}_{\mathsf{sub}}$ correctly realizes the functionality $\mathcal{F}^{\mathsf{c}}_{\mathsf{sub}}$.

Security. Let $x = (x_1, x_2) \in (\mathbb{Z}_m)^2$ be any input. The probability distribution of a view of the protocol starting with the sequence $s_{\mathsf{in}}(x) = ([\![x_1]\!], [\![x_2]\!])$ is given as follows:

$$\mathsf{view}_{\mathcal{P}^{\mathsf{c}}_{\mathsf{sub}}}(s_{\mathsf{in}}(x), x) = \big((?, ?) \to (?, ?) \to (x'_1, ?) \to (x'_1, ?) \to (0, ?)\big),$$

where $x'_1 = x_1 - r$ for a uniform random value $r \in \mathbb{Z}_m$. This is equivalent to a probability distribution view^* defined as follows:

$$\mathsf{view}^* = \big((?, ?) \to (?, ?) \to (r', ?) \to (r', ?) \to (0, ?)\big),$$

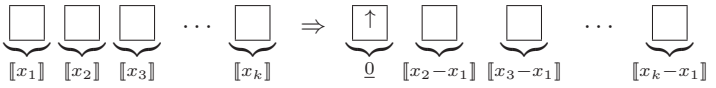
where $r' \in \mathbb{Z}_m$ is a uniform random value. The distribution view^* does not depend on x. Thus, for every $x, x' \in (\mathbb{Z}_m)^2$, the following holds:

$$\mathsf{view}_{\mathcal{P}^{\mathsf{c}}_{\mathsf{sub}}}(s_{\mathsf{in}}(x), x) = \mathsf{view}_{\mathcal{P}^{\mathsf{c}}_{\mathsf{sub}}}(s_{\mathsf{in}}(x'), x') = \mathsf{view}^*.$$

Therefore, $\mathcal{P}^{\mathsf{c}}_{\mathsf{sub}}$ securely realizes $\mathcal{F}^{\mathsf{c}}_{\mathsf{sub}}$.

Efficiency. The number of cards is two. Note that this is the minimum number of cards since the number of inputs is two. The number of probabilistic operations is one (one rotation shuffle).

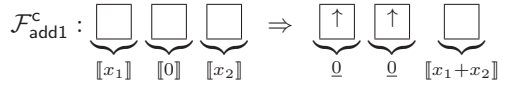
Extended subtraction protocol. We can extend to the above protocol into a protocol that (correctly and securely) realizes a functionality defined as follows:



We call it an *extended subtraction protocol*.

5.1.6 Addition protocol from a rotation shuffle

Functionality. A functionality $\mathcal{F}^{\mathsf{c}}_{\mathsf{add1}}$ is defined as follows:



Protocol. An addition protocol $\mathcal{P}^{\mathsf{c}}_{\mathsf{add1}}$ is defined as follows:

$$\mathcal{P}^{\mathsf{c}}_{\mathsf{add1}} = (2, \mathbb{Z}_m, (\mathsf{Cyclic}_m, \{\underline{0}, \underline{0}, \underline{0}\}), \mathcal{O}^{\mathsf{c}}_{m,3} \cup \mathsf{Oracle}[\mathcal{P}^{\mathsf{c}}_{\mathsf{sub}}], A).$$

It proceeds as follows:

1. $(\mathsf{oracle}, \mathcal{P}^{\mathsf{c}}_{\mathsf{sub}}, \{1, 2\})$: Apply the subtraction protocol to the first and second cards:
$$\underbrace{\square}_{[\![x_1]\!]}\ \underbrace{\square}_{[\![0]\!]}\ \underbrace{\square}_{[\![x_2]\!]} \to \underbrace{\boxed{\uparrow}}_{\underline{0}}\ \underbrace{\square}_{[\![-x_1]\!]}\ \underbrace{\square}_{[\![x_2]\!]}.$$
2. $(\mathsf{oracle}, \mathcal{P}^{\mathsf{c}}_{\mathsf{sub}}, \{2, 3\})$: Apply the subtraction protocol to the second and third cards:
$$\underbrace{\boxed{\uparrow}}_{\underline{0}}\ \underbrace{\square}_{[\![-x_1]\!]}\ \underbrace{\square}_{[\![x_2]\!]} \to \underbrace{\boxed{\uparrow}}_{\underline{0}}\ \underbrace{\boxed{\uparrow}}_{\underline{0}}\ \underbrace{\square}_{[\![x_2+x_1]\!]}.$$
The protocol terminates.

Correctness. The correctness is trivial from the above description.

Security. Let $x = (x_1, x_2) \in (\mathbb{Z}_m)^2$ be any input. The probability distribution of a view of the protocol starting with the sequence $s_{\mathsf{in}}(x) = ([\![x_1]\!], [\![0]\!], [\![x_2]\!])$ is given as follows:

$$\mathsf{view}_{\mathcal{P}^{\mathsf{c}}_{\mathsf{add1}}}(s_{\mathsf{in}}(x), x) = \big((?, ?, ?) \to (0, ?, ?) \to (0, 0, ?)\big).$$

It does not depend on x since it is just a fixed sequence. Thus, for every $x, x' \in (\mathbb{Z}_m)^2$, the following holds:

$$\mathsf{view}_{\mathcal{P}^{\mathsf{c}}_{\mathsf{add1}}}(s_{\mathsf{in}}(x), x) = \mathsf{view}_{\mathcal{P}^{\mathsf{c}}_{\mathsf{add1}}}(s_{\mathsf{in}}(x'), x').$$

Therefore, $\mathcal{P}^{\mathsf{c}}_{\mathsf{add1}}$ securely realizes $\mathcal{F}^{\mathsf{c}}_{\mathsf{add1}}$.

Efficiency. The number of cards is three. The number of oracle calls is two (two calls of the subtraction protocol). From Proposition 2.1, an addition protocol without oracles can be obtained. The number of shuffles of the protocol is two (two rotation shuffles).

5.1.7 Addition protocol from a backward rotation shuffle

Functionality. A functionality $\mathcal{F}^{\mathsf{c}}_{\mathsf{add2}}$ is defined as follows:

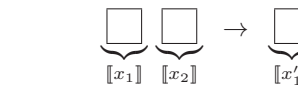
$$\mathcal{F}^{\mathsf{c}}_{\mathsf{add2}} : \underbrace{\square}_{[\![x_1]\!]}\ \underbrace{\square}_{[\![x_2]\!]} \Rightarrow \underbrace{\boxed{\uparrow}}_{\underline{0}}\ \underbrace{\square}_{[\![x_1+x_2]\!]}.$$

Protocol. An addition protocol $\mathcal{P}^{\mathsf{c}}_{\mathsf{add2}}$ is defined as follows:

$$\mathcal{P}^{\mathsf{c}}_{\mathsf{add2}} = (2, \mathbb{Z}_m, (\mathsf{Cyclic}_m, \{\underline{0}, \underline{0}\}), \mathcal{O}^{\mathsf{c}}_{m,2}, A).$$

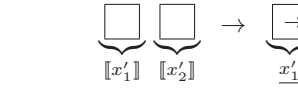
It proceeds as follows:

1. $(\mathsf{backrot}, \{1\}, \{2\})$: Apply a backward rotation shuffle to them:

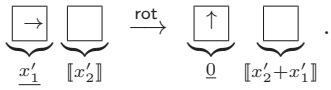


   where $x'_1 := x_1 - r$ and $x'_2 := x_2 + r$ for a random $r \in \mathbb{Z}_m$.

2. $(\mathsf{turn}, \{1\})$: Turn the left card $[\![x'_1]\!]$:



3. $(\mathsf{rot}, \{1, 2\}, -x'_1)$: Rotate them with a degree $-x'_1$:



   The protocol terminates.

Correctness. Since $x'_1 = x_1 - r$ and $x'_2 = x_2 + r$, the right card in the final sequence is $[\![x'_2 + x'_1]\!] = [\![(x_2 + r) + (x_1 - r)]\!] = [\![x_2 + x_1]\!]$. Thus, the addition protocol $\mathcal{P}^{\mathsf{c}}_{\mathsf{add2}}$ correctly realizes the functionality $\mathcal{F}^{\mathsf{c}}_{\mathsf{add2}}$.

Security. Let $x = (x_1, x_2) \in (\mathbb{Z}_m)^2$ be any input. The probability distribution of a view of the protocol starting with the sequence $s_{\mathsf{in}}(x) = ([\![x_1]\!], [\![x_2]\!])$ is given as follows:

$$\mathsf{view}_{\mathcal{P}^{\mathsf{c}}_{\mathsf{add2}}}(s_{\mathsf{in}}(x), x) = \big((?, ?) \to (?, ?) \to (x'_1, ?) \to (0, ?)\big),$$

where $x'_1 = x_1 - r$ for a uniform random value $r \in \mathbb{Z}_m$. This is equivalent to a probability distribution view^* defined as follows:

$$\mathsf{view}^* = \big((?, ?) \to (?, ?) \to (r', ?) \to (0, ?)\big),$$

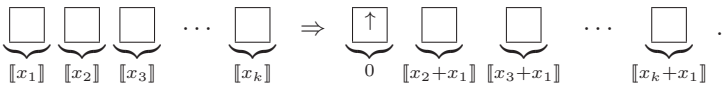
where $r' \in \mathbb{Z}_m$ is a uniform random value. The distribution view^* does not depend on x. Thus, for every $x, x' \in (\mathbb{Z}_m)^2$, the following holds:

$$\mathsf{view}_{\mathcal{P}^{\mathsf{c}}_{\mathsf{add2}}}(s_{\mathsf{in}}(x), x) = \mathsf{view}_{\mathcal{P}^{\mathsf{c}}_{\mathsf{add2}}}(s_{\mathsf{in}}(x'), x') = \mathsf{view}^*.$$

Therefore, $\mathcal{P}^{\mathsf{c}}_{\mathsf{add2}}$ securely realizes $\mathcal{F}^{\mathsf{c}}_{\mathsf{add2}}$.

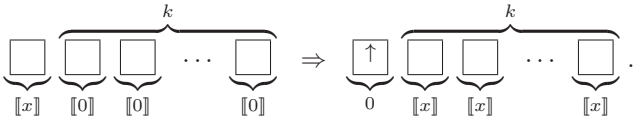
Efficiency. The number of cards is two. Note that this is the minimum number of cards since the number of inputs is two. The number of probabilistic operations is one (one backward rotation shuffle).

Extended addition protocol. We extend to the above protocol into a protocol that (correctly and securely) realizes a functionality defined as follows:



We call it an *extended addition protocol* denoted by $\mathcal{P}^{\mathsf{c}}_{\mathsf{exadd}}$.

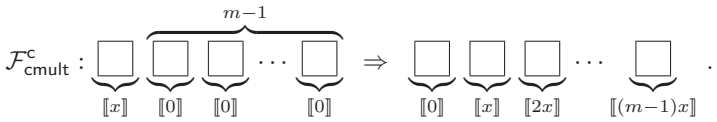
Copy protocol. By using a multiple addition protocol, we also obtain a protocol that (correctly and securely) realizes a functionality defined as follows:



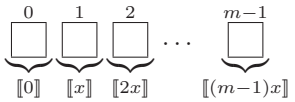
We call it a *copy protocol* denoted by $\mathcal{P}^{\mathsf{c}}_{\mathsf{copy}}$.

5.1.8 Constant multiplication protocol

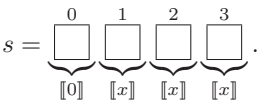
Functionality. A functionality $\mathcal{F}^{\mathsf{c}}_{\mathsf{cmult}}$ is defined as follows:



Notation. The goal of the protocol is for every $0 \leq i \leq m-1$, the card on position i is $[\![ix]\!]$ as follows:



For any sequence $s = ([\![a_0 x]\!], [\![a_1 x]\!], \ldots, [\![a_{m-1} x]\!])$ for $0 \leq a_i \leq m-1$, we define the *capacity of the card on position* i by $i - a_i$. We say that a *card on position* i *achieves the goal* if the capacity of the card on position i is 0, i.e., $a_i = i$. For example, when $m = 4$, consider a sequence s as follows:



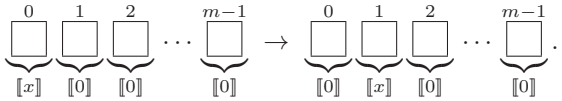
The capacities of the cards on positions $\{0, 1, 2, 3\}$ are $0, 0, 1$, and 2, respectively. Thus, the cards on positions $\{0, 1\}$ achieve the goal. On the other hand, the cards on positions $\{2, 3\}$ do not achieve the goal.

Protocol. A constant multiplication protocol $\mathcal{P}^{\mathsf{c}}_{\mathsf{cmult}}$ is defined as follows:

$$\mathcal{P}^{\mathsf{c}}_{\mathsf{cmult}} = (1, \mathbb{Z}_m, (\mathsf{Cyclic}_m, \{\underbrace{\underline{0}, \underline{0}, \cdots, \underline{0}}_{m}\}), \mathcal{O}^{\mathsf{c}}_{m,m} \cup \mathsf{Oracle}[\mathcal{P}^{\mathsf{c}}_{\mathsf{exadd}}], A)$$

It proceeds as follows:

1. $(\mathsf{perm}, (1\ 2))$: Swap the cards on positions $\{0, 1\}$ as follows:



   After the permutation, the cards on positions $\{0, 1\}$ achieve the goal.

2. Repeat the following until all cards achieve the goal:

   (a) Let $i^* \in \{0, 1, 2, \ldots, m-1\}$ be the maximum value satisfying the conditions as follows:

   - The card on position i achieves the goal.
   - The capacity of the card on position $m-1$ is equal to or greater than i.

   (b) $(\mathsf{perm}, (1\ \ i^*+1))$: Swap the cards on positions $\{0, i^*\}$. After the permutation, the card on position i does not achieve the goal.

   (c) Let $T \subset \{0, 1, \ldots, m-1\}$ be a subset of positions $j \in \{0, 1, \ldots, m-1\}$ satisfying the conditions as follows:

   - The card on position j does not achieve the goal.
   - The capacity of the card on position j is equal or greater than i^*.

   Note that $i^* \in T$ since the card on position i does not achieve the goal. We note that $m-1 \in T$ since the card on position $m-1$ has the largest capacity. Apply $(\mathsf{oracle}, \mathcal{P}^{\mathsf{c}}_{\mathsf{exadd}}, T')$ for $T' = \{1\} \cup \{i+1 \mid i \in T\}$, where $\mathcal{P}^{\mathsf{c}}_{\mathsf{exadd}}$ is an extended addition protocol.

   (d) $(\mathsf{turn}, 1)$: Turn the card on position 0 so that it changed to a face-down card $[\![0]\!]$. Note that before applying the turning, the card on position 0 is a face-up card $\underline{0}$.

Correctness. We can observe that at least one $[\![x]\!]$ is contained in every step of the current sequence throughout a protocol execution. Thus, there exists i^* satisfying the conditions at Step 2(a). By applying the extended addition protocol a number of times, it eventually matches the finial sequence of $\mathcal{F}^{\mathsf{c}}_{\mathsf{cmult}}$. Thus, the above protocol $\mathcal{P}^{\mathsf{c}}_{\mathsf{cmult}}$ correctly realizes $\mathcal{F}^{\mathsf{c}}_{\mathsf{cmult}}$.

Efficiency. The number of cards is m. The number of oracle calls $a_m \in \mathbb{N}$ is computed as follows:

$$a_m = 2L + \mathsf{p}(m \geq 2^L + 2^{L-1}) - 2,$$

where $L = \lceil \log m \rceil$. From Proposition 2.1, a constant multiplication protocol without oracles can be obtained. The number of shuffles of the protocol is a_m (a_m backward rotation shuffles).

Security. Let $x \in \mathbb{Z}_m$ be any input. The probability distribution of a view of the protocol starting with the sequence $s_{\mathsf{in}}(x) = (\llbracket x \rrbracket, \llbracket 0 \rrbracket, \llbracket 0 \rrbracket, \ldots, \llbracket 0 \rrbracket)$ is given as follows:

$$\mathsf{view}_{\mathcal{P}^{\mathsf{c}}_{\mathsf{cmult}}}(s_{\mathsf{in}}(x), x) = \big(?^m(\to ?^m \to ?^m \to (0, ?^{m-1}) \to ?^m)^{a_m}\big),$$

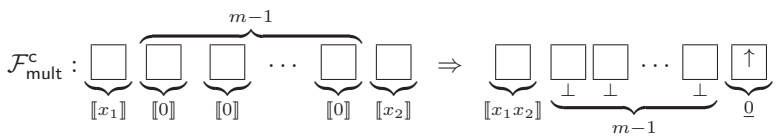
Note that it means a_m repetitions of "$\to ?^m \to ?^m \to (0, ?^{m-1}) \to ?^m$". The distribution view^* does not depend on x since it is just a fixed sequence. Thus, for every $x, x' \in \mathbb{Z}_m$, the following holds:

$$\mathsf{view}_{\mathcal{P}^{\mathsf{c}}_{\mathsf{cmult}}}(s_{\mathsf{in}}(x), x) = \mathsf{view}_{\mathcal{P}^{\mathsf{c}}_{\mathsf{cmult}}}(s_{\mathsf{in}}(x'), x') = \mathsf{view}^*.$$

Therefore, $\mathcal{P}^{\mathsf{c}}_{\mathsf{cmult}}$ securely realizes $\mathcal{F}^{\mathsf{c}}_{\mathsf{cmult}}$.

5.1.9 Multiplication protocol

Functionality. A functionality $\mathcal{F}^{\mathsf{c}}_{\mathsf{mult}}$ is defined as follows:

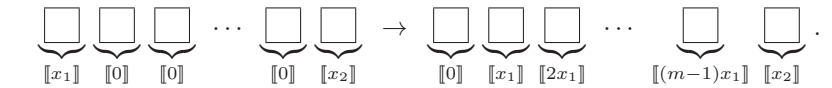


Protocol. A multiplication protocol $\mathcal{P}^{\mathsf{c}}_{\mathsf{mult}}$ is defined as follows:

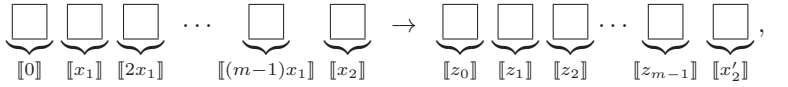
$$\mathcal{P}^{\mathsf{c}}_{\mathsf{mult}} = (2, \mathbb{Z}_m, (\mathsf{Cyclic}_m, \{\underbrace{\underline{0}, \underline{0}, \cdots, \underline{0}}_{m+1}\}), \mathcal{O}^{\mathsf{c}}_{m,m+1} \cup \mathsf{Oracle}[\mathcal{P}^{\mathsf{c}}_{\mathsf{cmult}}], A).$$

It proceeds as follows:

1. $(\mathsf{oracle}, \mathcal{P}^{\mathsf{c}}_{\mathsf{cmult}}, \{1, 2, \ldots, m\})$: Apply the constant multiplication protocol $\mathcal{P}^{\mathsf{c}}_{\mathsf{cmult}}$ to the former m cards as follows:

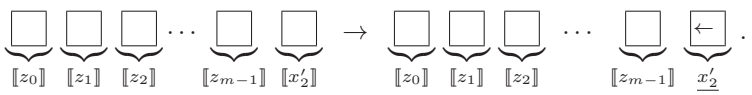


2. $(\mathsf{flower}, \{m+1\}, (1), (2), \ldots, (m))$: Apply a flower shuffle to them:

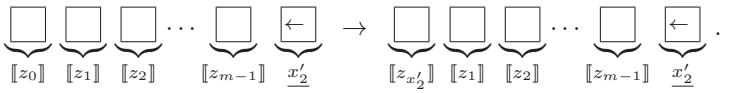


where $z_i = (r+i)x_1$ and $x'_2 = x_2 - r$ for a random $r \in \mathbb{Z}_m$.

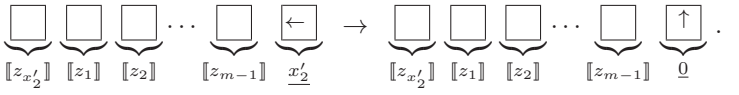
3. $(\mathsf{turn}, \{m+1\})$: Turn the rightmost card:



4. $(\mathsf{perm}, (1 \;\; x_2'))$: Swap $[\![z_0]\!]$ and $[\![z_{x_2'}]\!]$ as follows:



5. $(\mathsf{rot}, \{m+1\}, -x_2')$: Rotate the rightmost card with a degree $-x_2'$:



The protocol terminates.

Correctness. Since $z_i = (r+i)x_1$ and $x_2' = x_2 - r$ for a random $r \in \mathbb{Z}_m$, the card $[\![z_{x_2'}]\!]$ is equal to $[\![z_{x_2'}]\!] = [\![(r + x_2')x_1]\!] = [\![(r + x_2 - r)x_1]\!] = [\![x_1 x_2]\!]$. Thus, the above protocol $\mathcal{P}^{\mathsf{c}}_{\mathsf{mult}}$ correctly realizes the functionality $\mathcal{F}^{\mathsf{c}}_{\mathsf{mult}}$.

Security. Let $x = (x_1, x_2) \in (\mathbb{Z}_m)^2$ be any input. The probability distribution of a view of the protocol starting with the sequence $s_{\mathsf{in}}(x) = ([\![x_1]\!], [\![0]\!], [\![0]\!], \ldots, [\![0]\!], [\![x_2]\!])$ is given as follows:

$$\mathsf{view}_{\mathcal{P}^{\mathsf{c}}_{\mathsf{mult}}}(s_{\mathsf{in}}(x), x) = \big(\boldsymbol{?}^{m+1} \to \boldsymbol{?}^{m+1} \to \boldsymbol{?}^{m+1} \to (\boldsymbol{?}^m, x_2') \to (\boldsymbol{?}^m, x_2') \to (\boldsymbol{?}^m, 0)\big),$$

where $x_2' = x_2 - r$ for a uniform random value $r \in \mathbb{Z}_m$. This is equivalent to a probability distribution view^* defined as follows:

$$\mathsf{view}^* = \big(\boldsymbol{?}^{m+1} \to \boldsymbol{?}^{m+1} \to \boldsymbol{?}^{m+1} \to (\boldsymbol{?}^m, r') \to (\boldsymbol{?}^m, r') \to (\boldsymbol{?}^m, 0)\big),$$

where $r' \in \mathbb{Z}_m$ is a uniform random value. The distribution view^* does not depend on x. Thus, for every $x, x' \in (\mathbb{Z}_m)^2$, the following holds:

$$\mathsf{view}_{\mathcal{P}^{\mathsf{c}}_{\mathsf{mult}}}(s_{\mathsf{in}}(x), x) = \mathsf{view}_{\mathcal{P}^{\mathsf{c}}_{\mathsf{mult}}}(s_{\mathsf{in}}(x'), x') = \mathsf{view}^*.$$

Therefore, $\mathcal{P}^{\mathsf{c}}_{\mathsf{mult}}$ securely realizes $\mathcal{F}^{\mathsf{c}}_{\mathsf{mult}}$.

Efficiency. The number of cards is $m + 1$. The number of oracle calls is one (one call of the constant multiplication protocol). From Proposition 2.1, a multiplication protocol without oracles can be obtained. The number of shuffles of the protocol is a_m defined in Efficiency of the constant multiplication protocol (a_m backward rotation shuffles).

5.1.10 Oblivious conversion from a pile random cut

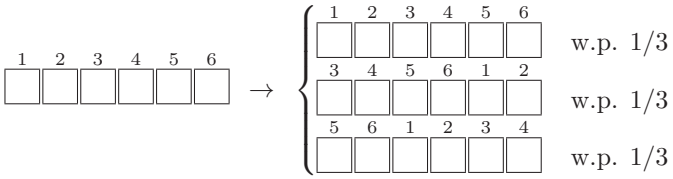
Functionality. A functionality $\mathcal{F}^{\mathsf{c}}_{\mathsf{oc1}} = (s_{\mathsf{in}}, s_{\mathsf{out}})$ is defined as follows:

$$s_{\mathsf{in}} = (\llbracket x \rrbracket, \underbrace{\llbracket 0 \rrbracket, \llbracket 0 \rrbracket, \ldots, \llbracket 0 \rrbracket}_{m}, E_0, E_1, \ldots, E_{m-1}).$$

$$s_{\mathsf{out}} = (\underbrace{\underline{0}, \underline{0}, \underline{0}, \ldots, \underline{0}}_{m+1}, E_x, E_{x+1}, \ldots, E_{x+m-1}).$$

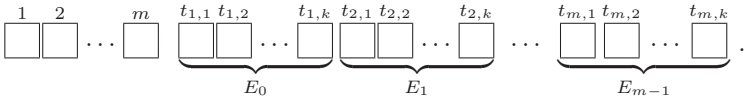
Here, $E_0, E_1, \ldots, E_{m-1}$ are face-down sequences of the same length k, i.e., $\mathsf{vis}(E_i) = \mathbf{?}^k$ for all i.

Pile random cut. A pile random cut is a shuffle operation, which is a generalized version of random cuts. An example of a pile random cut is as follows:



That is, it randomly shifts the order of piles like a random cut.

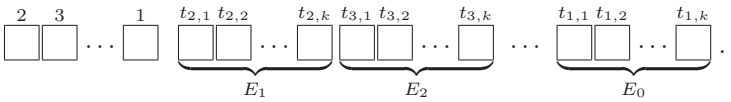
Now we define a pile random cut used in the oblivious conversion protocol. The sequence just before applying the shuffle is as follows:



Define a shift permutation σ as follows:

$$\sigma = (m\ m-1\ \cdots\ 2\ 1)\tau_1\tau_2\cdots\tau_k,$$

for $\tau_i = (t_{m,i}\ t_{m-1,i}\ \cdots\ t_{2,i}\ t_{1,i})$ $(i \in [k])$. By applying the shift permutation σ to the above sequence, it is changed to a sequence as follows:



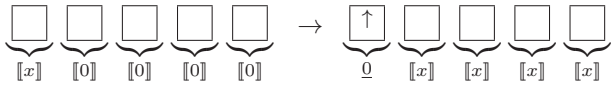
The pile random cut is a shuffle that applies a shift permutation σ^r for a random $0 \le r \le k-1$. Formally, it is defined by $(\mathsf{shuffle}, \Pi = \{\mathsf{id}, \sigma, \sigma^2, \ldots, \sigma^{k-1}\}, U_\Pi)$ where U_Π is the uniform distribution over Π.

Protocol. An oblivious conversion protocol $\mathcal{P}^{\mathsf{c}}_{\mathsf{oc1}}$ is defined as follows:

$$\mathcal{P}^{\mathsf{c}}_{\mathsf{oc1}} = (1, \mathbb{Z}_m, (\mathsf{Cyclic}_m, \{\underbrace{\underline{0}, \underline{0}, \ldots, \underline{0}}_{\ell}\}), \mathcal{O}^{\mathsf{c}}_{m,\ell} \cup \mathsf{Oracle}[\mathcal{P}^{\mathsf{c}}_{\mathsf{copy}}], A),$$

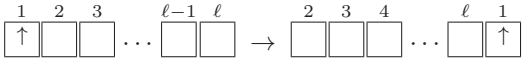
where ℓ is defined by $\ell = m + 1 + mk$ for $k = |E_0|$. It proceeds as follows:

1. $(\mathsf{oracle}, \mathcal{P}^{\mathsf{c}}_{\mathsf{copy}}, \{1, 2, 3, \ldots, m\})$: Applying the copy protocol to the leftmost $m+1$ cards as follows:

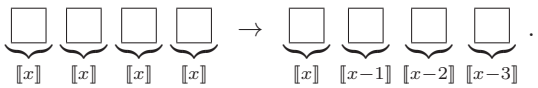


   Note that this is the former $m+1$ cards when $m = 4$.

2. $(\mathsf{perm}, (\ell \ \cdots \ 3 \ 2 \ 1))$ for $\ell = m + 1 + mk$: Rearrange the order of the sequence so that the leftmost card is moved to the rightmost as follows:



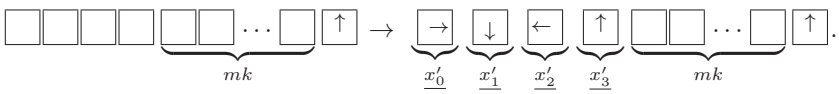
3. For $i \in \{2, 3, \ldots, m\}$, apply a rotation operation $(\mathsf{rot}, \{i\}, i-1)$ as follows:



   Note that this is the former m cards when $m = 4$.

4. $(\mathsf{shuffle}, \Pi, U_\Pi)$: Apply the pile random cut defined in the above.

5. $(\mathsf{turn}, \{1, 2, \ldots, m\})$: Turn the leftmost m cards as follows:



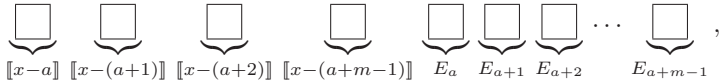
   A list of the opened values $(x'_0, x'_1, \ldots, x'_{m-1})$ is expected to be a cyclic shit of the decreasing order $(m-1, m-2, \ldots, 1, 0)$. If it is not the case, the action function outputs $\perp$ (i.e., the protocol execution terminates before entering the next operation). Let $i^* \in [m]$ be the index such that $x'_{i^*} = 0$. (In the above sequence, $i^* = 3$ since $x'_3 = 0$.)

6. Apply a permutation operation $(\mathsf{perm}, \sigma^{i^*})$ to the sequence.

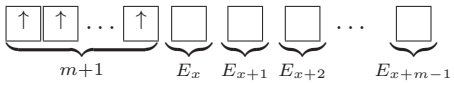
7. Apply rotation operations so that every faced-up card is changed to $\underline{0}$. (Thus, $m-1$ rotation operations are applied.)

8. Apply a permutation operation so that all face-up cards $\underline{0}$ are moved to the leftmost cards without changing the order of other face-down cards.

Correctness. Suppose that in the pile random cut at Step 4, a shift permutation σ^r is chosen for some $0 \le r \le k-1$. In addition, at Step 6, it is applied a shift permutation σ^{i^*}. Thus, the sequence just after Step 6 should be given as follows:



where $a = r + i^*$ and each E_j is depicted by a single card. Just after Step 3, the x-th card of the sequence is $[\![0]\!]$. (Here, the leftmost card is the 0-th card.) From this, we can observe that $i^* = x - r$. Thus, $a = r + i^* = r + (x - r) = x$. It means that the final sequence is:



which is matched with the output sequence of s_{out}. Therefore, the protocol $\mathcal{P}^{\mathsf{c}}_{\mathsf{oc1}}$ correctly realizes the functionality $\mathcal{F}^{\mathsf{c}}_{\mathsf{oc1}}$.

Security. Let $x \in \mathbb{Z}_m$ be any input. The probability distribution of a view of the protocol starting with the sequence $s_{\mathsf{in}}(x) = ([\![x]\!], [\![0]\!]^m, E_0, E_1, \ldots, E_{m-1})$ is given as follows:

$$\begin{aligned}\mathsf{view}_{\mathcal{P}^{\mathsf{c}}_{\mathsf{oc1}}}(s_{\mathsf{in}}(x), x) = \Big(&\cdots \to (\mathbf{?}^{m(k+1)}, 0) \to (x'_0, x'_1, \ldots, x'_{m-1}, \mathbf{?}^{mk}, 0)\\ &\to (0, m-1, m-2, \ldots, 1, \mathbf{?}^{mk}, 0) \to \cdots\\ &\to (0^m, \mathbf{?}^{mk}, 0) \to \cdots \to (\mathbf{?}^{m(k+1)}, 0)\Big).\end{aligned}$$

(Note that the view before Step 4 is omitted since it is not essential for proving the security.) The above $x' = (x'_0, x'_1, \ldots, x'_{m-1})$ is a cyclic shit of the decreasing order $(m-1, m-2, \ldots, 1, 0)$. Since the index i^* such that $x'_{i^*} = 0$ is $i^* = x - r$, it is equivalent to a probability distribution view^* defined as follows:

$$\begin{aligned}\mathsf{view}^* = \Big(&\cdots \to (\mathbf{?}^{m(k+1)}, 0) \to (r'_0, r'_1, \ldots, r'_{m-1}, \mathbf{?}^{mk}, 0)\\ &\to (0, m-1, m-2, \ldots, 1, \mathbf{?}^{mk}, 0) \to \cdots\\ &\to (0^m, \mathbf{?}^{mk}, 0) \to \cdots \to (\mathbf{?}^{m(k+1)}, 0)\Big),\end{aligned}$$

where $(r'_0, r'_1, \ldots, r'_{m-1})$ is a uniform random shift of $(m-1, m-2, \ldots, 1, 0)$. The distribution view^* does not depend on x. Thus, for every $x, x' \in \mathbb{Z}_m$, the following holds:

$$\mathsf{view}_{\mathcal{P}^{\mathsf{c}}_{\mathsf{oc1}}}(s_{\mathsf{in}}(x), x) = \mathsf{view}_{\mathcal{P}^{\mathsf{c}}_{\mathsf{oc1}}}(s_{\mathsf{in}}(x'), x') = \mathsf{view}^*.$$

Therefore, $\mathcal{P}^{\mathsf{c}}_{\mathsf{oc1}}$ securely realizes $\mathcal{F}^{\mathsf{c}}_{\mathsf{oc1}}$.

Efficiency. The number of cards is $(k+1)m+1$. The number of oracle calls is one (one call of the copy protocol). From Proposition 2.1, an oblivious conversion without oracles can be obtained. The number of probabilistic operations is two (one backward rotation shuffle and one pile random cut).

5.1.11 Oblivious conversion from a flower shuffle

Functionality. A functionality $\mathcal{F}^{\mathsf{c}}_{\mathsf{oc2}} = (s_{\mathsf{in}}, s_{\mathsf{out}})$ is defined as follows:

$$s_{\mathsf{in}} = (E_0, E_1, \ldots, E_{m-1}, [\![x]\!]).$$
$$s_{\mathsf{out}} = (E_x, E_{x+1}, \ldots, E_{x+m-1}, \underline{0}).$$

Here, $E_0, E_1, \ldots, E_{m-1}$ are face-down sequences of the same length k, i.e., $\mathsf{vis}(E_i) = \mathsf{?}^k$ for all i.

Protocol. An oblivious conversion protocol $\mathcal{P}^{\mathsf{c}}_{\mathsf{oc2}}$ is defined as follows:

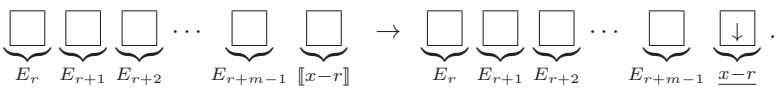
$$\mathcal{P}^{\mathsf{c}}_{\mathsf{oc2}} = (1, \mathbb{Z}_m, (\mathsf{Cyclic}_m, \{\underbrace{\underline{0}, \underline{0}, \ldots, \underline{0}}_{\ell}\}), \mathcal{O}^{\mathsf{c}}_{m,\ell}, A),$$

where ℓ is defined by $\ell = 1 + mk$ for $k = |E_0|$. It proceeds as follows:

1. $(\mathsf{flower}, \{mk{+}1\}, T_0, T_1, \ldots, T_{m-1})$ for $T_i = ((i{-}1)k{+}1, (i{-}1)k{+}2, \ldots, ik)$: Apply a flower shuffle to the sequence:

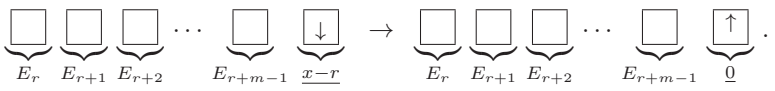
   where $r \in \mathbb{Z}_m$ is a random number and each E_i is depicted by a single card although it is a sequence of k cards.

2. $(\mathsf{turn}, \{mk+1\})$: Turn the rightmost card as follows:



   Let $x' = x - r \in \mathbb{Z}_m$ be the open value.

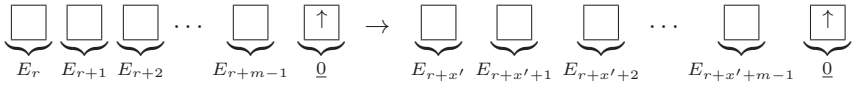
3. $(\mathsf{rot}, \{mk+1\}, -x')$: Rotate the rightmost card so that it is changed to $\underline{0}$ as follows:



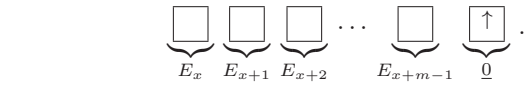
4. $(\mathsf{perm}, \sigma^{x'})$ where σ is defined by $\sigma = \tau_1\tau_2\cdots\tau_k$ where:

$$\tau_i = ((m-1)k+i \;\; (m-2)k+i \;\; \cdots \;\; 2k+i \;\; k+i \;\; i).$$

Correctness. By applying a permutation $\sigma^{x'}$ at Step 4, the sequence is changed as follows:



Since $x' = x - r$, this is equivalent to the following:



Therefore, the protocol $\mathcal{P}^{\mathsf{c}}_{\mathsf{oc2}}$ correctly realizes the functionality $\mathcal{F}^{\mathsf{c}}_{\mathsf{oc2}}$.

Security. Let $x \in \mathbb{Z}_m$ be any input. The probability distribution of a view of the protocol starting with the sequence $s_{\mathsf{in}}(x) = (E_0, E_1, \ldots, E_{m-1}, [\![x]\!])$ is given as follows:

$$\mathsf{view}_{\mathcal{P}^{\mathsf{c}}_{\mathsf{oc2}}}(s_{\mathsf{in}}(x), x) = \big(?^{mk+1} \to ?^{mk+1} \to (?^{mk}, x') \to (?^{mk}, 0) \to (?^{mk}, 0)\big),$$

where $x' = x - r$ for a uniform random $r \in \mathbb{Z}_m$. It is equivalent to a probability distribution view^* defined as follows:

$$\mathsf{view}^* = \big(?^{mk+1} \to ?^{mk+1} \to (?^{mk}, r') \to (?^{mk}, 0) \to (?^{mk}, 0)\big),$$

where $r' \in \mathbb{Z}_m$ is a uniform random number. The distribution view^* does not depend on x. Thus, for every $x, x' \in \mathbb{Z}_m$, the following holds:

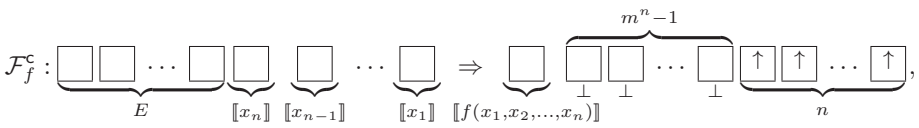
$$\mathsf{view}_{\mathcal{P}^{\mathsf{c}}_{\mathsf{oc2}}}(s_{\mathsf{in}}(x), x) = \mathsf{view}_{\mathcal{P}^{\mathsf{c}}_{\mathsf{oc2}}}(s_{\mathsf{in}}(x'), x') = \mathsf{view}^*.$$

Therefore, $\mathcal{P}^{\mathsf{c}}_{\mathsf{oc2}}$ securely realizes $\mathcal{F}^{\mathsf{c}}_{\mathsf{oc2}}$.

Efficiency. The number of cards is $km + 1$. The number of probabilistic operations is one (one flower shuffle).

5.1.12 General protocol from oblivious conversion

Functionality. Let $f : \{0,1\}^n \to \{0,1\}$ be any function. A functionality $\mathcal{F}^{\mathsf{c}}_f$ is defined as follows:



where E is a sequence of m^n cards defined as follows:

$$E = ([\![f(z)]\!])_{z \in (\mathbb{Z}_m)^n}.$$

The order of E is the lexical order as follows:

$$E = ([\![f(0, \cdots, 0, 0)]\!], [\![f(0, \cdots, 0, 1)]\!], [\![f(0, \cdots, 0, 2)]\!], \ldots, [\![f(0, \cdots, 0, m-1)]\!],$$
$$[\![f(0, \cdots, 1, 0)]\!], [\![f(0, \cdots, 1, 1)]\!], \ldots, [\![f(m-1, \cdots, m-1, m-1)]\!]).$$

For any $z \in (\mathbb{Z}_m)^n$ and $k \in \mathbb{N}$, we define $E_{z:k}$ by a subsequence of E of length k starting from $[\![f(z)]\!]$.

Protocol. A protocol $\mathcal{P}^{\mathsf{c}}_f$ for f is defined as follows:

$$\mathcal{P}^{\mathsf{c}}_f = (n, \mathbb{Z}_m, (\mathsf{Cyclic}_m, \{\underbrace{\underline{0}, \underline{0}, \ldots, \underline{0}}_{\ell}\}), \mathcal{O}^{\mathsf{c}}_{m,\ell} \cup \mathsf{Oracle}[\mathcal{P}^{\mathsf{c}}_{\mathsf{oc2}}], A),$$

where ℓ is defined by $\ell = n + m^n$. It proceeds as follows:

1. For any $x \in \mathbb{Z}_m$, define $z(x) \in (\mathbb{Z}_m)^n$ by $z(x) = (x, 0, 0, \ldots, 0)$. Apply the oblivious conversion protocol $\mathcal{P}^{\mathsf{c}}_{\mathsf{oc2}}$ to the sequence

   $$(E_{z(0):m^{n-1}}, E_{z(1):m^{n-1}}, \ldots, E_{z(m-1):m^{n-1}})$$

   with a commitment $[\![x_1]\!]$. Then, the first m^{n-1} cards are changed to $E_{z(x_1):m^{n-1}}$.

2. For any $x \in \mathbb{Z}_m$, define $z(x_1, x) \in (\mathbb{Z}_m)^n$ by $z(x_1, x) = (x_1, x, 0, \ldots, 0)$. Apply the oblivious conversion protocol $\mathcal{P}^{\mathsf{c}}_{\mathsf{oc2}}$ to the sequence

   $$(E_{z(x_1,0):m^{n-2}}, E_{z(x_1,1):m^{n-2}}, \ldots, E_{z(x_1,m-1):m^{n-2}})$$

   with a commitment $[\![x_2]\!]$. Then, the first m^{n-2} cards are changed to $E_{z(x_1,x_2):m^{n-2}}$.

3. Similarly, in the i-th iteration $1 \leq i \leq n$, define $z(x_1, x_2, \ldots, x_{i-1}, x) \in (\mathbb{Z}_m)^n$ by $z(x_1, x_2, \ldots, x_{i-1}, x) = (x_1, x_2, \ldots, x_{i-1}, x, 0, \ldots, 0)$. Apply the oblivious conversion protocol $\mathcal{P}^{\mathsf{c}}_{\mathsf{oc2}}$ to the sequence

   $$(E_{z(x_1,x_2,\ldots,x_{i-1},0):m^{n-i}}, E_{z(x_1,x_2,\ldots,x_{i-1},1):m^{n-i}}, \ldots, E_{z(x_1,x_2,\ldots,x_{i-1},m-1):m^{n-i}})$$

   with a commitment $[\![x_i]\!]$. Then, the first m^{n-i} cards are changed to $E_{z(x_1,x_2,\ldots,x_{i-1},x_i):m^{n-i}}$.

4. After n iterations, the first card is $E_{z(x_1,x_2,\ldots,x_n):1} = [\![f(x_1, x_2, \ldots, x_n)]\!]$. The protocol terminates.

Correctness. We can observe that the first m^{n-i} cards in the sequence after the i-th iteration is $E_{z(x_1,x_2,\ldots,x_{i-1},x_i):m^{n-i}}$. Thus, the first card in the final sequence is $E_{z(x_1,x_2,\ldots,x_n):1}$ that is equivalent to $[\![f(x_1, x_2, \ldots, x_n)]\!]$. Thus, the protocol $\mathcal{P}^{\mathsf{c}}_f$ correctly realizes the functionality $\mathcal{F}^{\mathsf{c}}_f$.

Security. Let $x = (x_1, x_2, \ldots, x_n) \in (\mathbb{Z}_m)^n$ be any input. The probability distribution of a view of the protocol starting with the sequence $s_{\mathsf{in}}(x) = E = ([\![f(z)]\!])_{z \in (\mathbb{Z}_m)^n}$ is given as follows:

$$\begin{aligned}\mathsf{view}_{\mathcal{P}^{\mathsf{c}}_f}(s_{\mathsf{in}}(x), x) = \big(&?^{m^n+n} \to (?^{m^n+n-1}, 0) \to (?^{m^n+n-2}, 0^2) \to (?^{m^n+n-3}, 0^3) \\ &\to \cdots \to (?^{m^n+1}, 0^{n-1}) \to (?^{m^n}, 0^n)\big).\end{aligned}$$

It does not depend on x since it is just a fixed sequence. Thus, for every $x, x' \in (\mathbb{Z}_m)^n$, the following holds:

$$\mathsf{view}_{\mathcal{P}^{\mathsf{c}}_f}(s_{\mathsf{in}}(x), x) = \mathsf{view}_{\mathcal{P}^{\mathsf{c}}_f}(s_{\mathsf{in}}(x'), x').$$

Therefore, $\mathcal{P}^{\mathsf{c}}_f$ securely realizes $\mathcal{F}^{\mathsf{c}}_f$.

Efficiency. The number of cards is $m^n + n$. The number of oracle calls is n (n calls of the oblivious conversion). From Proposition 2.1, a general protocol without oracles can be obtained. The number of probabilistic operations is n (n flower shuffles).

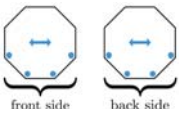
5.2 Protocols based on dihedral cards

5.2.1 Dihedral cards

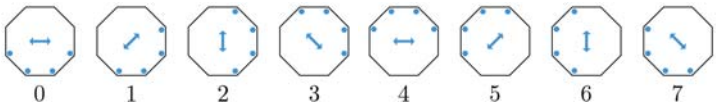
Let $m \geq 2$ be any integer. A *dihedral card of modulus* m is a card as follows:

- It holds a non-binary value $x \in \mathbb{Z}_{2m}$;
- A transformation from x to $x + c$ (for any constant $c \in \mathbb{Z}_{2m}$) is allowed;
- A transformation from x to $-x + c$ (for any constant $c \in \mathbb{Z}_{2m}$) is allowed;
- For a card holding x, it is possible to observe whether $x \geq m$ only;
- For a card holding x, it is possible to observe x mod m only.

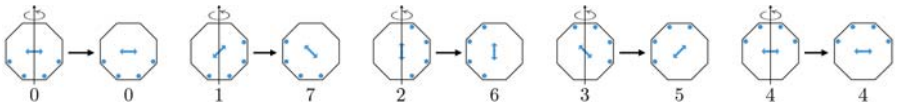
Thus, the shape of dihedral cards of modulus m is a regular $2m$-sided polygon. For example, a dihedral card of modulus 4 is implemented as follows:



Four vertices among eight vertices have blue dots and an arrow is written on the center. The front side and the back side are the same pattern satisfying that any vertex having a blue dot in the front side also has a dot in the back side. Here, all blue circles and arrows are written by *invisible ink*[1] in order to hide a value of a card. Since it is a hexagon, it can hold a value $x \in \mathbb{Z}_8$ as follows:

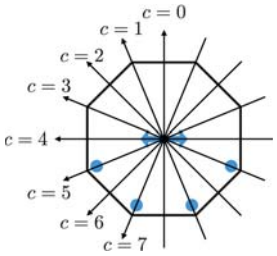


The first transformation from x to $x + c$ is done by a rotation with $(360/2m)°$ as in the case of cyclic cards. A nontrivial property is to allow the second transformation from x to $-x + c$. This is done by a *flipping*. Say $c = 0$. A transformation from x to $-x$ is done by a flipping with a vertical line as follows:

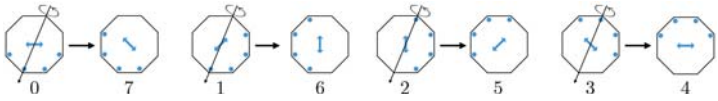


[1] Invisible ink is used for writing, which is invisible but can be made visible with illuminating a *black light*. It can be used for *steganography*, which hides the existence of plain texts while *cryptography* hides the contents of plain texts.

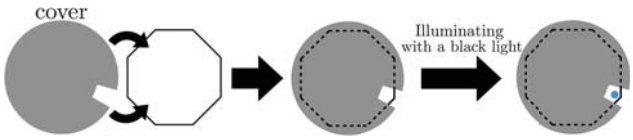
Each axis of line symmetry corresponds to some $c \in \mathbb{Z}_8$ as follows:



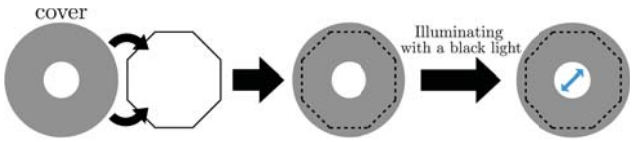
Indeed, a transformation from x to $-x + 7$ is done by a flipping as follows:



Finally, we need to open a bit $\mathsf{p}(x \geq m)$ and a value x mod m. Thanks to the property of invisible ink, this is done by illuminating a black light with a *cover*. For a card holding x, it is possible to observe $\mathsf{p}(x \geq m)$ only as follows:



In the above case, since the vertex has a blue dot, the predicate $\mathsf{p}(x \geq m)$ is 0. (We can observe that for a card holding x, the vertex has a blue dot if and only if $x < 4$.) Similarly, it is possible to observe the value x mod m only as follows:



In the above case, since the card holds either 1 or 5, the value x mod m is 1. For $x \in \mathbb{Z}_{2m}$, $\mathsf{p}(x \geq m)$ is called a *sign of* x and x mod m is called a *value of* x.

A card specification of dihedral cards. For $x \in \mathbb{Z}_{2m}$, we denote a card holding x by $[\![x]\!]$. The card set of dihedral cards of modulus m, denoted by $\mathcal{C}_m^{\mathsf{d}}$, is defined as follows:

$$\mathcal{C}_m^{\mathsf{d}} = \{[\![0]\!], [\![1]\!], \ldots, [\![2m-1]\!]\}.$$

Let $[\![x]\!] \in \mathcal{C}_m^{\mathsf{d}}$ be a card holding a value $x \in \mathbb{Z}_{2m}$. For any constant $a \in \mathbb{Z}_{2m}$, a *rotation operation with a degree* a is defined as follows:

$$\mathsf{rot}^a([\![x]\!]) = [\![x + a]\!]$$

For any constant $a \in \mathbb{Z}_{2m}$, a *flipping operation with an axis a* is defined as follows:

$$\mathsf{flip}_a([\![x]\!]) = [\![-x + a]\!].$$

The transformation set of dihedral cards of modulus m, denoted by $\mathcal{T}^{\mathsf{d}}_m$, is defined as follows:

$$\mathcal{T}^{\mathsf{d}}_m = \{\mathsf{id}, \mathsf{rot}, \mathsf{rot}^2, \ldots, \mathsf{rot}^{2m-1}, \mathsf{flip}_0, \mathsf{flip}_1, \ldots, \mathsf{flip}_{2m-1}\}.$$

The symbol set of dihedral cards of modulus m, denoted by Σ^{d}_m, is defined as follows:

$$\Sigma^{\mathsf{d}}_m = \{\mathsf{?}\}.$$

The vision function $\mathsf{vis}^{\mathsf{d}}_m : \mathcal{C}^{\mathsf{d}}_m \to \Sigma^{\mathsf{d}}_m$ of dihedral cards of modulus m is defined as follows:

$$\mathsf{vis}^{\mathsf{d}}_m([\![x]\!]) = \mathsf{?} \quad \text{for any } x \in \mathbb{Z}_{2m}.$$

A card specification of dihedral cards of modulus m, denoted by $\mathsf{Dihedral}_m$, is defined as follows:

$$\mathsf{Dihedral}_m = (\mathcal{C}^{\mathsf{d}}_m, \mathcal{T}^{\mathsf{d}}_m, \Sigma^{\mathsf{d}}_m, \mathsf{vis}^{\mathsf{d}}_m).$$

Commitment. A *commitment to $x \in \mathbb{Z}_{2m}$* is defined by $[\![x]\!]$.

5.2.2 Operations for dihedral cards

As the model of cyclic cards, we use the following operations:

- Permutation $\mathsf{Perm}_{m,\ell}$;
- Shuffle $\mathsf{Shuf}_{m,\ell}$;
- Rotation $\mathsf{Rot}_{m,\ell}$;
- Rotation shuffle $\mathsf{RotShuf}_{m,\ell}$;
- Flower shuffle $\mathsf{Flower}_{m,\ell}$;

For dihedral cards, we additionally introduce five operations: *flipping*, *flipping shuffle*, *two-sided rotation shuffle*, *sign opening*, and *value opening*.

Flipping. A *flipping operation* is defined as follows:

$$(\mathsf{flip}, a, T),$$

where $a \in \mathbb{Z}_{2m}$ is an axis of flipping and $T \subset [\ell]$ is a subset of positions. By applying a flipping operation (flip, a, T), a sequence is converted as follows:

$$([\![x_1]\!], [\![x_2]\!], \ldots, [\![x_\ell]\!]) \;\to\; ([\![x'_1]\!], [\![x'_2]\!], \ldots, [\![x'_\ell]\!]),$$

where $x'_i = -x_i + a$ for all $i \in T$ and $x'_i = x_i$ for all $i \notin T$. For example, for a sequence $([\![0]\!], [\![2]\!], [\![5]\!], [\![7]\!])$ of modulus $m = 4$, a flipping operation $(\mathsf{flip}, 0, \{1, 2, 3, 4\})$ converts it into a new sequence $([\![0]\!], [\![6]\!], [\![3]\!], [\![1]\!])$. The set of flipping operations $\mathsf{Flip}_{m,\ell}$ is defined as follows:

$$\mathsf{Flip}_{m,\ell} = \{(\mathsf{flip}, j, T) \mid j \in \mathbb{Z}_{2m}, T \subset [\ell]\}.$$

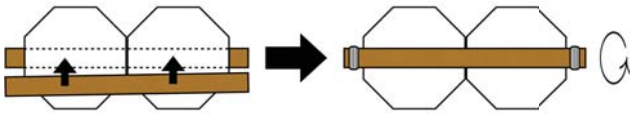
Flipping shuffle. A *flipping shuffle* is defined as follows:

$$(\mathsf{flipshuf}, (a_1, a_2, \ldots, a_k), T_1, T_2, \ldots, T_k),$$

where $k \in [\ell]$ is the number of axes, $a_1, a_2, \ldots, a_k \in \mathbb{Z}_{2m}$ are axes of flipping and $T_1, T_2, \ldots, T_k \subset [\ell]$ are disjoint subsets of positions. For all $1 \leq i \leq k$, all cards on T_i are flipped (by flip_{a_i}) randomly and simultaneously. Here, the random bit designating whether flipped or not is common for all i. Other cards are unchanged. For example, for a sequence $(\llbracket 0 \rrbracket, \llbracket 2 \rrbracket, \llbracket 5 \rrbracket, \llbracket 7 \rrbracket)$ of modulus $m = 4$, a flipping shuffle $(\mathsf{flipshuf}, (0, 1), \{1, 2\}, \{3, 4\})$ generates a new sequence:

$$(\llbracket 0 \rrbracket, \llbracket 2 \rrbracket, \llbracket 5 \rrbracket, \llbracket 7 \rrbracket) \;\rightarrow\; \begin{cases} (\llbracket 0 \rrbracket, \llbracket 2 \rrbracket, \llbracket 5 \rrbracket, \llbracket 7 \rrbracket) & \text{with probability } 1/2 \\ (\llbracket 0 \rrbracket, \llbracket 6 \rrbracket, \llbracket 4 \rrbracket, \llbracket 2 \rrbracket) & \text{with probability } 1/2 \end{cases}$$

A flipping shuffle is implemented by using two wooden boards as follows:



The set of flipping shuffles is defined as follows:

$$\begin{aligned} \mathsf{FlipShuf}_{m,\ell} = \{&(\mathsf{flipshuf}, (a_1, a_2, \ldots, a_k), T_1, T_2, \ldots, T_k) \mid \\ &k \in [\ell],\ a_1, a_2, \ldots, a_k \in \mathbb{Z}_{2m}, \\ &T_1, T_2, \ldots, T_k \subset [\ell] \text{ s.t. } \forall a, b \in [k], T_a \cap T_b = \emptyset\}. \end{aligned}$$

Two-sided rotation shuffle. A *two-sided rotation shuffle* is defined by:

$$(\mathsf{twoshuf}, T),$$

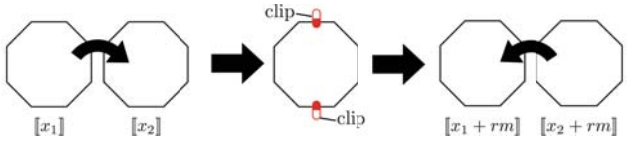
where $T \subset [\ell]$ is a subset of positions. By applying a two-sided rotation shuffle $(\mathsf{twoshuf}, T)$, a sequence is converted as follows:

$$(\llbracket x_1 \rrbracket, \llbracket x_2 \rrbracket, \ldots, \llbracket x_\ell \rrbracket) \;\rightarrow\; (\llbracket x'_1 \rrbracket, \llbracket x'_2 \rrbracket, \ldots, \llbracket x'_\ell \rrbracket),$$

where $x'_i = x_i + rm$ for a random bit $r \in \{0, 1\}$ if $i \in T$ and $x'_i = x_i$ otherwise. Note that the random bit r is common for all $i \in T$. For example, for a sequence $(\llbracket 0 \rrbracket, \llbracket 2 \rrbracket, \llbracket 5 \rrbracket, \llbracket 7 \rrbracket)$ of modulus $m = 4$, a two-sided rotation shuffle $(\mathsf{twoshuf}, \{1, 2, 3, 4\})$ generates a new sequence as follows:

$$(\llbracket 0 \rrbracket, \llbracket 2 \rrbracket, \llbracket 5 \rrbracket, \llbracket 7 \rrbracket) \;\rightarrow\; \begin{cases} (\llbracket 0 \rrbracket, \llbracket 2 \rrbracket, \llbracket 5 \rrbracket, \llbracket 7 \rrbracket) & \text{with probability } 1/2 \\ (\llbracket 4 \rrbracket, \llbracket 6 \rrbracket, \llbracket 0 \rrbracket, \llbracket 1 \rrbracket) & \text{with probability } 1/2 \end{cases}$$

A two-sided rotation shuffle is implemented by using two clips as follows:



The set of two-sided rotation shuffles is defined as follows:

$$\mathsf{TwoShuf}_{m,\ell} = \{(\mathsf{twoshuf}, T) \mid T \subset [\ell]\}.$$

Sign opening. A *sign opening* is defined as follows:

$$(\mathsf{sgnopen}, i),$$

where $i \in [\ell]$ is a position. For a sequence $(\llbracket x_1 \rrbracket, \llbracket x_2 \rrbracket, \ldots, \llbracket x_\ell \rrbracket)$, it publicly reveals a bit value $\mathsf{p}(x_i \geq m) \in \{0, 1\}$. It is treated as revealed information. That is, it outputs revealed information $r = \mathsf{p}(x_i \geq m)$ without changing the sequence. For example, for a sequence $(\llbracket 0 \rrbracket, \llbracket 2 \rrbracket, \llbracket 5 \rrbracket, \llbracket 7 \rrbracket)$ of modulus $m = 4$, a sign opening $(\mathsf{sgnopen}, 3)$ outputs the sign of the third card "1" $(\mathsf{p}(5 \geq 4))$ as revealed information. The set of sign openings is defined as follows:

$$\mathsf{SgnOpen}_{m,\ell} = \{(\mathsf{sgnopen}, i) \mid i \subset [\ell]\}.$$

Value opening. A *value opening* is defined as follows:

$$(\mathsf{valopen}, i),$$

where $i \in [\ell]$ is a position. For a sequence $(\llbracket x_1 \rrbracket, \llbracket x_2 \rrbracket, \ldots, \llbracket x_\ell \rrbracket)$, it publicly reveals a value $x_i \bmod m \in \mathbb{Z}_m$. It is treated as revealed information. That is, it outputs revealed information $r = (x_i \bmod m)$ without changing the sequence. For example, for a sequence $(\llbracket 0 \rrbracket, \llbracket 2 \rrbracket, \llbracket 5 \rrbracket, \llbracket 7 \rrbracket)$ of modulus $m = 4$, a value opening $(\mathsf{valopen}, 4)$ outputs the value of the fourth card "3" $(= 7 \bmod 4)$ as revealed information. The set of value openings is defined as follows:

$$\mathsf{ValOpen}_{m,\ell} = \{(\mathsf{valopen}, i) \mid i \subset [\ell]\}.$$

5.2.3 Notations

Hereafter, we use notations as follows.

Operations. We assume that the set of operations is $\mathcal{O}^{\mathsf{d}}_{m,\ell}$ defined as follows:

$$\mathcal{O}^{\mathsf{d}}_{m,\ell} = \mathsf{Perm}_{m,\ell} \cup \mathsf{Turn}_{m,\ell} \cup \mathsf{Shuf}_{m,\ell} \cup \mathsf{Rot}_{m,\ell} \cup \mathsf{RotShuf}_{m,\ell} \cup \mathsf{BackRot}_{m,\ell} \cup \mathsf{Flower}_{m,\ell}.$$

5.2.4 Outline of protocols

Our protocols based on dihedral cards in this chapter are as follows:

- Initialization protocol (Section 5.2.5): It takes a card $\llbracket x \rrbracket$ $(x \in \mathbb{Z}_{2m})$ as input and outputs a card $\llbracket 0 \rrbracket$.
- Addition protocol (Section 5.2.6): It takes two cards $(\llbracket x_1 \rrbracket, \llbracket x_2 \rrbracket)$ $(x_1, x_2 \in \mathbb{Z}_{2m})$ as input and outputs a card $\llbracket x_1 + x_2 \rrbracket$. The difference from the addition protocol #2 in Section 5.1.7 is that the protocol in this section uses a rotation shuffle while that in Section 5.1.7 uses a backward rotation shuffle.

- Sign normalization protocol (Section 5.2.7): It takes a card $[\![x]\!]$ ($x \in \mathbb{Z}_{2m}$) as input and outputs a card $[\![x \bmod m]\!]$.
- Sign-to-value protocol (Section 5.2.8): It takes two cards $([\![x]\!], [\![0]\!])$ ($x \in \mathbb{Z}_{2m}$) as input and outputs a card $[\![\mathsf{p}(x \geq m)]\!]$. It calls the initialization protocol once.
- Carry protocol (Section 5.2.9): It takes two cards $([\![x_1]\!], [\![x_2]\!])$ ($x_1, x_2 \in \mathbb{Z}_m$) as input and outputs a card $[\![\mathsf{p}(x_1 + x_2 \geq m)]\!]$. It calls the addition protocol once and the sign-to-value protocol once.
- Equality with zero protocol (Section 5.2.10): It takes two cards $([\![x]\!], [\![0]\!])$ ($x \in \mathbb{Z}_m$) as input and outputs a card $[\![\mathsf{p}(x = m)]\!]$. It calls the sign-to-value protocol once.
- Equality protocol (Section 5.2.11): It takes two cards $([\![x_1]\!], [\![x_2]\!])$ ($x_1, x_2 \in \mathbb{Z}_m$) as input and outputs a card $[\![\mathsf{p}(x_1 = x_2)]\!]$. It calls the subtraction protocol once, the sign normalization protocol once, and the equality with zero protocol once.
- Greater than protocol (Section 5.2.12): It takes two cards $([\![x_1]\!], [\![x_2]\!])$ ($x_1, x_2 \in \mathbb{Z}_m$) as input and outputs a card $[\![\mathsf{p}(x_1 \geq x_2)]\!]$. It calls the subtraction protocol once and the sign-to-value protocol once.

5.2.5 Initialization protocol

Functionality. A functionality $\mathcal{F}^{\mathsf{d}}_{\mathsf{init}}$ is defined as follows:

$$\mathcal{F}^{\mathsf{d}}_{\mathsf{init}} : [\![x]\!] \;\Rightarrow\; [\![0]\!].$$

where $x \in \mathbb{Z}_{2m}$.

Protocol. An initialization protocol $\mathcal{P}^{\mathsf{d}}_{\mathsf{init}}$ is defined as follows:

$$\mathcal{P}^{\mathsf{d}}_{\mathsf{init}} = (1, \mathbb{Z}_{2m}, (\mathsf{Dihedral}_m, \{[\![0]\!]\}), \mathcal{O}^{\mathsf{d}}_{m,1}, A).$$

It proceeds as follows:

1. $(\mathsf{rotshuf}, \{1\})$: Apply a rotation shuffle to it:

   $$[\![x]\!] \;\rightarrow\; [\![x']\!].$$

2. $(\mathsf{open}, 1)$: Apply an opening operation to it. Let $x' \in \mathbb{Z}_{2m}$ be the opened value, which is treated as revealed information.

   $$\text{revealed information } x'.$$

3. $(\mathsf{rot}, \{1\}, -x')$: Rotate it with a degree $-x'$ as follows:

   $$[\![x']\!] \;\rightarrow\; [\![0]\!]$$

   The protocol terminates.

Correctness. The correctness is trivial.

Security. Let $x \in \mathbb{Z}_{2m}$ be any input. The probability distribution of a view of the protocol starting with the sequence $s_{\mathsf{in}}(x) = [\![x]\!]$ is given as follows:

$$\mathsf{view}_{\mathcal{P}^{\mathsf{d}}_{\mathsf{init}}}(s_{\mathsf{in}}(x), x) = \big((?, \bot) \to (?, \bot) \to (?, x') \to (?, \bot)\big),$$

where $x' = x + r$ for a uniform random value $r \in \mathbb{Z}_{2m}$. This is equivalent to a probability distribution view^* defined as follows:

$$\mathsf{view}^* = \big((?, \bot) \to (?, \bot) \to (?, r') \to (?, \bot)\big),$$

where $r' \in \mathbb{Z}_{2m}$ is a uniform random value. The distribution view^* does not depend on x. Thus, for every $x, x' \in \mathbb{Z}_{2m}$, the following holds:

$$\mathsf{view}_{\mathcal{P}^{\mathsf{d}}_{\mathsf{init}}}(s_{\mathsf{in}}(x), x) = \mathsf{view}_{\mathcal{P}^{\mathsf{d}}_{\mathsf{init}}}(s_{\mathsf{in}}(x'), x') = \mathsf{view}^*.$$

Therefore, $\mathcal{P}^{\mathsf{d}}_{\mathsf{init}}$ securely realizes $\mathcal{F}^{\mathsf{d}}_{\mathsf{init}}$.

Efficiency. The number of cards is one. Note that this is the minimum number of cards. The number of probabilistic operations is one (one rotation shuffle).

5.2.6 Addition protocol

Functionality. A functionality $\mathcal{F}^{\mathsf{d}}_{\mathsf{add}}$ is defined as follows:

$$\mathcal{F}^{\mathsf{d}}_{\mathsf{add}} : ([\![x_1]\!], [\![x_2]\!]) \;\Rightarrow\; ([\![0]\!], [\![x_1 + x_2]\!])\,.$$

where $x_1, x_2 \in \mathbb{Z}_{2m}$.

Protocol. An addition protocol $\mathcal{P}^{\mathsf{d}}_{\mathsf{add}}$ is defined as follows:

$$\mathcal{P}^{\mathsf{d}}_{\mathsf{add}} = (2, \mathbb{Z}_{2m}, (\mathsf{Dihedral}_m, \{[\![0]\!], [\![0]\!]\}), \mathcal{O}^{\mathsf{d}}_{m,2}, A).$$

It proceeds as follows:

1. $(\mathsf{flip}, 0, \{1\})$: Flip the left card along with the 0-axis as follows:

$$([\![x_1]\!], [\![x_2]\!]) \;\to\; ([\![-x_1]\!], [\![x_2]\!]).$$

2. $(\mathsf{rotshuf}, \{1, 2\})$: Apply a rotation shuffle to them:

$$([\![-x_1]\!], [\![x_2]\!]) \;\to\; ([\![x'_1]\!], [\![x'_2]\!]).$$

3. $(\mathsf{sgnopen}, 1)$: Apply a sign opening operation to the left card. Let $s' \in \{0, 1\}$ be the opened value, which is treated as revealed information.

$$\text{revealed information } s'.$$

4. $(\mathsf{valopen}, 1)$: Apply a value opening operation to the left card. Let $v' \in \mathbb{Z}_m$ be the opened value, which is treated as revealed information.

$$\text{revealed information } v'.$$

5. $(\mathsf{rot}, \{1,2\}, -(s'm+v'))$: Rotate them so that they are added by $-(s'm+v')$:

$$([\![x'_1]\!], [\![x'_2]\!]) \ \rightarrow \ ([\![x'_1 - (s'm+v')]\!], [\![x'_2 - (s'm+v')]\!])$$

Correctness. By the rotation shuffle, $x'_1 = -x_1 + r$ and $x'_2 = x_2 + r$ for a uniform random value $r \in \mathbb{Z}_{2m}$. Since s' and v' are the sign and the value of x'_1, the degree of rotation $-(s'm+v')$ equals to $-x'_1$. Thus, the left card in the final sequence is $[\![x'_1 - (s'm+v')]\!] = [\![0]\!]$ and the right card in the final sequence is $[\![x'_2 - x'_1]\!] = [\![(x_2+r) - (-x_1+r)]\!] = [\![x_1+x_2]\!]$. Therefore, the above protocol $\mathcal{P}^{\mathsf{d}}_{\mathsf{add}}$ correctly realizes the functionality $\mathcal{F}^{\mathsf{d}}_{\mathsf{add}}$.

Security. Let $x = (x_1, x_2) \in (\mathbb{Z}_{2m})^2$ be any input. The probability distribution of a view of the protocol starting with the sequence $s_{\mathsf{in}}(x) = ([\![x_1]\!], [\![x_2]\!])$ is given as follows:

$$\mathsf{view}_{\mathcal{P}^{\mathsf{d}}_{\mathsf{add}}}(s_{\mathsf{in}}(x), x) = \big((?^2, \bot) \rightarrow (?^2, \bot) \rightarrow (?^2, \bot) \rightarrow (?^2, s') \rightarrow (?^2, v') \rightarrow (?^2, \bot)\big),$$

Since s' and v' are the sign and the value of $x'_1 = x_1 + r$ for a uniform random value $r \in \mathbb{Z}_{2m}$, the above distribution is equivalent to a probability distribution view^* defined as follows:

$$\mathsf{view}^* = \big((?^2, \bot) \rightarrow (?^2, \bot) \rightarrow (?^2, \bot) \rightarrow (?^2, r'_0) \rightarrow (?^2, r'_1) \rightarrow (?^2, \bot)\big).$$

where $r'_0 \in \{0,1\}$ and $r'_1 \in \mathbb{Z}_m$ are uniform random values. The distribution view^* does not depend on x. Thus, for every $x, x' \in \mathbb{Z}_{2m}$, the following holds:

$$\mathsf{view}_{\mathcal{P}^{\mathsf{d}}_{\mathsf{add}}}(s_{\mathsf{in}}(x), x) = \mathsf{view}_{\mathcal{P}^{\mathsf{d}}_{\mathsf{add}}}(s_{\mathsf{in}}(x'), x') = \mathsf{view}^*.$$

Therefore, $\mathcal{P}^{\mathsf{d}}_{\mathsf{add}}$ securely realizes $\mathcal{F}^{\mathsf{d}}_{\mathsf{add}}$.

Efficiency. The number of cards is two. Note that this is the minimum number of cards since the number of inputs is two. The number of probabilistic operations is one (one rotation shuffle).

5.2.7 Sign normalization protocol

Functionality. A functionality $\mathcal{F}^{\mathsf{d}}_{\mathsf{sign}}$ is defined as follows:

$$\mathcal{F}^{\mathsf{d}}_{\mathsf{sign}} : [\![x]\!] \ \Rightarrow \ [\![x \bmod m]\!],$$

where $x \in \mathbb{Z}_{2m}$.

Protocol. A protocol $\mathcal{P}^{\mathsf{d}}_{\mathsf{sign}}$ is defined as follows:

$$\mathcal{P}^{\mathsf{d}}_{\mathsf{sign}} = (1, \mathbb{Z}_{2m}, (\mathsf{Dihedral}_m, \{[\![0]\!]\}), \mathcal{O}^{\mathsf{d}}_{m,1}, A).$$

It proceeds as follows:

1. $(\mathsf{twoshuf}, \{1\})$: Apply a two-sided rotation shuffle to the input card as follows:
$$[\![x]\!] \to [\![x']\!],$$
where $x' = x + rm$ for a uniform random bit $r \in \{0, 1\}$.

2. $(\mathsf{sgnopen}, 1)$: Apply the sign opening to the card. Let $s' \in \{0, 1\}$ be the sign of the card, which is treated as revealed information.
$$[\![x']\!] \to [\![x']\!], \quad \text{revealed information } s'.$$

3. $(\mathsf{rot}, \{1\}, s'm)$: Rotate the card with a degree $s'm$:
$$[\![x']\!] \to [\![x' + s'm]\!].$$

Correctness. Let $x = v + sm$ for $v \in \mathbb{Z}_m$ and $s \in \{0, 1\}$. Due to the property of a two-sided rotation shuffle, x' is represented by $x' = v + (s \oplus r)m$ and s' is represented by $s' = s \oplus r$. Thus, the card in the final sequence is $[\![x' + s'm]\!] = [\![v + (s \oplus r)m + s'm]\!] = [\![v + (s \oplus r)m + (s \oplus r)m]\!] = [\![v]\!]$. (Note that every computation is done over $\mathbb{Z}_{2m}$.) Therefore, the above protocol $\mathcal{P}^{\mathsf{d}}_{\mathsf{sign}}$ correctly realizes the functionality $\mathcal{F}^{\mathsf{d}}_{\mathsf{sign}}$.

Security. Let $x = v + sm \in \mathbb{Z}_{2m}$ ($v \in \mathbb{Z}_m$ and $s \in \{0, 1\}$) be any input. The probability distribution of a view of the protocol starting with the sequence $s_{\mathsf{in}}(x) = [\![x]\!]$ is given as follows:

$$\mathsf{view}_{\mathcal{P}^{\mathsf{d}}_{\mathsf{sign}}}(s_{\mathsf{in}}(x), x) = \big((?, \bot) \to (?, s') \to (?, \bot) \to (?, \bot)\big),$$

where $s' = s \oplus r \in \{0, 1\}$ for a uniform random bit r. It is equivalent to a probability distribution view^* defined as follows:

$$\mathsf{view}^* = \big((?, \bot) \to (?, r') \to (?, \bot) \to (?, \bot)\big).$$

where $r' \in \{0, 1\}$ is a uniform random value. Thus, for every $x, x' \in \mathbb{Z}_{2m}$, the following holds:

$$\mathsf{view}_{\mathcal{P}^{\mathsf{d}}_{\mathsf{sign}}}(s_{\mathsf{in}}(x), x) = \mathsf{view}_{\mathcal{P}^{\mathsf{d}}_{\mathsf{sign}}}(s_{\mathsf{in}}(x'), x') = \mathsf{view}^*.$$

Therefore, $\mathcal{P}^{\mathsf{d}}_{\mathsf{sign}}$ securely realizes $\mathcal{F}^{\mathsf{d}}_{\mathsf{sign}}$.

Efficiency. The number of cards is one. Note that this is the minimum number of cards. The number of probabilistic operations is one (one two-sided rotation shuffle).

5.2.8 Sign-to-value protocol

Functionality. A functionality $\mathcal{F}_{\mathsf{sv}}^{\mathsf{d}}$ is defined as follows:

$$\mathcal{F}_{\mathsf{sv}}^{\mathsf{d}} : (\llbracket x \rrbracket, \llbracket 0 \rrbracket) \ \Rightarrow \ (\llbracket \mathsf{p}(x \geq m) \rrbracket, \llbracket 0 \rrbracket),$$

where $x \in \mathbb{Z}_{2m}$.

Protocol. A protocol $\mathcal{P}_{\mathsf{sv}}^{\mathsf{d}}$ is defined as follows:

$$\mathcal{P}_{\mathsf{sv}}^{\mathsf{d}} = (1, \mathbb{Z}_{2m}, (\mathsf{Dihedral}_m, \{\llbracket 0 \rrbracket, \llbracket 0 \rrbracket\}), \mathcal{O}_{m,2}^{\mathsf{d}} \cup \mathsf{Oracle}[\mathcal{P}_{\mathsf{init}}^{\mathsf{d}}], A).$$

It proceeds as follows:

1. $(\mathsf{twoshuf}, \{1\})$: Apply a two-sided rotation shuffle to the input card as follows:
$$(\llbracket x \rrbracket, \llbracket 0 \rrbracket) \ \rightarrow \ (\llbracket x + r_1 m \rrbracket, \llbracket r_1 m \rrbracket),$$
where $r_1 \in \{0, 1\}$ is a uniform random bit.

2. $(\mathsf{sgnopen}, 1)$: Apply the sign opening to the left card. Let $s_1 \in \{0, 1\}$ be the sign of the left card, which is treated as revealed information. (We can observe that $s_1 = \mathsf{p}(x \geq m) \oplus r_1$.)

3. $(\mathsf{rot}, \{2\}, s_1 m)$: Rotate the right card with a degree $s_1 m$:
$$(\llbracket x + r_1 m \rrbracket, \llbracket r_1 m \rrbracket) \ \rightarrow \ (\llbracket x + r_1 m \rrbracket, \llbracket (r_1 \oplus s_1) m \rrbracket).$$

4. $(\mathsf{oracle}, \mathcal{P}_{\mathsf{init}}^{\mathsf{d}}, \{1\})$: Apply the initialization protocol $\mathcal{P}_{\mathsf{init}}^{\mathsf{d}}$ as follows:
$$(\llbracket x + r_1 m \rrbracket, \llbracket (r_1 \oplus s_1) m \rrbracket) \ \rightarrow \ (\llbracket 0 \rrbracket, \llbracket (r_1 \oplus s_1) m \rrbracket).$$

5. $(\mathsf{flipshuf}, (\mathsf{flip}_1, \mathsf{flip}_m), (1, 2))$: Apply a flipping shuffle as follows:
$$(\llbracket 0 \rrbracket, \llbracket (r_1 \oplus s_1) m \rrbracket) \ \rightarrow \ (\llbracket r_2 \rrbracket, \llbracket (r_1 \oplus s_1 \oplus r_2) m \rrbracket),$$
where $r_2 \in \{0, 1\}$ is a uniform random bit.

6. $(\mathsf{sgnopen}, 2)$: Apply the sign opening to the right card. Let $s_2 \in \{0, 1\}$ be the sign of the right card, which is treated as revealed information. (We can observe that $s_2 = r_1 \oplus s_1 \oplus r_2$.) If $s_2 = 0$, the protocol terminates.

7. $(\mathsf{rot}, \{2\}, m)$: If $s_2 = 1$, rotate the right card with a degree m:
$$(\llbracket r_2 \rrbracket, \llbracket m \rrbracket) \ \rightarrow \ (\llbracket r_2 \rrbracket, \llbracket 0 \rrbracket).$$

8. $(\mathsf{flip}, 1, \{1\})$: If $s_2 = 1$, apply a flipping with an axis 1 as follows:
$$(\llbracket r_2 \rrbracket, \llbracket 0 \rrbracket) \ \rightarrow \ (\llbracket -r_2 + 1 \rrbracket, \llbracket 0 \rrbracket).$$
The protocol terminates.

Correctness. If $s_2 = 0$ at Step 6, the protocol terminates. In this case, the left card in the final sequence is given as follows:

$$[\![r_2]\!] = [\![r_1 \oplus s_1]\!] = [\![\mathsf{p}(x \geq m)]\!].$$

If $s_2 = 1$ at Step 6, the protocol proceeds to Step 8. In this case, the left card in the final sequence is given as follows:

$$[\![-r_2 + 1]\!] = [\![-(1 - r_1 \oplus s_1) + 1]\!] = [\![r_1 \oplus s_1]\!] = [\![\mathsf{p}(x \geq m)]\!].$$

Therefore, the above protocol $\mathcal{P}^{\mathsf{d}}_{\mathsf{sv}}$ correctly realizes the functionality $\mathcal{F}^{\mathsf{d}}_{\mathsf{sv}}$.

Security. Let $x = v + sm \in \mathbb{Z}_{2m}$ ($v \in \mathbb{Z}_m$ and $s \in \{0, 1\}$) be any input. The probability distribution of a view of the protocol starting with the sequence $s_{\mathsf{in}}(x) = ([\![x]\!], [\![0]\!])$ is given as follows:

$$\begin{aligned}\mathsf{view}_{\mathcal{P}^{\mathsf{d}}_{\mathsf{sv}}}(s_{\mathsf{in}}(x), x) = \Big(& (?^2, \bot) \to (?^2, \bot) \to (?^2, s_1) \to (?^2, \bot) \to (?^2, \bot) \\ & \to (?^2, \bot) \to (?^2, s_2) \big[\to (?^2, \bot) \to (?^2, \bot) \big]^{s_2} \Big),\end{aligned}$$

where $s_1 = \mathsf{p}(x \geq m) \oplus r_1 \in \{0, 1\}$ for a uniform random bit r_1, $s_2 = r_1 \oplus s_1 \oplus r_2 \in \{0, 1\}$ for a uniform random bit r_2, and the last two components "$\to (?^2, \bot) \to (?^2, \bot)$" appears only when $s_2 = 0$. It is equivalent to a probability distribution view^* defined as follows:

$$\begin{aligned}\mathsf{view}^* = \Big(& (?^2, \bot) \to (?^2, \bot) \to (?^2, r'_1) \to (?^2, \bot) \to (?^2, \bot) \\ & \to (?^2, \bot) \to (?^2, r'_2) \big[\to (?^2, \bot) \to (?^2, \bot) \big]^{r'_2} \Big),\end{aligned}$$

where $r'_1, r'_2 \in \{0, 1\}$ are uniform random bits and the last two components appears only when $r'_2 = 0$. Thus, for every $x, x' \in \mathbb{Z}_{2m}$, the following holds:

$$\mathsf{view}_{\mathcal{P}^{\mathsf{d}}_{\mathsf{sv}}}(s_{\mathsf{in}}(x), x) = \mathsf{view}_{\mathcal{P}^{\mathsf{d}}_{\mathsf{sv}}}(s_{\mathsf{in}}(x'), x') = \mathsf{view}^*.$$

Therefore, $\mathcal{P}^{\mathsf{d}}_{\mathsf{sv}}$ securely realizes $\mathcal{F}^{\mathsf{d}}_{\mathsf{sv}}$.

Efficiency. The number of cards is two. The number of oracle calls is one (one call of the initialization protocol). From Proposition 2.1, a sign-to-value protocol without oracles can be obtained. The number of probabilistic operations is three (one rotation shuffle, one two-sided rotation shuffle, and one flipping shuffle).

5.2.9 Carry protocol

Functionality. A functionality $\mathcal{F}^{\mathsf{d}}_{\mathsf{carry}}$ is defined as follows:

$$\mathcal{F}^{\mathsf{d}}_{\mathsf{carry}} = ([\![x_1]\!], [\![x_2]\!]) \;\Rightarrow\; ([\![\mathsf{p}(x_1 + x_2 \geq m)]\!], [\![0]\!]),$$

where $x_1, x_2 \in \mathbb{Z}_m$.

Protocol. A carry protocol $\mathcal{P}^{\mathsf{d}}_{\mathsf{carry}}$ is defined as follows:

$$\mathcal{P}^{\mathsf{d}}_{\mathsf{carry}} = (2, \mathbb{Z}_m, (\mathsf{Dihedral}_{2m}, \{[\![0]\!], [\![0]\!]\}), \mathcal{O}^{\mathsf{d}}_{2m,2} \cup \mathsf{Oracle}[\mathcal{P}^{\mathsf{d}}_{\mathsf{add}}, \mathcal{P}^{\mathsf{d}}_{\mathsf{sv}}], A).$$

It proceeds as follows:

1. $(\mathsf{oracle}, \mathcal{P}^{\mathsf{d}}_{\mathsf{add}}, \{1, 2\})$: Apply the addition protocol in Section 5.2.6 to the sequence as follows:

$$([\![x_1]\!], [\![x_2]\!]) \ \rightarrow \ ([\![x_1 + x_2]\!], [\![0]\!]).$$

2. $(\mathsf{oracle}, \mathcal{P}^{\mathsf{d}}_{\mathsf{sv}}, \{1\})$: Apply the sign-to-value protocol in Section 5.2.8 to the first card as follows:

$$([\![x_1 + x_2]\!], [\![0]\!]) \ \rightarrow \ ([\![\mathsf{p}(x_1 + x_2 \geq m)]\!], [\![0]\!]).$$

Correctness. The correctness is trivial.

Security. Let $x = (x_1, x_2) \in (\mathbb{Z}_m)^2$ be any input. The probability distribution of a view of the protocol starting with the sequence $s_{\mathsf{in}}(x) = ([\![x_1]\!], [\![x_2]\!])$ is given as follows:

$$\mathsf{view}_{\mathcal{P}^{\mathsf{d}}_{\mathsf{carry}}}(s_{\mathsf{in}}(x), x) = \big((?^2, \bot) \rightarrow (?^2, \bot) \rightarrow (?^2, \bot)\big).$$

It does not depend on x since it is just a fixed sequence. Thus, for every $x, x' \in (\mathbb{Z}_m)^2$, the following holds:

$$\mathsf{view}_{\mathcal{P}^{\mathsf{d}}_{\mathsf{carry}}}(s_{\mathsf{in}}(x), x) = \mathsf{view}_{\mathcal{P}^{\mathsf{d}}_{\mathsf{carry}}}(s_{\mathsf{in}}(x'), x').$$

Therefore, $\mathcal{P}^{\mathsf{d}}_{\mathsf{carry}}$ securely realizes $\mathcal{F}^{\mathsf{d}}_{\mathsf{carry}}$.

Efficiency. The number of cards is two. The number of oracle calls is two (one call of the addition protocol and one call of the sign-to-value protoocol). From Proposition 2.1, a carry protocol without oracles can be obtained. The number of probabilistic operations is four (two rotation shuffles, one two-sided rotation shuffle, and one flipping shuffle).

5.2.10 Equality with zero protocol

Functionality. A functionality $\mathcal{F}^{\mathsf{d}}_{\mathsf{zero}}$ is defined as follows:

$$\mathcal{F}^{\mathsf{d}}_{\mathsf{zero}} = ([\![x]\!], [\![0]\!]) \ \Rightarrow \ ([\![\mathsf{p}(x = 0)]\!], [\![0]\!]),$$

where $x \in \mathbb{Z}_m$.

Protocol. A carry protocol $\mathcal{P}^{\mathsf{d}}_{\mathsf{zero}}$ is defined as follows:

$$\mathcal{P}^{\mathsf{d}}_{\mathsf{zero}} = (1, \mathbb{Z}_m, (\mathsf{Dihedral}_{2m}, \{[\![0]\!], [\![0]\!]\}), \mathcal{O}^{\mathsf{d}}_{2m,2} \cup \mathsf{Oracle}[\mathcal{P}^{\mathsf{d}}_{\mathsf{sv}}], A).$$

It proceeds as follows:

1. $(\mathsf{flip}, 0, \{1\})$: Flip the first card along with the axis 0 as follows:

$$([\![x]\!], [\![0]\!]) \ \rightarrow \ ([\![2m - x]\!], [\![0]\!]).$$

2. $(\mathsf{oracle}, \mathcal{P}^{\mathsf{d}}_{\mathsf{sv}}, \{1\})$: Apply the sign-to-value protocol in Section 5.2.8 to the first card as follows:

$$([\![2m - x]\!], [\![0]\!]) \ \rightarrow \ ([\![s]\!], [\![0]\!]),$$

where $s = \mathsf{p}(2m - x \geq m)$.

3. $(\mathsf{flip}, 1, \{1\})$: Flip the first card along with the axis 1 as follows:

$$([\![s]\!], [\![0]\!]) \ \rightarrow \ ([\![-s + 1]\!], [\![0]\!]).$$

The protocol terminates.

Correctness. For any $x \in \mathbb{Z}_m$, it holds $\mathsf{p}(2m - x \geq m) = 0$ if and only if $x = 0$. Thus, the above protocol $\mathcal{P}^{\mathsf{d}}_{\mathsf{zero}}$ correctly realizes the functionality $\mathcal{F}^{\mathsf{d}}_{\mathsf{zero}}$.

Security. Let $x \in \mathbb{Z}_m$ be any input. The probability distribution of a view of the protocol starting with the sequence $s_{\mathsf{in}}(x) = ([\![x]\!], [\![0]\!])$ is given as follows:

$$\mathsf{view}_{\mathcal{P}^{\mathsf{d}}_{\mathsf{zero}}}(s_{\mathsf{in}}(x), x) = \big((\mathbf{?}^2, \bot) \rightarrow (\mathbf{?}^2, \bot) \rightarrow (\mathbf{?}^2, \bot)\big).$$

It does not depend on x since it is just a fixed sequence. Thus, for every $x, x' \in (\mathbb{Z}_m)^2$, the following holds:

$$\mathsf{view}_{\mathcal{P}^{\mathsf{d}}_{\mathsf{zero}}}(s_{\mathsf{in}}(x), x) = \mathsf{view}_{\mathcal{P}^{\mathsf{d}}_{\mathsf{zero}}}(s_{\mathsf{in}}(x'), x').$$

Therefore, $\mathcal{P}^{\mathsf{d}}_{\mathsf{zero}}$ securely realizes $\mathcal{F}^{\mathsf{d}}_{\mathsf{zero}}$.

Efficiency. The number of cards is two. The number of oracle calls is one (one call of the sign-to-value protocol). From Proposition 2.1, an equality with zero protocol without oracles can be obtained. The number of probabilistic operations is three (one rotation shuffle, one two-sided rotation shuffle, and one flipping shuffle).

5.2.11 Equality protocol

Functionality. A functionality $\mathcal{F}^{\mathsf{d}}_{\mathsf{equal}}$ is defined as follows:

$$\mathcal{F}^{\mathsf{d}}_{\mathsf{equal}} = ([\![x_1]\!], [\![x_2]\!]) \ \Rightarrow \ ([\![\mathsf{p}(x_1 = x_2)]\!], [\![0]\!]),$$

where $x_1, x_2 \in \mathbb{Z}_m$.

Protocol. A carry protocol $\mathcal{P}^{\mathsf{d}}_{\mathsf{equal}}$ is defined as follows:

$$\mathcal{P}^{\mathsf{d}}_{\mathsf{equal}} = (2, \mathbb{Z}_m, (\mathsf{Dihedral}_{2m}, \{[\![0]\!], [\![0]\!]\}), \mathcal{O}^{\mathsf{d}}_{2m,2} \cup \mathsf{Oracle}[\mathcal{P}^{\mathsf{d}}_{\mathsf{sub}}, \mathcal{P}^{\mathsf{d}}_{\mathsf{sign}}, \mathcal{P}^{\mathsf{d}}_{\mathsf{zero}}], A).$$

It proceeds as follows:

1. $(\mathsf{oracle}, \mathcal{P}^{\mathsf{d}}_{\mathsf{sub}}, \{1\})$: Apply the subtraction protocol to the sequence as follows:
$$([\![x_1]\!], [\![x_2]\!]) \ \rightarrow \ ([\![x_2 - x_1]\!], [\![0]\!]).$$

2. $(\mathsf{oracle}, \mathcal{P}^{\mathsf{d}}_{\mathsf{sign}}, \{1\})$: Apply the sign normalization protocol in Section 5.2.7 to the first card as follows:
$$([\![x_2 - x_1]\!], [\![0]\!]) \ \rightarrow \ ([\![z]\!], [\![0]\!]).$$

3. $(\mathsf{oracle}, \mathcal{P}^{\mathsf{d}}_{\mathsf{zero}}, \{1, 2\})$: Apply the equality with zero protocol in Section 5.2.10 as follows:
$$([\![z]\!], [\![0]\!]) \ \rightarrow \ ([\![\mathsf{p}(z = 0)]\!], [\![0]\!]).$$

Correctness. By the sign normalization protocol $\mathcal{P}^{\mathsf{d}}_{\mathsf{sign}}$, $z = x_2 - x_1 \bmod m$. Thus, the sequence $([\![z]\!], [\![0]\!])$ is matched with an oracle of $\mathcal{P}^{\mathsf{d}}_{\mathsf{zero}}$. We can also observe that $z = 0$ if and only if $x_1 = x_2$. Thus, the above protocol $\mathcal{P}^{\mathsf{d}}_{\mathsf{equal}}$ correctly realizes the functionality $\mathcal{F}^{\mathsf{d}}_{\mathsf{equal}}$.

Security. Let $x = (x_1, x_2) \in (\mathbb{Z}_m)^2$ be any input. The probability distribution of a view of the protocol starting with the sequence $s_{\mathsf{in}}(x) = ([\![x_1]\!], [\![x_2]\!])$ is given as follows:

$$\mathsf{view}_{\mathcal{P}^{\mathsf{d}}_{\mathsf{equal}}}(s_{\mathsf{in}}(x), x) = \big((\mathbf{?}^2, \bot) \rightarrow (\mathbf{?}^2, \bot) \rightarrow (\mathbf{?}^2, \bot) \rightarrow (\mathbf{?}^2, \bot)\big).$$

It does not depend on x since it is just a fixed sequence. Thus, for every $x, x' \in (\mathbb{Z}_m)^2$, the following holds:

$$\mathsf{view}_{\mathcal{P}^{\mathsf{d}}_{\mathsf{equal}}}(s_{\mathsf{in}}(x), x) = \mathsf{view}_{\mathcal{P}^{\mathsf{d}}_{\mathsf{equal}}}(s_{\mathsf{in}}(x'), x').$$

Therefore, $\mathcal{P}^{\mathsf{d}}_{\mathsf{equal}}$ securely realizes $\mathcal{F}^{\mathsf{d}}_{\mathsf{equal}}$.

Efficiency. The number of cards is two. The number of oracle calls is three (one call of the subtraction protocol, one call of the sign normalization protocol, and one call of the equality with zero protocol). From Proposition 2.1, an equality protocol without oracles can be obtained. The number of probabilistic operations is five (two rotation shuffles, two two-sided rotation shuffles, and one flipping shuffle).

5.2.12 Greater-than protocol

Functionality. A functionality $\mathcal{F}_{\mathsf{gr}}^{\mathsf{d}}$ is defined as follows:

$$\mathcal{F}_{\mathsf{gr}}^{\mathsf{d}} = (\llbracket x_1 \rrbracket, \llbracket x_2 \rrbracket) \;\Rightarrow\; (\llbracket \mathsf{p}(x_2 \geq x_1) \rrbracket, \llbracket 0 \rrbracket),$$

where $x_1, x_2 \in \mathbb{Z}_m$.

Protocol. A carry protocol $\mathcal{P}_{\mathsf{gr}}^{\mathsf{d}}$ is defined as follows:

$$\mathcal{P}_{\mathsf{gr}}^{\mathsf{d}} = (2, \mathbb{Z}_m, (\mathsf{Dihedral}_{2m}, \{\llbracket 0 \rrbracket, \llbracket 0 \rrbracket\}), \mathcal{O}_{2m,2}^{\mathsf{d}} \cup \mathsf{Oracle}[\mathcal{P}_{\mathsf{sub}}^{\mathsf{d}}, \mathcal{P}_{\mathsf{sv}}^{\mathsf{d}}], A).$$

It proceeds as follows:

1. $(\mathsf{oracle}, \mathcal{P}_{\mathsf{sub}}^{\mathsf{d}}, \{1, 2\})$: Apply the subtraction protocol in Section 5.2.6 to the sequence as follows:
$$(\llbracket x_1 \rrbracket, \llbracket x_2 \rrbracket) \;\to\; (\llbracket x_2 - x_1 \rrbracket, \llbracket 0 \rrbracket).$$
2. $(\mathsf{oracle}, \mathcal{P}_{\mathsf{sv}}^{\mathsf{d}}, \{1, 2\})$: Apply the sign-to-value protocol in Section 5.2.8 as follows:
$$(\llbracket x_2 - x_1 \rrbracket, \llbracket 0 \rrbracket) \;\to\; (\llbracket 1 - \mathsf{p}(x_2 \geq x_1) \rrbracket, \llbracket 0 \rrbracket).$$
3. $(\mathsf{flip}, 1, \{1\})$: Flip the first card along with the axis 1 as follows:
$$(\llbracket 1 - \mathsf{p}(x_2 \geq x_1) \rrbracket, \llbracket 0 \rrbracket) \;\to\; (\llbracket \mathsf{p}(x_2 \geq x_1) \rrbracket, \llbracket 0 \rrbracket).$$
The protocol terminates.

Correctness. The correctness is trivial.

Security. Let $x = (x_1, x_2) \in (\mathbb{Z}_m)^2$ be any input. The probability distribution of a view of the protocol starting with the sequence $s_{\mathsf{in}}(x) = (\llbracket x_1 \rrbracket, \llbracket x_2 \rrbracket)$ is given as follows:

$$\mathsf{view}_{\mathcal{P}_{\mathsf{gr}}^{\mathsf{d}}}(s_{\mathsf{in}}(x), x) = \big((\mathbf{?}^2, \bot) \to (\mathbf{?}^2, \bot) \to (\mathbf{?}^2, \bot) \to (\mathbf{?}^2, \bot)\big).$$

It does not depend on x since it is just a fixed sequence. Thus, for every $x, x' \in (\mathbb{Z}_m)^2$, the following holds:

$$\mathsf{view}_{\mathcal{P}_{\mathsf{gr}}^{\mathsf{d}}}(s_{\mathsf{in}}(x), x) = \mathsf{view}_{\mathcal{P}_{\mathsf{gr}}^{\mathsf{d}}}(s_{\mathsf{in}}(x'), x').$$

Therefore, $\mathcal{P}_{\mathsf{gr}}^{\mathsf{d}}$ securely realizes $\mathcal{F}_{\mathsf{gr}}^{\mathsf{d}}$.

Efficiency. The number of cards is two. The number of oracle calls is two (one call of the subtraction protocol and one call of the sign-to-value protocol). From Proposition 2.1, a greater than protocol without oracles can be obtained. The number of probabilistic operations is four (two rotation shuffles, one two-sided rotation shuffle, and one flipping shuffle).

Chapter 6

Conclusion

In this dissertation, we have studied easy to perform card-based protocols.

In Chapter 3, we have studied protocols with uniform closed shuffles. In particular, we have provided a general protocol with a single uniform closed shuffle. An important open problem involves minimizing the number of cards in a general protocol with a single uniform closed shuffle. From the aspect of being easy to perform, it is also important to consider a similar problem in the case of a restricted shuffle such as a random cut, random bisection cut, and pile-scramble shuffle. Although we have obtained a general protocol with two pile-scramble shuffles that is moderately efficient in terms of the number of cards, there lacks an efficient construction in the case of random cuts and random bisection cuts.

In Chapter 4, we have studied protocols based on private permutations. We have solved the security problem that is inherent to private actions by defining the active security and introducing the commit-and-prove technique. In particular, we have given a $2n + 7$-card protocol with active security for any function $f : \{0, 1\}^n \to \{0, 1\}$. A natural question is whether the construction is optimal or not in terms of the number of cards. From the aspect of being easy to perform, it is important to study protocols with a small number of both cards and private permutations.

In Chapter 5, we have introduced cyclic cards and dihedral cards and constructed protocols based on them. Due to the power of partial openings, we have developed efficient protocols for interesting predicates including the carry of addition, the equality predicate, and the greater-than predicate. One problem for future work is to construct other efficient protocols using cyclic cards and dihedral cards. Another interesting problem is to design a new card that enables efficient computation of interesting functions.

Bibliography

[1] Yoshiki Abe, Mitsugu Iwamoto, and Kazuo Ohta. Efficient private PEZ protocols for symmetric functions. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part I*, volume 11891 of *Lecture Notes in Computer Science*, pages 372–392. Springer, 2019.

[2] Yuta Abe, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. Five-card AND protocol in committed format using only practical shuffles. In *Proceedings of the 5th ACM on ASIA Public-Key Cryptography Workshop, APKC@AsiaCCS, Incheon, Republic of Korea, June 4, 2018*, pages 3–8, 2018.

[3] József Balogh, János A. Csirik, Yuval Ishai, and Eyal Kushilevitz. Private computation using a PEZ dispenser. *Theor. Comput. Sci.*, 306(1-3):69–84, 2003.

[4] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc^1. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 1–5, 1986.

[5] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 784–796, 2012.

[6] Xavier Bultel, Jannik Dreier, Jean-Guillaume Dumas, and Pascal Lafourcade. Physical zero-knowledge proofs for akari, takuzu, kakuro and kenken. In Erik D. Demaine and Fabrizio Grandoni, editors, *8th International Conference on Fun with Algorithms, FUN 2016, June 8-10, 2016, La Maddalena, Italy*, volume 49 of *LIPIcs*, pages 8:1–8:20. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

[7] Xavier Bultel, Jannik Dreier, Jean-Guillaume Dumas, Pascal Lafourcade, Daiki Miyahara, Takaaki Mizuki, Atsuki Nagao, Tatsuya Sasaki, Kazumasa Shinagawa, and Hideaki Sone. Physical zero-knowledge proof for makaro. In Taisuke Izumi and Petr Kuznetsov, editors, *Stabilization, Safety, and*

Security of Distributed Systems - 20th International Symposium, SSS 2018, Tokyo, Japan, November 4-7, 2018, Proceedings, volume 11201 of *Lecture Notes in Computer Science*, pages 111–125. Springer, 2018.

[8] Eddie Cheung, Chris Hawthorne, and Patrick Lee. Cs 758 project: Secure computation with playing cards, 2013. `https://csclub.uwaterloo.ca/~cdchawth/files/papers/secure_playing_cards.pdf`.

[9] Claude Crépeau and Joe Kilian. Discreet solitary games. In *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, pages 319–330, 1993.

[10] Bert den Boer. More efficient match-making and satisfiability: *The Five Card Trick*. In *Advances in Cryptology - EUROCRYPT '89, Workshop on the Theory and Application of of Cryptographic Techniques, Houthalen, Belgium, April 10-13, 1989, Proceedings*, pages 208–217, 1989.

[11] Jean-Guillaume Dumas, Pascal Lafourcade, Daiki Miyahara, Takaaki Mizuki, Tatsuya Sasaki, and Hideaki Sone. Interactive physical zero-knowledge proof for norinori. In Ding-Zhu Du, Zhenhua Duan, and Cong Tian, editors, *Computing and Combinatorics - 25th International Conference, COCOON 2019, Xi'an, China, July 29-31, 2019, Proceedings*, volume 11653 of *Lecture Notes in Computer Science*, pages 166–177. Springer, 2019.

[12] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229, 1987.

[13] Ronen Gradwohl, Moni Naor, Benny Pinkas, and Guy N. Rothblum. Cryptographic and physical zero-knowledge proof systems for solutions of sudoku puzzles. In *Fun with Algorithms, 4th International Conference, FUN 2007, Castiglioncello, Italy, June 3-5, 2007, Proceedings*, pages 166–182, 2007.

[14] Yuji Hashimoto, Koji Nuida, Kazumasa Shinagawa, Masaki Inamura, and Goichiro Hanaoka. Toward finite-runtime card-based protocol for generating a hidden random permutation without fixed points. *IEICE Transactions*, 101-A(9):1503–1511, 2018.

[15] Yuji Hashimoto, Kazumasa Shinagawa, Koji Nuida, Masaki Inamura, and Goichiro Hanaoka. Secure grouping protocol using a deck of cards. In *Information Theoretic Security - 10th International Conference, ICITS 2017, Hong Kong, China, November 29 - December 2, 2017, Proceedings*, pages 135–152, 2017.

[16] James Heather, Steve Schneider, and Vanessa Teague. Cryptographic protocols with everyday objects. *Formal Asp. Comput.*, 26(1):37–62, 2014.

[17] Rie Ishikawa, Eikoh Chida, and Takaaki Mizuki. Efficient card-based protocols for generating a hidden random permutation without fixed points. In *Unconventional Computation and Natural Computation - 14th International Conference, UCNC 2015, Auckland, New Zealand, August 30 - September 3, 2015, Proceedings*, pages 215–226, 2015.

[18] Julia Kastner, Alexander Koch, Stefan Walzer, Daiki Miyahara, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. The minimum number of cards in practical card-based protocols. In *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part III*, pages 126–155, 2017.

[19] Alexander Koch. The landscape of optimal card-based protocols. *IACR Cryptology ePrint Archive*, 2018:951, 2018.

[20] Alexander Koch, Michael Schrempp, and Michael Kirsten. Card-based cryptography meets formal verification. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I*, volume 11921 of *Lecture Notes in Computer Science*, pages 488–517. Springer, 2019.

[21] Alexander Koch, Stefan Walzer, and Kevin Härtel. Card-based cryptographic protocols using a minimal number of cards. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part I*, pages 783–807, 2015.

[22] Pascal Lafourcade, Daiki Miyahara, Takaaki Mizuki, Tatsuya Sasaki, and Hideaki Sone. A physical ZKP for slitherlink: How to perform physical topology-preserving computation. In Swee-Huay Heng and Javier López, editors, *Information Security Practice and Experience - 15th International Conference, ISPEC 2019, Kuala Lumpur, Malaysia, November 26-28, 2019, Proceedings*, volume 11879 of *Lecture Notes in Computer Science*, pages 135–151. Springer, 2019.

[23] Pascal Lafourcade, Takaaki Mizuki, Atsuki Nagao, and Kazumasa Shinagawa. Light cryptography. In Lynette Drevin and Marianthi Theocharidou, editors, *Information Security Education. Education in Proactive Information Security - 12th IFIP WG 11.8 World Conference WISE 12, Lisbon, Portugal, June 25-27, 2019, Proceedings*, volume 557 of *IFIP Advances in Information and Communication Technology*, pages 89–101. Springer, 2019.

[24] Antonio Marcedone, Zikai Wen, and Elaine Shi. Secure dating with four or fewer cards. Cryptology ePrint Archive, Report 2015/1031, 2015.

[25] Daiki Miyahara, Tatsuya Sasaki, Takaaki Mizuki, and Hideaki Sone. Card-based physical zero-knowledge proof for kakuro. *IEICE Transactions*, 102-A(9):1072–1078, 2019.

[26] Takaaki Mizuki. Applications of card-based cryptography to education. *IEICE Technical Report*, 116(289):13–17, 2016.

[27] Takaaki Mizuki. Card-based protocols for securely computing the conjunction of multiple variables. *Theor. Comput. Sci.*, 622:34–44, 2016.

[28] Takaaki Mizuki. Efficient and secure multiparty computations using a standard deck of playing cards. In *Cryptology and Network Security - 15th International Conference, CANS 2016, Milan, Italy, November 14-16, 2016, Proceedings*, pages 484–499, 2016.

[29] Takaaki Mizuki, Isaac Kobina Asiedu, and Hideaki Sone. Voting with a logarithmic number of cards. In *Unconventional Computation and Natural Computation - 12th International Conference, UCNC 2013, Milan, Italy, July 1-5, 2013. Proceedings*, pages 162–173, 2013.

[30] Takaaki Mizuki, Yoshinori Kugimoto, and Hideaki Sone. Secure multiparty computations using a dial lock. In *Theory and Applications of Models of Computation, 4th International Conference, TAMC 2007, Shanghai, China, May 22-25, 2007, Proceedings*, pages 499–510, 2007.

[31] Takaaki Mizuki, Michihito Kumamoto, and Hideaki Sone. The five-card trick can be done with four cards. In *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, pages 598–606, 2012.

[32] Takaaki Mizuki and Hiroki Shizuya. Practical card-based cryptography. In *Fun with Algorithms - 7th International Conference, FUN 2014, Lipari Island, Sicily, Italy, July 1-3, 2014. Proceedings*, pages 313–324, 2014.

[33] Takaaki Mizuki and Hideaki Sone. Six-card secure AND and four-card secure XOR. In *Frontiers in Algorithmics, Third International Workshop, FAW 2009, Hefei, China, June 20-23, 2009. Proceedings*, pages 358–369, 2009.

[34] Takaaki Mizuki, Fumishige Uchiike, and Hideaki Sone. Securely computing XOR with 10 cards. *The Australasian Journal of Combinatorics*, 36:279–293, 2006.

[35] Takeshi Nakai, Satoshi Shirouchi, Mitsugu Iwamoto, and Kazuo Ohta. Four cards are sufficient for a card-based three-input voting protocol utilizing private permutations. In *Information Theoretic Security - 10th International Conference, ICITS 2017, Hong Kong, China, November 29 - December 2, 2017, Proceedings*, pages 153–165, 2017.

[36] Takeshi Nakai, Yuuki Tokushige, Yuto Misawa, Mitsugu Iwamoto, and Kazuo Ohta. Efficient card-based cryptographic protocols for millionaires' problem utilizing private permutations. In *Cryptology and Network Security - 15th International Conference, CANS 2016, Milan, Italy, November 14-16, 2016, Proceedings*, pages 500–517, 2016.

[37] Moni Naor and Adi Shamir. Visual cryptography. In *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, pages 1–12, 1994.

[38] Valtteri Niemi and Ari Renvall. Secure multiparty computations without computers. *Theor. Comput. Sci.*, 191(1-2):173–183, 1998.

[39] Valtteri Niemi and Ari Renvall. Solitaire zero-knowledge. *Fundam. Inform.*, 38(1-2):181–188, 1999.

[40] Takuya Nishida, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. Card-based protocols for any boolean function. In *Theory and Applications of Models of Computation - 12th Annual Conference, TAMC 2015, Singapore, May 18-20, 2015, Proceedings*, pages 110–121, 2015.

[41] Takuya Nishida, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. Securely computing three-input functions with eight cards. *IEICE Transactions*, 98-A(6):1145–1152, 2015.

[42] Takuya Nishida, Takaaki Mizuki, and Hideaki Sone. Securely computing the three-input majority function with eight cards. In *Theory and Practice of Natural Computing - Second International Conference, TPNC 2013, Cáceres, Spain, December 3-5, 2013, Proceedings*, pages 193–204, 2013.

[43] Akihiro Nishimura, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. An implementation of non-uniform shuffle for secure multi-party computation. In *Proceedings of the 3rd ACM International Workshop on ASIA Public-Key Cryptography, AsiaPKC@AsiaCCS, Xi'an, China, May 30 - June 03, 2016*, pages 49–55, 2016.

[44] Akihiro Nishimura, Takuya Nishida, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. Five-card secure computations using unequal division shuffle. In *Theory and Practice of Natural Computing - Fourth International Conference, TPNC 2015, Mieres, Spain, December 15-16, 2015. Proceedings*, pages 109–120, 2015.

[45] Hibiki Ono and Yoshifumi Manabe. Card-based cryptographic protocols with the minimum number of cards using private operations. In A. Nur Zincir-Heywood, Guillaume Bonfante, Mourad Debbabi, and Joaquín García-Alfaro, editors, *Foundations and Practice of Security - 11th International Symposium, FPS 2018, Montreal, QC, Canada, November 13-15, 2018, Revised Selected Papers*, volume 11358 of *Lecture Notes in Computer Science*, pages 193–207. Springer, 2018.

[46] Hibiki Ono and Yoshifumi Manabe. Efficient card-based cryptographic protocols for the millionaires' problem using private input operations. In *13th Asia Joint Conference on Information Security, AsiaJCIS 2018, Guilin, China, August 8-9, 2018*, pages 23–28. IEEE Computer Society, 2018.

[47] Hibiki Ono and Yoshifumi Manabe. Card-based cryptographic protocols with the minimum number of rounds using private operations. In Cristina Pérez-Solà, Guillermo Navarro-Arribas, Alex Biryukov, and Joaquín García-Alfaro, editors, *Data Privacy Management, Cryptocurrencies and Blockchain Technology - ESORICS 2019 International Workshops, DPM 2019 and CBT 2019, Luxembourg, September 26-27, 2019, Proceedings*, volume 11737 of *Lecture Notes in Computer Science*, pages 156–173. Springer, 2019.

[48] Tatsuya Sasaki, Takaaki Mizuki, and Hideaki Sone. Card-based zero-knowledge proof for sudoku. In *9th International Conference on Fun with Algorithms, FUN 2018, June 13-15, 2018, La Maddalena, Italy*, pages 29:1–29:10, 2018.

[49] Kazumasa Shinagawa. Deterministic cryptographic protocols with active security using a deck of cards, envelopes and chains. In *2018 Symposium on Cryptography and Information Security, SCIS 2018, Niigata, Japan, January 23-26, 2018, Proceedings*, 2018.

[50] Kazumasa Shinagawa. Card-based cryptographic protocols based on private transpositions. In *2019 Symposium on Cryptography and Information Security, SCIS 2019, Shiga, Japan, January 22-25, 2018, Proceedings*, 2019.

[51] Kazumasa Shinagawa. Card-based cryptography with invisible ink. In T. V. Gopal and Junzo Watada, editors, *Theory and Applications of Models of Computation - 15th Annual Conference, TAMC 2019, Kitakyushu, Japan, April 13-16, 2019, Proceedings*, volume 11436 of *Lecture Notes in Computer Science*, pages 566–577. Springer, 2019.

[52] Kazumasa Shinagawa and Takaaki Mizuki. Card-based protocols using triangle cards. In *9th International Conference on Fun with Algorithms, FUN 2018, June 13-15, 2018, La Maddalena, Italy*, pages 31:1–31:13, 2018.

[53] Kazumasa Shinagawa and Takaaki Mizuki. The six-card trick: Secure computation of three-input equality. In *Information Security and Cryptology - ICISC 2018 - 21st International Conference, Seoul, South Korea, November 28-30, 2018, Revised Selected Papers*, pages 123–131, 2018.

[54] Kazumasa Shinagawa and Takaaki Mizuki. Secure computation of any boolean function based on any deck of cards. In Yijia Chen, Xiaotie Deng, and Mei Lu, editors, *Frontiers in Algorithmics - 13th International Workshop, FAW 2019, Sanya, China, April 29 - May 3, 2019, Proceedings*, volume 11458 of *Lecture Notes in Computer Science*, pages 63–75. Springer, 2019.

[55] Kazumasa Shinagawa, Takaaki Mizuki, Jacob C. N. Schuldt, Koji Nuida, Naoki Kanayama, Takashi Nishide, Goichiro Hanaoka, and Eiji Okamoto. Multi-party computation with small shuffle complexity using regular polygon cards. In *Provable Security - 9th International Conference, ProvSec 2015, Kanazawa, Japan, November 24-26, 2015, Proceedings*, pages 127–146, 2015.

[56] Kazumasa Shinagawa, Takaaki Mizuki, Jacob C. N. Schuldt, Koji Nuida, Naoki Kanayama, Takashi Nishide, Goichiro Hanaoka, and Eiji Okamoto. Secure multi-party computation using polarizing cards. In *Advances in Information and Computer Security - 10th International Workshop on Security, IWSEC 2015, Nara, Japan, August 26-28, 2015, Proceedings*, pages 281–297, 2015.

[57] Kazumasa Shinagawa, Takaaki Mizuki, Jacob C. N. Schuldt, Koji Nuida, Naoki Kanayama, Takashi Nishide, Goichiro Hanaoka, and Eiji Okamoto. Card-based protocols using regular polygon cards. *IEICE Transactions*, 100-A(9):1900–1909, 2017.

[58] Kazumasa Shinagawa and Koji Nuida. A single shuffle is enough for secure card-based computation of any circuit. *IACR Cryptology ePrint Archive*, 2019:380, 2019.

[59] Anton Stiglic. Computations with a deck of cards. *Theor. Comput. Sci.*, 259(1-2):671–678, 2001.

[60] Ken Takashima, Yuta Abe, Tatsuya Sasaki, Daiki Miyahara, Kazumasa Shinagawa, Takaaki Mizuki, and Hideaki Sone. Card-based secure ranking computations. In *Combinatorial Optimization and Applications - 13th International Conference, COCOA 2019, Xiamen, China, December 13-15, 2019, Proceedings*, pages 461–472, 2019.

[61] Itaru Ueda, Akihiro Nishimura, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. How to implement a random bisection cut. In *Theory and Practice of Natural Computing - 5th International Conference, TPNC 2016, Sendai, Japan, December 12-13, 2016, Proceedings*, pages 58–69, 2016.

[62] Yohei Watanabe, Yoshihisa Kuroki, Shinnosuke Suzuki, Yuta Koga, Mitsugu Iwamoto, and Kazuo Ohta. Card-based majority voting protocols with three inputs using three cards. In *International Symposium on Information Theory and Its Applications, ISITA 2018, Singapore, October 28-31, 2018*, pages 218–222. IEEE, 2018.

[63] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 160–164, 1982.

[64] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167, 1986.